Računarska geometrija

Predrag Janičić

RAČUNARSKA GEOMETRIJA

Beograd
2016.

Autor:
dr Predrag Janičić, redovni profesor na Matematičkom fakultetu u Beogradu

RAČUNARSKA GEOMETRIJA

Matematički fakultet Univerziteta u Beogradu
Studentski trg 16, 11000 Beograd

...

Obrada teksta, crteži i korice: *autor*

Predgovor

Ovo su beleške koji prate predavanja za predmet *Geometrijski algoritmi* na master studijama računarstva i informatike na Matematičkom fakultetu (http://www.math.rs/files/R313_-_Geometrijski_algoritmi.pdf.

To je zapravo kurs računarske geometrije (koji se na mnogim univerzitetima obično drži pod imenom „Computational geometry"), pa je i ovaj materijal naslovljen *Računarska geometrija* (mada bi i naslov *Računska geometrija* bio adekvatan). Kurs se bavi dizajnom i analizom efikasnih algoritama za geometrijske probleme, obično u ravni ili prostoru. Ovi problemi važni su u mnogim oblastima primene, uključujući geografske informacione sisteme, računarsku grafiku, geometrijsko modelovanje, itd. Cilj kursa je da upozna studente sa važnim tehnikama i rezultatima računarske geometrije i da im omogući savladavanje teorijskih i praktičnih problema iz raznih oblasti.

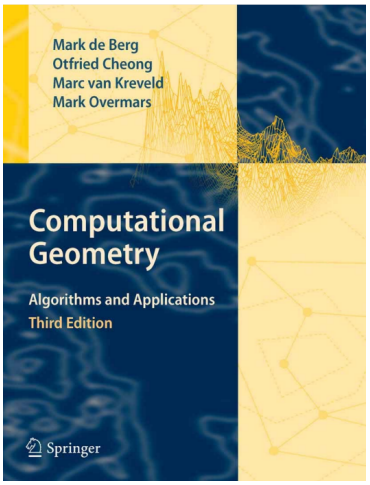
Teme uključuju konveksni omotač, algoritme zasnovane na brišućoj pravoj, Voronoj dijagrame, Delune triangulacije, pseudotriangulacije, linearno programiranje, raspoređivanje, podele ravni i prostora, kvodtri, oktri, modelovanje tela, modelovanje terena, planiranje kretanja itd. Kurs je koncipiran po ugledu na slične kurseve na drugim univerzitetima, kao, na primer,

- U Utrecht (Holandija) http://www.cs.uu.nl/docs/vakken/ga/
- MIT (SAD): https://people.csail.mit.edu/indyk/6.838/
- U Berkley (SAD): http://www.cs.berkeley.edu/~jrs/274/
- ETH (Zirih, Svacarska): http://www.ti.inf.ethz.ch/ew/Lehre/CG13/index.html#description
- U Santa Barbara (SAD): https://www.cs.ucsb.edu/~suri/cs235/235.html
- U AArhus (Danska): http://www.cs.au.dk/~gerth/cg10/

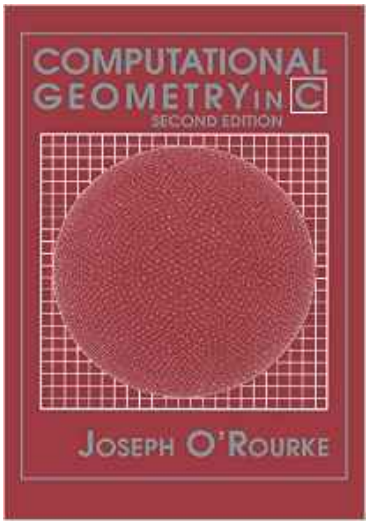
- U Maryland (SAD): http://www.cs.umd.edu/~mount/754/Lects/754lects.pdf

Osnovna literatura:

- Mark de Berg, Otfried Cheong, Marc van Kreveld, Mark Overmars: Computational Geometry: Algorithms and Applications, Springer; 3rd edition (April 16, 2008)



- Joseph O'Rourke: Computational Geometry in C, Cambridge University Press; 2 edition, 1998.



- CGAL biblioteka: http://www.cgal.org/

Sadržaj

Glava 1

Informacije o kursu

Vežbe:

- GCLC
- Qt i OpenGL
- algoritmi sa predavanja i primene

Ispit – sa predispitnim obavezama:

- Projekat u GCLC-u: 10 poena
- Projekat (implementacija nekog algoritma): 30 poena
- Predstavljanje naučnog rada: 10 poena
- Rad na vežbama: 10 poena
- Završni teorijski test: 40 poena *(prag za prolaz: 16 poena)*

ili – bez predispitnih obaveza:

- Završni praktični ispit na računaru: 50 poena *(prag za prolaz: 20 poena)*
- Završni teorijski test: 50 poena *(prag za prolaz: 20 poena)*

Glava 2

Uvod

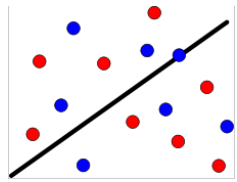
Računarska geometrija potomak je klasične geometrije i računarstva. Ona je deo oblasti algoritmike i bavi se razvojem i analizom efikasnih algoritama i struktura podataka pogodnih za geometrijske probleme. Posebno su važni problemi kod kojih rešenje grubom silom nije praktično upotrebljivo.

Računarska geometrija razvija se zahvaljujući problemima i primenama, pre svega u računarskoj grafici, računarskoj viziji, robotici, bazama podataka, geografskim informacionim sistemima, CAD/CAM sistemima, molekularnoj biologiji, itd. Neke od konkretnih primena su primene u virtuelnoj stvarnosti, planiranju kretanja, detektovanju sudara, dizajnu lekova, dinamici fluida, itd.

Oblast računarske geometrije obično se bavi problemima u euklidskoj ravni ili prostoru i podrazumeva raspoloživost elementarnih operacija kao što su: provera da li tačka pripada pravoj/krugu, da li se prave-duži seku i slično.

Računska geometrija često se oslanja na kombinatornu geometriju i njena tvrđenja, poput sledećih primera:

- Helijeva teorema: ako svake tri konveksne figure iz datog skupa ravni imaju presek, onda postoji i zajednički presek za sve figure.
- Teorema o sendviču (diskretna varijanta): ako je dato n crvenih i n plavih tačaka u ravni, postoji prava koja razdvaja i skup crvenih i skup plavi tačaka na delove jednake brojnosti.



- Erdoš-Sekerešova teorema (uopštenje "happy-ending" teoreme): za svaki prirodan broj k postoji prirodan broj F_k takav da svaki skup od F_k u ravni

sadrži k tačaka koje čine konveksan poligon. Hipoteza: $F_k = 1 + 2^{k-2}$, za $k \geq 3$.

2.1 Oblasti primena

Računska geometrija vođena je primenama u širokom skupu oblasti. U nastavku teksta biće ukratko opisane samo neke od njih.

2.1.1 Računarska grafika

Jedan od centralnih problema računarske grafike je kreiranje slika modelovane scene. Često je cilj kreiranje foto-realističnih slika, te odgovarajući modeli imaju milione poligona i drugih površi. Kako se scene opisuju korišćenjem geometrijskih objekata, geometrijski algoritmi imaju veoma važnu ulogu u računarskoj grafici. Jedan primer važnih problema je određivanje površi vidljivih (ili nevidljivih) iz neke tačke, a drugi je problem određivanja preseka zadatih objekata.

2.1.2 Geografski informacioni sistemi

Geografski informacioni sistemi, skraćeno GIS, čuvaju mape i dodatne podatke kao što su podaci o vegetaciji, padavinama, stanovništvu, infrastrukturi, itd. Oni se koriste za dobijanje informacija o kombinacijama različitih tipova podataka. Na primer, može se proveriti da li postoje gasne instalacije na terenu gde se planira iskopavanje, koje kuće se nalaze u šumi, gde se nalazi najbliža picerija, itd.

Važan problem u navigaciji vozila je efikasno lociranje vozila na mapi a i prikazivanje odgovarajućeg dela mape na računaru.

Informacija o visini je na mapama obično dostupna samo za neke tačke i za ostale tačke visina se procenjuje interpolacijom na osnovu podataka za poznate tačke iz okoline. Važan problem je kako se biraju te tačke iz okoline.

2.1.3 Robotika

U robotici je važan problem pronalaženja puta u okruženju sa preprekama. Jedna primena ovog problema je, na primer, u kretanju robota po površini Marsa, a druga kod automatskih usisivača (koji treba da na najefikasniji način usisaju celu prostoriju). Druga grupa geometrijskih algoritama koji se koriste u robotici odnosi se na kontrolu robotskih komponenti, na primer - robotske ruke, prilikom različitih operacija.

2.1.4 CAD/CAM

Računarski podržan dizajn (Computer aided design, CAD) i računarski podržana proizvodnja (Computer aided manufacturing, CAM) primenjuju se

u pravljenju štampanih ploča, ali i u pravljenju nameštaja ili komponenti u građevinarstvu. U ovoj oblasti geometrijski problemi su sveprisutni, u zadacima određivanja preseka i unije objekata, dekomponovanju objekata, prikupljanju komponenti, itd

2.2 Osnovni pojmovi i konvencije

2.2.1 Osnovne konvencije

- U opisu ravanskih algoritama, podrazumevaće se da su tačkama pridružene koordinate (tj. podrazumevaće se da se radi sa Dekartovom ravni).
- Tačke se označavaju velikim latiničnim slovima (kao što je uobičajeno u geometriji), iako je u preporučenom udžbeniku drugačije (tačke se označavaju malim latiničnim slovima).
- Iako to nije baš terminološki savršeno, za tačku A (Dekartove ravni) ćemo govoriti da je *levo*/*desno* od tačke B ako ima manju/veću x koordinatu. Za tačku A ćemo govoriti da je *dole*/*gore* od tačke B ako ima manju/veću z koordinatu. Analogno za prostor.
- Za pravu (u Dekartovoj ravni) za koju je vrednost y konstantna reći ćemo da je *horizontalna*, a za pravu za koju je vrednost x konstantna reći ćemo da je *vertikalna*.
- Za pozitivnu orijentaciju (Dekartove) ravni reći ćemo da je suprotna kazaljci na satu (ccw — counter-clockwise), za negativno reći ćemo da je orijentacija kazaljke na satu (cw — clockwise).

2.2.2 Primitivne operacije

- provera da li tačka pripada pravoj/krugu,
- provera da li se prave-duži seku i slično,
- ...

2.2.3 Osnovne formule

- Jednačina prave u ravni je

$$ax + by + c = 0$$

- Jednačina ravni je

$$Ax + By + Cz + D = 0$$

  Normala ravni je $[A, B, C]$.

- Ako tri nekolinearne tačke P_1, P_2 i P_3 pripadaju ravni, onda je vektor $P_1P_2 \times P_1P_3$ kolinearan vektoru normale ravni $[A, B, C]$. Ako su tačke P_1, P_2, P_3 kolinearne, onda ovim tačkama nije određena ravan i vektorski proizvod $P_1P_2 \times P_1P_3$ jednak je 0

- Rastojanje tačke (x, y, z) od ravni određeno je jednakošću

$$d = \frac{Ax + By + Cz + D}{\sqrt{A^2 + B^2 + C^2}}$$

  Za sve tačke sa jedne strane ravni, vrednosti $Ax+By+Cz+D$ imaju isti znak (dakle, da bi se odredilo sa koje strane ravni je neka tačka dovoljno je izračunati vrednost $Ax + By + Cz + D$.

- Orijentacija trojke tačaka:

  Neka je

$$ccw(A, B, C) = (x_b - x_a)(y_c - y_a) - (x_c - x_a)(y_b - y_a)$$

  Ako je $ccw(A, B, C) > 0$, onda je trojka $A-B-C$ pozitivno orijentisana.
  Ako je $ccw(A, B, C) < 0$, onda je trojka $A-B-C$ negativno orijentisana.
  Ako je $ccw(A, B, C) = 0$, onda su tačke A, B i C kolinearne.

Ugao ϕ između dva vektora v i u može se izračunati na osnovu sledeće veze: $v \cdot u = |v||u|cos\phi$.

2.2.4 2D transformacije

Translacija:

$$x' = x + t_x \quad y' = y + t_y$$

u matričnoj formi:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix}$$

tj.

$$P' = P + T$$

Skaliranje:

$$x' = s_x \cdot x \quad y' = s_y \cdot y$$

(nije ne nužno uniformno, tj. nije nužno $s_x = s_y$) u matričnoj formi:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix}$$

tj.

$$P' = S \cdot P$$

Rotacija (rotacija oko koordinatnog početka; uglovi su pozitivno orijentisani):

$$x' = x \cdot \cos\varphi - y \cdot \sin\varphi$$

$$y' = x \cdot \sin\varphi + y \cdot \cos\varphi$$

u matričnoj formi:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\varphi & -\sin\varphi \\ \sin\varphi & \cos\varphi \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix}$$

tj.

$$P' = R \cdot P$$

2.3 Primeri problema računarske geometrije

2.3.1 Opšti pristup rešavanju problema

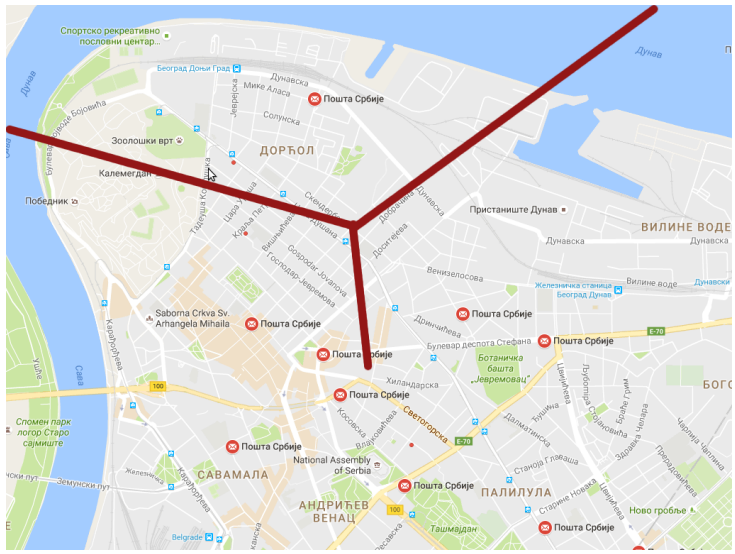
Rešavanje problema računarske geometrije obično ide kroz ove faze:

- Razumevanje problema i relevantnih geometrijskih svojstava;
- Ignorišu se specijalni i degenerisani slučajevi;
- Razmatra se pristup grubom silom i njegova složenost kao polazna tačka koju treba popraviti;
- Kreira se algoritam;
- Proverava se njegova ispravnost i analizira vremenska i prostorna složenost;
- Algoritam se proverava i, ako treba, modifikuje tako da ispravno radi i u specijalnim/degenerisanim slučajevima bez povećavanja složenosti.

2.3.2 Neki od problema sa brojnim primenama

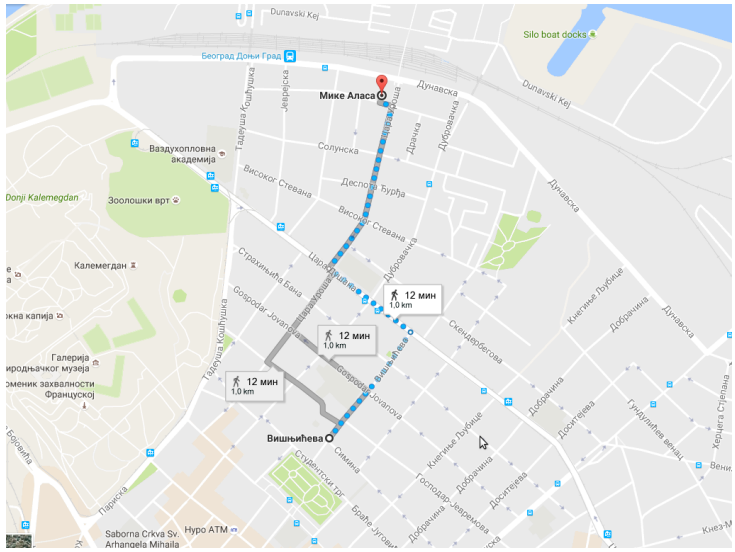
Problem 2.1. *Odrediti konveksni omotač za skup tačaka (u ravni ili prostoru).*

Problem 2.2. *Za zadatu tačku odrediti najbližu tačku iz datog skupa. (Na primer, odrediti najbližu poštu za zadatu tačku.)*

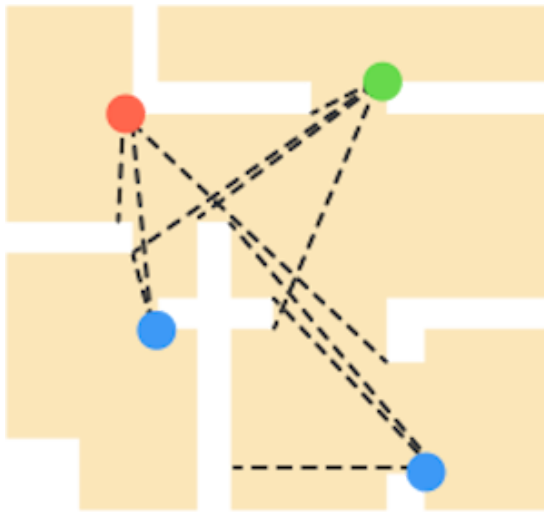


Problem 2.3. *Za zadatu tačku odrediti kom poligonu pripada. (Na primer, u pregledačima veba.)*

Problem 2.4. *Odrediti najkraći put od jedne do druge tačke u gradu.*



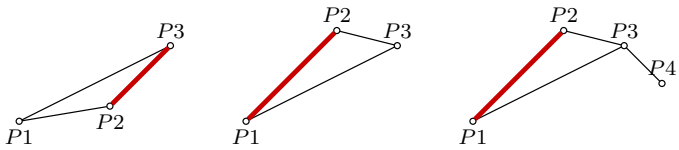
Problem 2.5. *Rasporediti kamere u stanu tako da je svaki njegov deo pod nadzorom.*



2.3.3 Određivanje najvećeg nagiba

Problem 2.6. *Dato je n tačaka u ravni; odrediti među njima dve takve da duž koja ih povezuje ima najveći nagib. Složenost algoritma treba da bude $O(n \log n)$*

Rešenje: Najpre se tačke sortiraju prema x koordinatama rastuće a onda se među nagibima duži koje povezuju *uzastopne* tačke bira najveći i on odgovara traženom paru.

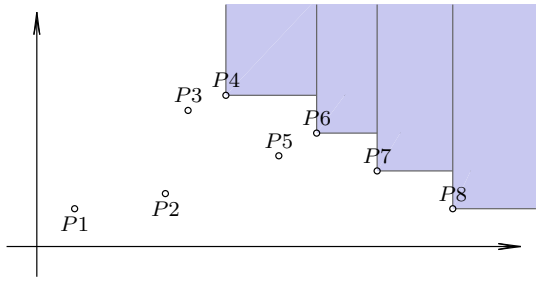


Korektnost se zasniva na tvrđenju: ako između tačaka A_0 i A_k u sortiranom redosledu postoje još neke tačke A_1, A_2, …, A_{k-1} onda postoje takve dve uzastopne tačke A_i i A_{i+1} takve da je nagib A_iA_{i+1} veći ili jednak nagibu A_0A_k (dokaz indukcijom).

Složenost rešenja je zaista $O(n \log n)$. □

2.3.4 Određivanje maksimalnih tačaka

Problem 2.7. *Za tačku P ravni kažemo da* dominira *tačkom Q ako su i x i y koordinate tačke P veće ili jednake od x i y koordinata tačke Q. Tačka P je* maksimalna *u datom skupu tačaka S ako nijedna od tačaka tog skupa ne dominira tačkom P. Opisati algoritam reda $O(n \log n)$ za određivanje svih maksimalnih tačaka datog skupa S od n tačaka.*



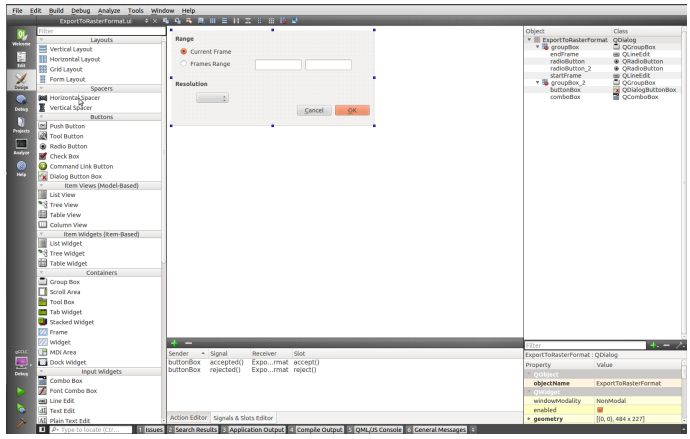
Rešenje: Najpre sortiramo u opadajućem poretku sve tačke po vrednostima x koordinata. Ako ima više tačaka sa istom x koordinatom, onda uzimamo onu sa najvećom y koordinatom, a ostale tačke zanemarujemo. Tačka P_1 je sigurno maksimalna. Neka je Y vrednost njene y koordinate. Tačka P_2 je maksimalna ako je vrednost y koordinate veća od Y; u tom slučaju ažuriramo Y (dobija vrednost y koordinate tačke P_2) i dodajemo P_2 u skup maksimalnih tačaka. Inače, P_2 ne ulazi u skup maksimalnih tačaka, a postupak nastavljamo dalje analogno. □

Glava 3

Alati Qt Creator i GCLC

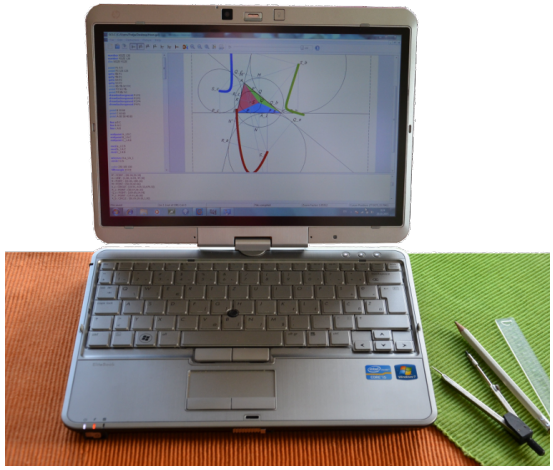
3.1 Qt Creator

Qt Creator je višeplatformsko integrisano razvojno okruženje za C++, JavaScript i QML (koje je deo razvojnog okruženja Qt GUI Application development framework). Uključuje debager i alatke za vizuelno programiranje. Qt Creator koristi različite C++ kompilatore, na primer, pod Linuxom koristi GNU kompilatore, a pod sistemom Windows MSVC ili MinGW.



U okviru kursa Geometrijski algoritmi, Qt biblioteka i Qt Creator koristiće se za implementacije geometrijskih algoritama (koje uključuju vizuelizacije).

3.2 GCLC



GCLC (od "Geometry Constructions -> LaTeX Converter") je alat za vizuelizaciju i pravljenje ilustracija. Glavne svrhe su mu:

- pravljenje matematičkih ilustracija visokog kvaliteta
- upotreba u nastavi matematike (posebno geometrije)
- kao alatka u geometrijskom rezonovanju

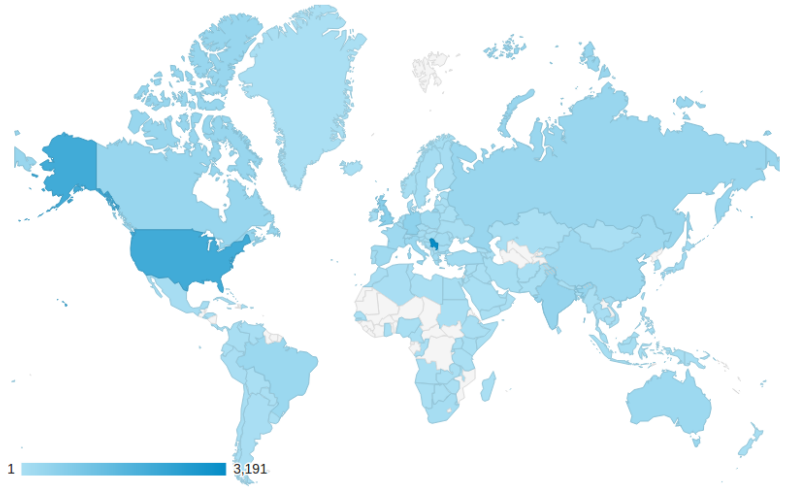
Osnovna ideja koji stoji iza sistema GCLC je da geometrijske konstrukcije nisu slike, nego formalne procedure. Zbog toga, u GCLC-u, pravljenje matematičkih ilustracija zasnovano je na njihovom opisivanju, a ne crtanju (slično kao što se u LaTeX-u tekst *opisuje*).

U opisivanju ilustracija može se koristiti veliki broj osnovnih i složenih konstrukcija i izometrijskih transformacija, krive, površi, simbolički izrazi, moduli za crtanje stabla i grafova, kontrola toka, korisnički definisane funkcije, itd.

I za Linux i za Windows postoje verzije za komandnu liniju i sa grafičkim korisničkim interfejsom. Verzije sa grafičkim korisničkim interfejsom pružaju niz dodatnih funkcionalnosti: podršku za interaktivan rad, animacije, praćenje tačaka, „watch window“ itd.

Kreirane slike mogu biti eksportovane u različite formate — jednostavan LaTeXformat, PSTricks format, TikZ format, EPS (Encapsualted PostScript), SVG (Scalable Vector Graphics), bitmap, PNG, JPG format. Podržan je import iz JavaView JVX formata.

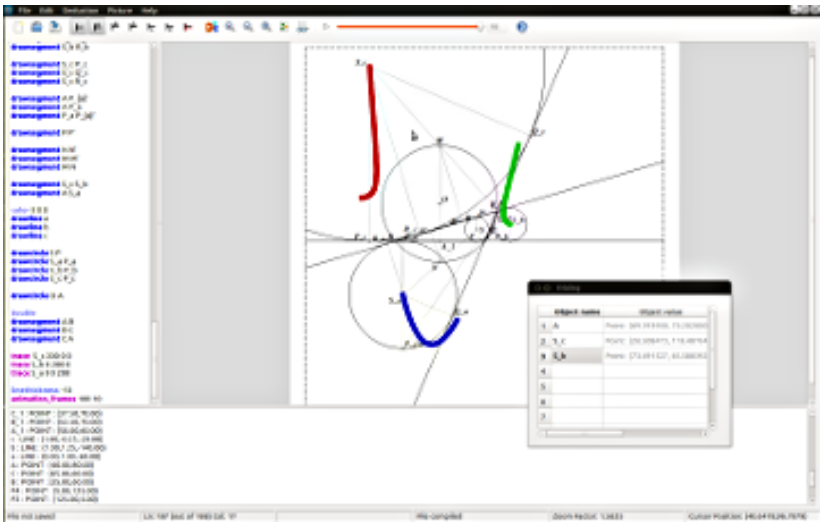
GCLC ima ugrađena i tri automatska dokazivača (jedan zasnovan na metodi površina, jedan zasnovan na Vuovoj metodi, jedan zasnovan na Grebnerovim bazama) koji mogu da dokažu veliki broj kompleksnih geometrijskih teorema.



Mapa korisnika GCLC-a

Uloga alata GCLC u kursu Geometrijski algoritmi može da bude sledeća:

- da služi kao alat za vizuelizaciju i pravljenje ilustracija rada geometrijskih algoritama;
- da služi kao ilustracija geometrijskog softvera;
- novi moduli za GCLC mogu da čine studentski projekat.



Alat GCLC je slobodno dostupan.

Glava 4

Konveksni omotač

Skup tačaka X (ravni ili prostora) je *konveksan* ako za svake dve tačke A i B skupa X svaka tačka duži AB pripada skupu X. Za poligon (tj. za skup tačaka poligona) važi: poligon je konveksan ako su za svaku njegovu stranicu sve njegove tačke sa iste njene strane. Važi i sledeće: poligon je konveksan, ako su mu svi unutrašnji uglovi manji od opruženog.

Konveksni omotač skupa tačaka je najmanji konveksan skup tačaka koji sadrži X. Za konačan skup tačaka u ravni, konveksni omotač je (konveksni) poligon. Za fiksiran skup tačaka, konveksni omotač je određen jednoznačno.

Konveksni omotač ima primene u okviru mnogih algoritama računarske geometrije, ali u mnogim drugim oblastima, uključujući prepoznavanje oblika, obradi slika, statistici, geografskim informacionim sistemima. Na primer, ako je potrebno pronaći put od tačke A do tačke B takav da ne seče poligon Φ, taj najkraći put je ili duž AB (ako ona ne seče Φ) ili ide duž konveksnog omotača za Φ. Konveksni omotači se koriste i za brzu proveru sudaranja dva objekta — ako se konveksni njihovi omotači ne seku, onda se sigurno ne seku ni sami objekti (a ako se konveksni omotači seku, onda treba sprovesti dodatnu proveru da li se sâmi objeksti seku). Kako se detektovanja sudara efikasno sprovodi za konveksne poligone, u opštem slučaju nekonveksni poligoni mogu biti razloženi na skupove konveksnih poligona čime se problem svodi na jednostavniji. Zone staništa životinja ili biljaka, epidemija, zračenja i slično, ponekad se opisuju konveksnim omotačom skupa pojedinačnih tačaka (gde je pojava bila primećena).

U nastavku će biti razmatran problem određivanja konveksnog omotača skupa tačaka ravni. Naivnim pristupom konveksni omotač konačnog skupa tačaka u ravni može se izračunati u vremenu $O(n^3)$. Bolji algoritmi imaju složenost $O(n \log n)$ ili $O(nh)$, gde je h broj temena koji učestvuju u konveksnom omotaču (dakle, u nekim slučajevima dostiže vreme $\Theta(n^2)$), a najbolji poznati (npr. Čanov algoritam iz 1996) imaju složenost $O(n \log h)$. Neki algoritmi za određivanje konveksnog omotača skupa tačaka u ravni koriste algoritme za određivanje prostog (nesamopresecajućeg) poligona čija su temena date

tačke.

4.1 Određivanje prostog mnogougla

Problem 4.1. *Dato je n tačaka u ravni, takvih da nisu sve kolinearne. Povezati ih zatvorenom prostom poligonskom linijom.*

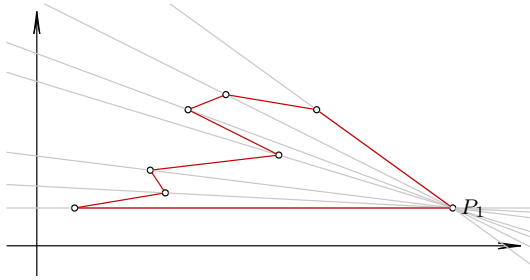
Algoritam: ProstMnogougao

Ulaz: Skup tačaka u ravni $\mathcal{P} = \{P_1, P_2, \ldots, P_n\}$

Izlaz: Niz tačaka koje određuju prost mnogougao.

1: Promeniti oznake tako da P_1 bude tačka sa najvećom x koordinatom, a ako ima više takvih, onda ona od njih koja ima najmanju y koordinatu
2: **za** $i := 2$ to n
3: izračunati ugao α_i između prave P_1P_i i x ose
4: Sortirati tačke prema uglovima α_i (ako za neki ugao ima više tačaka, onda ih sortirati prema rastojanju od P_1);

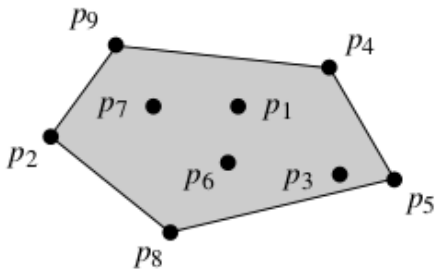
Složenost: $O(n \log n)$



4.2 Određivanje konveksnog omotača

Problem 4.2. *Za datih n tačaka u ravni odrediti konveksni omotač, tj. za dati skup tačaka ravni $\mathcal{P} = \{P_1, P_2, \ldots, P_n\}$, izračunati listu koja sadrži tačke iz $\mathcal{P}$ koje čine konveksni omotač, poređane u pozitivnom smeru (tj. u smeru suprotnom kretanju kazaljke na satu).*

Na primer, za skup tačaka: P_1, P_2, P_3, P_4, P_5, P_6, P_7, P_8, P_9 reprezentacija konveksnog omotača može da bude P_4, P_9, P_2, P_8, P_5.



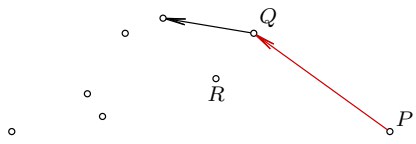
4.2.1 Naivni algoritam

Algoritam: KonveksniOmotac

Ulaz: Skup tačaka u ravni $\mathcal{P} = \{P_1, P_2, \ldots, P_n\}$

Izlaz: Lista L koja sadrži temena konveksnog omotača u smeru kazaljke na satu

1: $E := 0$
2: **za sve** parove različitih tačaka P i Q iz skupa $\mathcal{P}$
3: $\quad valid := true$
4: $\quad$ **za sve** tačke R iz $\mathcal{P}$ koje su različite od P i Q
5: $\quad\quad$ **ako je** trojka PQR negativno orijentisana **onda**
6: $\quad\quad\quad valid := false;$
7: $\quad$ **ako je** $valid$ **onda**
8: $\quad\quad$ dodaj usmerenu duž PQ u E
9: Od skupa stranica E konstruiši listu temena L tako da kreiraju poligon u negativnoj orijentaciji



Svojstva algoritma

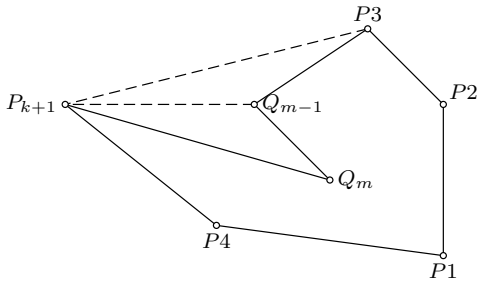
Ukupna vremenska složenost: $O(n^3)$ (ima $n(n-1)$ parova tačaka i za svaki treba ispitati $n-2$ tačke; taj dominira složenosti korak 9).

Degenerisani slučajevi: test "da li je trojka PQR negativno orijentisana" ispravno pokriva slučaj da su tačke P, Q i R kolinearne.

Robustnost: iako je algoritam korektan, zbog zaokruživanja u aritmetici sa pokrenim zarezom, možda njegova implementacija ne daje uvek ispravan rezultat (ako su neke tri tačke *skoro* kolinearne). Dakle, algoritam nije robustan.

4.2.2 Gremov algoritam

Osnovna ideja algoritma:[1] ako je dato n tačaka u ravni, uređenih na osnovu algoritma ProstMnogougao, onda umemo da konstruišemo podskup skupa prvih k tačaka takav da odgovarajući konveksni omotač pokriva prvih k tačaka. Kada se doda jedna tačka, proverava se ugao koji zahvataju dve poslednje ivice; ako nova tačka pripada unutrašnjosti tekućeg poligona (tj. ako je ugao $Q_{m-1}Q_mP_{k+1}$ na slici manji od opružnog ugla, tj. ako je trojka tačaka (Q_{m-1}, Q_m, P_{k+1}) pozitivno orijentisana), ona se dodaje nizu; inače se dodaje nizu ali se mora proveriti da li treba izbaciti neku od prethodnih tačaka.



Svojstva algoritma

Ispravnost: Indukcijom, koristeći navedenu invarijantu (tokom primene algoritma, svi unutrašnji uglovi tekućeg poligona su manji od opruženog). Petlja nikad ne dostiže vrednost $m = 1$ zbog načina na koji je izabrana tačka P_1. Na kraju primene algoritma, trojka $Q_{m-1}Q_mP_1$ sigurno nije negativno orijentisana (zbog poretka u kojem se obrađuju tačke P_i).

Složenost: $O(n \log n)$ (vraćanjem unazad svaka tačka može biti obrisana iz niza najviše jednom).

[1] Ovaj algoritam za određivanje konveksnog omotača objavio je Ronald Grem (Graham) 1972. godine.

Algoritam: Gremov algoritam

Ulaz: Skup tačaka u ravni $\mathcal{P} = \{P_1, P_2, \ldots, P_n\}$

Izlaz: Lista tačaka Q_1, Q_2, ... koje određuju konveksni omotač (pozitivno orijentisan)

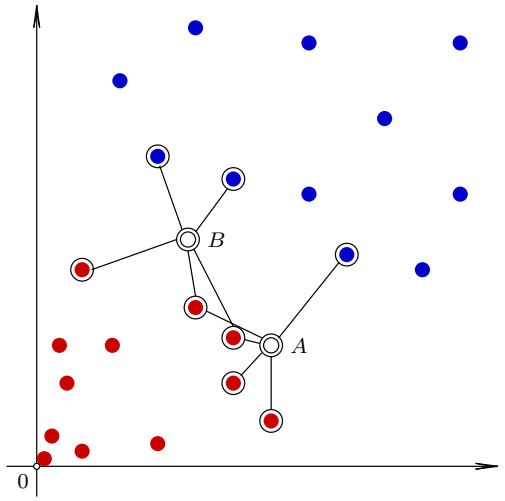
1: Promeniti oznake tako da P_1 bude tačka sa najvećom x koordinatom, a ako ima više takvih, onda ona od njih koja ima najmanju y koordinatu
2: Koristeći algoritam ProstMnogougao urediti tačke u odnosu na P_1 (neka je dobijen niz tačaka P_1, P_2, ... P_n)
3: $Q_1 := P_1$
4: $Q_2 := P_2$
5: $Q_3 := P_3$
6: m := 3
7: **za** vrednosti od $k = 4$ do n
8: **dok god** trojka tačaka (Q_{m-1}, Q_m, P_k) negativno orijentisana
9: m := m-1
10: m:=m+1
11: $Q_m := P_k$

Glava 5

Određivanje najbližih N tačaka

Problem 5.1. *Za datu tačku X odrediti skup najbližih N tačaka iz zadatog skupa S.*

Navedeni problem važan je, na primer, za efikasnu primenu metoda N najbližih suseda iz mašinskog učenja.



Dobro rešenje zavisi od toga da li će se problem rešavati jednom ili više puta za isti skup tačaka S.

Ako se najpre izračunaju rastojanja od X do svih tačaka skupa S, problem se svodi na određivanje najmanjih k elemenata datog skupa (ali se to onda radi za svaku zadatu tačku iznova).

Veoma naivno rešenje je da se rastojanja sortiraju i da se iz sortiranog niza izabere najmanjih N elemenata. Ovo rešenje ima složenost $O(n \log n)$, gde je n

broj elemenata skupa S. Drugo naivno rešenje je da se nađe najmanji element skupa S, pa sledeći najmanji, i tako dalje, sve do N-tog najmanjeg. Ovo rešenje ima složenost $O(Nn)$.

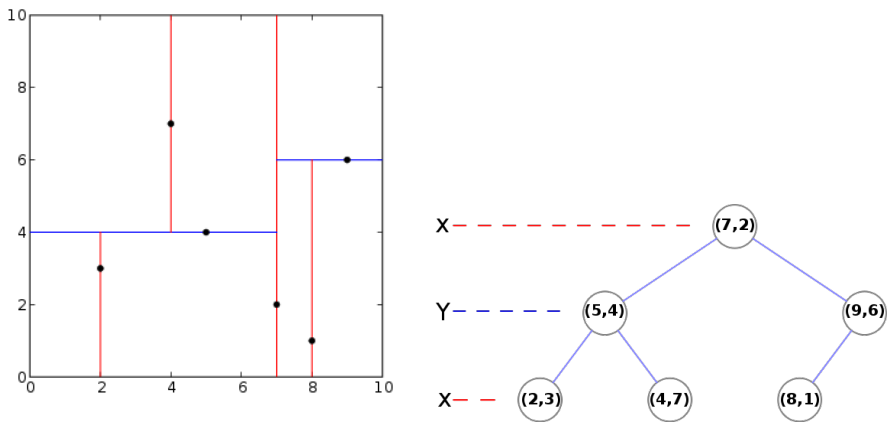
Moguće rešenje je da se koristi hip fiksne veličine N, da se prođe kroz skup rastojanja, proveri da li je tekuće rastojanje manje od najmanjeg u hipu i, ako jeste, da ga zameni. Ovo rešenje ima složenost $O(n \log N)$.

Problem može biti rešen i primenom algoritma `quickselect`, čiji je autor, kao i algoritma `quicksort`, Toni Hor. Algoritam `quickselect` služi za određivanje N-tog najmanjeg elementa niza ili N najmanjih elemanata. Sličan je algoritmu `quicksort`, pronalazi se pivot, niz deli (u linearnom vremenu po broju elemenata) na elemente manje ili jednake pivotu i elemente veće od pivota. Razlika je u rekurzivnom koraku: umesto da se algoritam rekurzivno pozove za oba podniza (kao u algoritmu `quicksort`), algoritam se poziva za **tačno jedan** podniz: levi, ako je indeks pivota veći od N, i desni – inače. Slično kao u algoritmu `quicksort`, složenost je u najgorem slučaju $O(n^2)$, a u najboljem je opisana vezom $T(n) = T(n/2) + cn$, odakle je $T(n) = O(n)$. U prosečnom slučaju i mnogim primenama, algoritam se ponaša linearno po n.

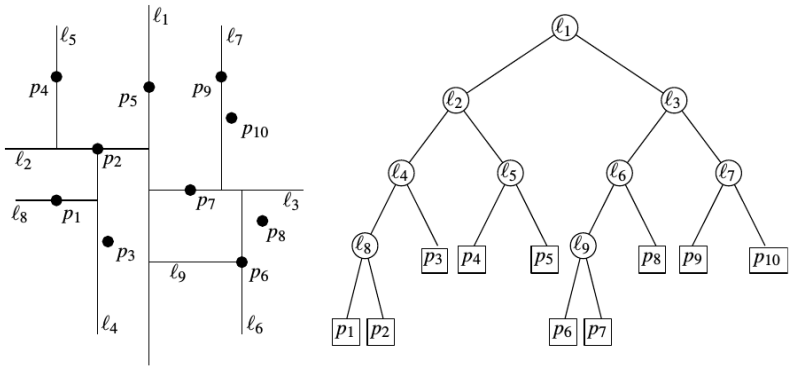
5.1 kd-stabla

kd-stablo opisuje skup tačaka u k-dimenzionom prostoru. To je struktura pogodna za rešavanje prostornih problema kao što je pretraga oblasti i nalaženje najbližih tačaka. Koristi se kada je za dati skup S potrebno pronaći N najbližih za više zadatih tačaka (a ne samo jednu). U tim situacijama najpre se tačke skupa S pohranjuju u kd-stablo koje se koristi za svaki upit.

U čvorovima stabla su tačke skupa S. Ako se radi o tačkama u prostoru, na nivoima stabla su ključne redom vrednosti x koordinate tačke u čvoru, pa y koordinate, pa z koordinate (i tako redom ponovo). Na nivou x, levo su tačke sa manjim ili jednakim x koordinatama, desno sa većim x koordinatama. Analogno za nivoe y i z. Za svaki čvor, broju tačaka u njegovom levom i desnom podstablu razlikuje se najviše za 1. Ilustracija za $k = 2$:



U unutrašnjim čvorovima ne moraju da se čuvaju tačke, tačke mogu da se čuvaju samo u listovima:



kd-stablo može da se konstruiše u vremenu $O(n \log n)$, gde je n broj tačaka u skupu S. Ideja je da se naprave tri niza koja sadrže sve tačke, a zatim da se oni sortiraju – prvi po x, drugi po y, treći po z (zapravo – u nizovima mogu da se čuvaju samo indeksi tačaka u polaznom nizu). Kada se pravi prvo razdvajanje po x koordinati – lako se nalazi središnji element po x. U odnosu na tu tačku se onda ažuriraju i preostala dva niza (u linearnom vremenu). Postupak se onda ponavlja za naredne nivoe. Ilustracija za $k = 3$:

	Tuples (x,y,z)	Initial Indices xyz	yzx	zxy	After First Split xyz	yzx	zxy	After Second Split xyz	yzx	zxy
0	(2,3,3)	11	13	9	11	13	9	13	13	9
1	(5,4,2)	13	4	6	13	9	1	0	9	13
2	(9,6,7)	0	5	1	0	0	13	9	0	0
3	(4,7,9)	10	9	7	10	1	0			
4	(8,1,5)	3	0	13	3	10	10	11	10	10
5	(7,2,6)	1	6	0	1	11	11	10	11	11
6	(9,4,1)	9	1	12	9	3	3	3	3	3
7	(8,4,2)	5	7	10						
8	(9,7,8)	4	10	4	4	4	6	4	4	6
9	(6,3,1)	7	12	5	7	6	7	7	6	7
10	(3,4,5)	14	2	14	14	7	12	6	7	4
11	(1,6,8)	6	11	2	6	12	4			
12	(9,5,3)	12	14	11	12	2	14	14	2	14
13	(2,1,3)	2	8	8	2	14	2	2	14	2
14	(8,7,6)	8	3	3	8	8	8	8	8	8

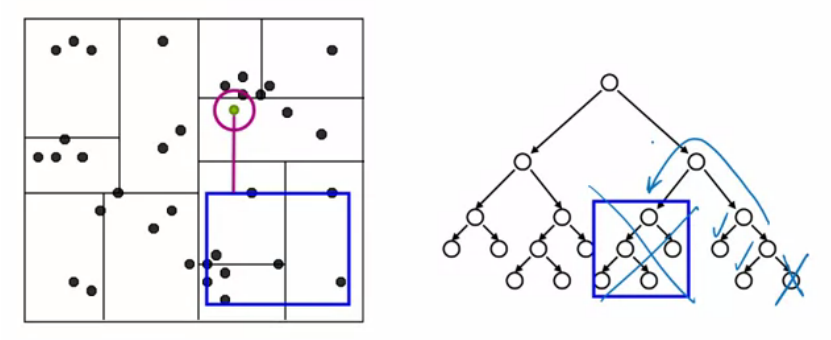
Za vremensku složenost opisanog postupka konstruisanja kd-stabla važi:

$$T(n) = 2T(n/2) + O(n),$$

odakle je $T(n) = O(n \log n)$.

Traženje najbliže tačke sprovodi se na sledeći način: najbliža tačka traži se obilaskom u dubinu, najpre obilaskom grane koja sadrži zadatu tačku. Kada se dođe do lista, dobija se tekuća najbliža tačka i onda se bektrekuje. Ključno je što može da se odbaci celo podstablo ako je rastojanje od zadate tačke do odgovaraće oblasti veće nego do trenutno najbliže tačke.

Ilustracija za ravanski slučaj, tj. $k = 2$ (tada se ne kaže da se radi o 2d-stablu, nego o dvodimenzionom kd-stablu):



Složenost za pronalaženje jedne najbliže tačke je $O(\log n)$ ako su tačke slučajno raspoređene (analiza u opštem slučaju je veoma komplikovana).

Algoritam se lako modifikuje da traži N najbližih tačaka – potrebno je održavati listu N najbližih do tada i nova grana se može eliminisati ako ona ne može da sadrži nijednu tačku bližu nego trenutno najbližih N.

kd-stabla se koriste i za određivanje svih tačaka koje pripadaju nekom n-dimenzionionom intervalu:

Problem 5.2. *Za dati skup tačaka S odrediti podskup tačaka koje pripadaju datom n-dimenzionionom intervalu.*

Glava 6

Postojanje preseka duži na pravoj

Problem 6.1. *Za dati skup duži na jednoj pravoj, zadatih levim i desnim temenima, proveriti da li se neke dve seku.*

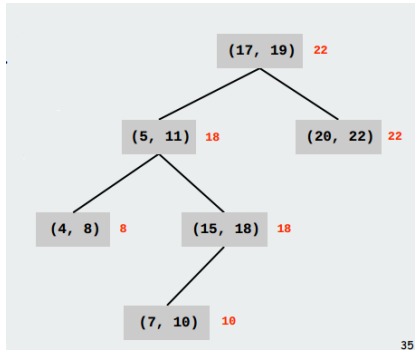
Navedeni problem može se rešiti korišćenjem strukture koja podžava sledeće operacije:

- dodaj duž (L, R).
- obriši duž (L, R).
- pretraga za dužima koje seku duž (L, R).

(Pretpostavlja se, za sada, da ne postoje dve duži sa istim koordinatama.)

Ta struktura može biti implementirana kao binarno stablo pretrage koje zovemo *stablo pretrage duži* (eng. interval search tree):

- u svakom čvoru čuva se jedna duž;
- čvorovi su sortirani po levim krajevima duži;
- kao dodatna informacija čuva se maksimalno desno teme u podstablu.



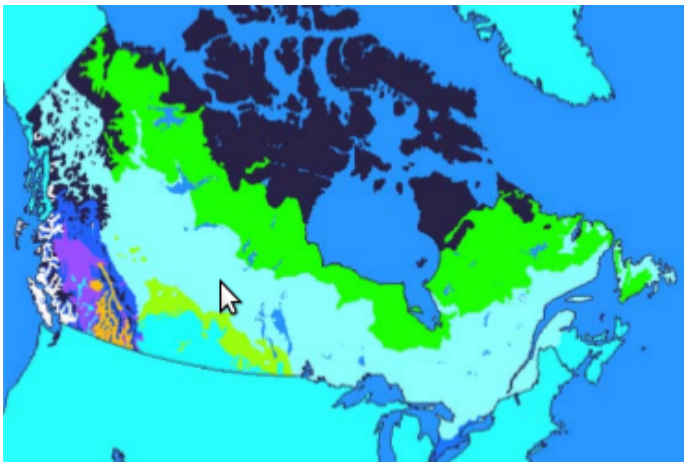
Kada se proverava da li postoji presek date duži (l, d) i skupa duži, proverava se da li duž seče duž u korenu stabla. Ako ne, proverava se da li je l veće od maksimuma koji se čuva u levom podstablu, ako jeste - ide se na desno stablo, ako nije - ide se na levo podstablo. Ovaj pristup je ispravan. Zaista, ako je l veće od maksimuma u levom podstablu, onda sigurno duž (l, d) ne seče nijednu duž iz levog podstabla. Suprotno, ako je l manje ili jednako od maksimuma u levom podstablu, dovoljno je proveriti levo podstablo. Naime, ako u levom podstablu nema preseka, onda ih nema ni u desnom podstablu. To je tačno zbog sledećeg. Pretpostavimo da u levom podstablu nema preseka i da postoji duž (a, max) gde je max maksimalna vrednost desnog temena u levom podstablu. Ako duž (l, d) nema preseka u levom podstablu, onda mora da važi $d < a$. Međutim, kako su u stablu duži poređane po levom temenu, ovo znači da je d manje i od svih levih temena u desnom podstablu, pa ni u desnom podstablu ne može da ima preseka.

Ukoliko je stablo balansirano – onda se u vremenu $O(\log n)$ može proveriti da li postoji presek neke duži sa dužima koje se čuvaju u stablu.

Glava 7

Metod brišuće prave

Metod brišuće prave koristi se u mnogim geometrijskim algoritmima. U nastavku će biti opisano nekoliko primera, od kojih je jedan motivisan, između ostalog, geografskim informacionim sistemima. Geografski informacioni sistemi čuvaju raznovrsne informacije (na primer, o vegetaciji, nadmorskoj visini, padavinama, itd) i na mapama ih prikazuju u *slojevima*, tj. *lejerima* (eng. layers). Nekad se prikazuje samo jedan sloj, a nekad više slojeva. Nekad je potrebno preklopiti informacije iz nekoliko slojeva i odrediti preseke ili unije regiona različitih tipova – na primer, odrediti regione male nadmorske gde dominiraju četinarske šume, ili – odrediti regione gde postoje jedna ili druga biljka koje izazivaju alergije). Ako su regioni reprezentovani prostim poligonima, potrebno je efikasno odrediti presek poligona. U tome je važna operacija efikasnog određivanja preseka duži.



7.1 Određivanje preseka duži

Problem 7.1. *Za dati skup S n zatvorenih duži u ravni, odrediti sve presečne tačke između duži iz S.*

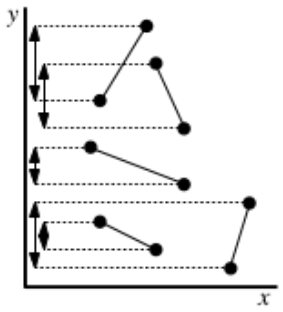
Rešenje navedenog problema može se upotrebiti i za određivanje međusobnih preseka stranica dva prosta poligona P i Q. Naime, mogu se odrediti svi preseci duži iz skupa svih stranica, a onda se lako mogu odbaciti preseci po dve stranice iz istog poligona (jer su to nužno temena stranica).

7.1.1 Rešenje grubom silom i nedostaci

Jednostavan pristup grubom silom, razmatra sve parove duži iz S i proverava da li se one seku. Ovaj algoritam ima složenost $O(n^2)$ i u određenom smislu je optimalan. Naime, postoji mogućnost da se svake dve duži skupa seku, te traženi algoritam u tom slučaju mora da izvrši $\Theta(n^2)$ operacija. S druge strane, u mnogim primenama, broj preseka je veoma mali, daleko manji od broja parova duži i za takve situacije treba razviti efikasniji algoritam. Drugim rečima, potrebno je razviti algoritam čije vreme izvršavanje zavisi ne samo od broja duži u skupu S, nego i od broja presečnih tačaka. Za takav algoritam kažemo da je *izlazno-zavisan* — vreme izvršavanja zavisi od veličine izlaza.

7.1.2 Osnovni algoritam zasnovan na brišućoj pravoj

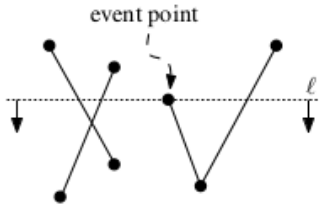
Ideja prvog naprednog algoritma je da se ne ispituju mogući preseci svih parova duži. Na primer, ako su dve duži „daleko“, one neće imati presek. Zato se mogu razmatrati projekcije duži na y-osu.



Ako projekcije nemaju zajedničkih tačaka, onda ni duži nemaju zajedničkih tačaka. Obratno naravno ne važi i provera projekcija služi samo za jednostavno odbacivanje nekih parova koji se ne seku. Ako se projekcije neke dve duži seku u nekoj tački sa y koordinatom jednakom b, onda horizontalna prava sa tom

istom y koordinatom b mora da seče obe duži. U algoritmu će ključnu ulogu imati horizontalna prava koja menja svoju y koordinatu i koju zovemo *brišuća prava* (eng. sweep line).

Ukoliko u nekom trenutku brišuća prava seče više duži, za njih treba da proverimo da li se neke dve međusobno seku.



Status brišuće prave je skup duži koje seče. Dok se brišuća prava pomera odozgo naniže status se menja, ali ne neprekidno, već samo u nekim posebnim tačkama — kada naiđe na teme neke duži. Ove trenutke (kada brišuća prava naiđe na teme neke duži) zvaćemo *događajima algoritma* (eng. events), a odgovarajuće teme *tačka događaja* (eng. event point). Između dva događaja ne može se promeniti tekući skup preseka, pa skup preseka treba ažurirati samo u trenucima događaja. Konkretno, ako je tačka događaja gornje teme duži, nova duž počinje da seče brišuću pravu i mora da se doda u status i da za svaku pravu iz statusa proveri da li je seče. Ako je tačka događaja donje teme duži, onda se ona izbacuje iz statusa. Na ovaj način proveravaju se preseci samo parova duži koje seče brišuća prava. No, može se desiti da brišuća prava u jednom trenutku seče sve duži (te se moraju proveravati svih parovi), a da se nikoje dve ne seku. Zbog toga ovaj algoritam ima složenost $O(n^3)$ i nije izlazno-zavisan te postoji prostor za dalja unapređenja.

7.1.3 Izlazno-zavisni algoritam zasnovan na brišućoj pravoj

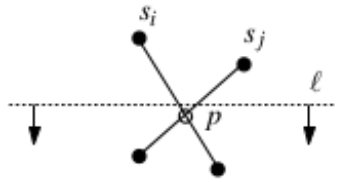
Umesto da se proverava da li postoji presek za svake dve duži koje seku brišuću pravu, dovoljno je proveravati samo neke, pažljivo odabrane parove. Da bi se smanjio broj parova duži za koje se proverava da li postoji presek, duži će da budu uređene prema tome kako seku brišuću pravu. Onda je dovoljno proveravati samo parove duži koje su *susedne* u tom horizontalnom uređenju. Preciznije, za novu duž (duž na čije se gornje teme nailazi brišućom pravom) proverava se mogući presek samo za dve duži — onu koje je neposredno levo i onu koja je neposredno desno od njenog gornjeg temena. Kasnije, tokom daljeg pomeranja brišuće prave, ta duž može postati susedna drugim dužima i biće provereno da li ih seče. Zbog ovoga, status više neće biti samo skup duži koje seku brišuću pravu, nego uređeni niz duži koje seku brišuću pravu. Tako definisan status ne menja se (kao u prvoj verziji) samo u temenima duži, nego i u presečnim tačkama duži. Kada se naiđe na presek, potrebno je obraditi

dve duži koje se seku i ažurirati status. Presečne tačke, zato, činiće nove tačke događaja.

Postavlja se pitanje da li će na opisani način biti pronađene sve presečne tačke. Drugim rečima, ako se dve duži seku, pitanje je da li uvek postoji pozicija brišuće prave u kojoj su te dve duži susedne. U prvom koraku zanemarimo specijalne slučajeve: pretpostavimo da nijedna duž nije horizontalna, da se nikoje dve duži ne poklapaju delom i da se nikoje tri duži ne seku u jednoj tački. Može se pokazati da se ovi specijalni slučajevi mogu lako obraditi. Presek koji je istovremeno gornje teme jedne duži lako se otkriva kada brišuća prava dođe do tog temena. Dakle, ključno pitanje je da li će biti otkriveni svi unutrašnji preseci duži.

Lema 7.1. *Pretpostavimo da su s_i i s_j dve duži koje nisu horizontalne i čije unutrašnjosti se seku u jednoj tački P, kao i da ne postoji nijedna više duž koja sadrži tačku P. Tada postoji tačka događaja iznad P, kada su duži s_i i s_j postale susedne i provereno je da li postoji njihov presek.*

Dokaz: Neka je l horizontalna prava iznad tačke P takva da ne postoji nijedna tačka događaja na njoj niti između nje i horizontalne prave kroz P.



Na ovoj pravoj, duži s_i i s_j su susedne. S druge strane, duži s_i i s_j nisu bile susedne na samom početku primene algoritma (kada je status morao biti prazan). Dakle, mora da postoji događaj kada su duži s_i i s_j postale susedne i kada provereno je da li postoji njihov presek. □

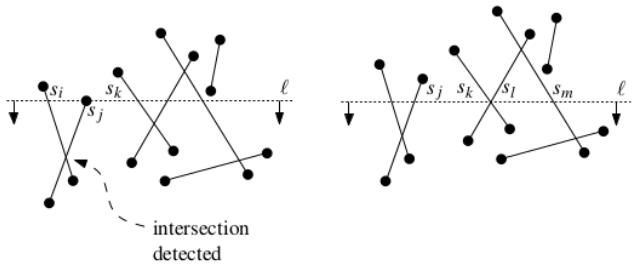
Osnovne ideje algoritma su:

- Horizontalna brišuća prava pomera se odozgo nadole.
- Brišuća prava zastaje na određenim tačkama događaja — temenima (koja su poznata na početku) i presečnim tačkama (koje se računaju u hodu). Tada je, u zavisnosti od vrste događaja, potrebno ažurirati status (uređenu listu duži koje seku brišuću pravu) i proveriti moguće preseke.

Obrada događaja

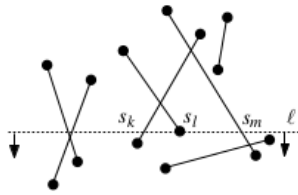
Obrada događaja:

Gornje teme duži: Nova duž seče brišuću pravu i mora biti provereno da li ona seče dve svoje susedne duži. Samo presečne tačke *ispod* brišuće prave su bitne, one iznad su već detektovane ranije. Ako su, na primer, duži s_i i s_k susedne na brišućoj pravoj i gornje teme duži s_j je između njih, onda je potrebno proveriti da li postoji presek s_j sa s_i, kao i s_j sa s_k. Ako je pronađen presek ispod brišuće prave, on će zapamćen kao tačka događaja (i biće obrađen u nekom trenutku kasnije).



Presečna tačka: U presečnoj tački, duži koje se seku treba da zamene mesta u uređenom nizu. Svaka od njih dobija najviše jednog novog suseda i sa njim mora da bude proveren potencijalni presek. Ponovo, samo presečne tačke *ispod* brišuće prave su bitne, one iznad su već detektovane ranije.

Pretpostavimo da na brišućoj pravoj postoje četiri duži s_j, s_k, s_l i s_m (u tom poretku) onda kada se naiđe na presek duži s_k i s_l. Tada duži s_k i s_l treba da razmene mesta u listi i potrebno je proveriti da li ispod brišuće prave postoji presek duži s_j i s_l, kao i da li postoji presek duži s_k i s_m. Eventualni preseci postaće tačke događaja (i biće obrađene kasnije). Moguće je da su te tačke preseka već otkrivene ranije (ako je par duži ranije bio susedan).



Donje teme duži: Kada se naiđe na donje teme duži, njena dva suseda će postati susedna i mora da se proveri da li imaju presek. Ako imaju presek

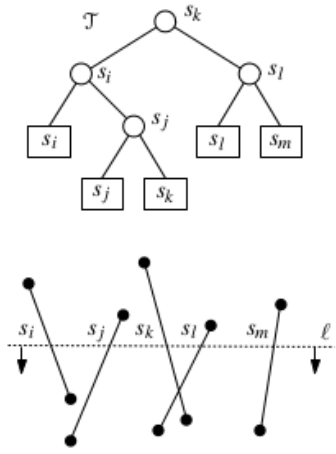
ispod brišuće prave, on će postati tačka događaja (koja je možda već ranije otkrivena). Pretpostavimo da se tri duži s_k, s_l i s_m nalaze u ovom poretku na brišućoj pravoj. Kada se naiđe na donje teme duži s_l, duži s_k i s_m će postati susedne i mora da se proveri da li imaju presek.

Strukture podataka

Red događaja. Događaji će biti smešteni u strukturu $\mathcal{Q}$ koju ćemo zvati red događaja. Potrebno je da struktura može efikasno da vrati sledeći događaj koji treba obraditi, tj. prvu tačku ispod brišuće prave. Ukoliko postoje dve tačke događaja sa istom y koordinatorm, biće vraćena ona sa manjom x koordinatom. Red događaja mora da omogući i efikasno umetanje (uključujući proveravanje da li data tačka događaja već postoji).

Potrebno je definisati uređenje $\prec$ nad tačkama događaja koje određuje poredak u kojem će one biti obrađene. Za tačke P i Q važi $P \prec Q$ ako je $P_y > Q_y$ ili važi $P_y = Q_y$ i $P_x < Q_x$. Tačke događaja biće čuvane u balansiranom binarnom uređenom stablu,[1] uređenom u skladu sa relacijom $\prec$. Sa svakom tačkom P u $\mathcal{Q}$ će biti čuvane i duže čije je to gornje teme. I operacija uzimanja sledećeg događaja i ubacivanje novog događaja imaju vremensku složenost $O(\log m)$, gde je m broj događaja u $\mathcal{Q}$.

Status. Status je uređen niz duži koje seku brišuću pravu. Biće reprezentovan dinamičkom strukturom $\mathcal{T}$ koja omogućava da se duži efikasno dodaju ili izbacuju tokom pomeranja brišuće prave. To može da bude balansirano binarno uređeno stablo sa uređenjem nad dužima određenim poretkom tačaka preseka sa brišućom pravom.



[1] Ne koristi se struktura hip, jer nije dovoljno da se efikasno nalazi najmanji element, već je potrebno da se efikasno proverava da li dati element već postoji.

Preciznije, duži koje seku brišuću pravu biće čuvane u listovima balansiranog binarnog stabla $\mathcal{T}$. Uređenje duži sleva-nadesno duž brišuće prave odgovara sleva-nadesno poretku listova. U unutrašnjim čvorovima stabla čuvaju se relevantne informacije kako bi se omogućila pretraga do listova. U svakom unutrašnjem čvoru čuva se duž koja najdesnija u levom podstablu.[2] Pretpostavimo da u stablu $\mathcal{T}$ tražimo duž koje je neposredno levo od tačke P koja pripada brišućoj pravoj. U svakom unutrašnjem čvoru v ispitujemo da li je P levo ili desno od duži u v. U zavisnosti od toga, nastavljamo pretragu levo ili desno od v, sve do nekog lista. Tražena duž je ili u tom listu ili u listu neposredno levo. Analogno se traži duž koja je desno od P ili sadrži P. Operacije pretrage i dodavanja u $\mathcal{T}$ imaju vremensku složenost $O(\log n)$.

Algoritam

Algoritam: OdrediPreseke

Ulaz: Skup S duži u ravni

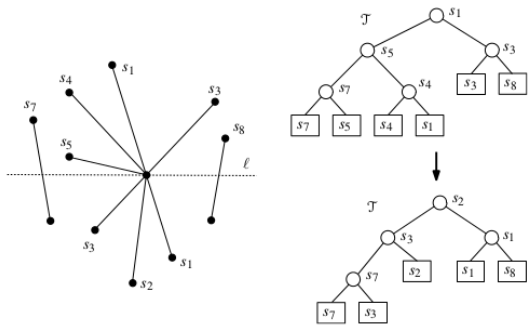
Izlaz: Skup presečnih tačaka duži iz S (sa informacijom o tome koje duži sadrže taj presek)

1: Inicijalizuj prazan red događaja $\mathcal{Q}$. Ubaci sva temena duži u $\mathcal{Q}$. Kada se dodaje gornje teme duži, sa njim se čuva i odgovarajuća duž.
2: Inicijalizuj praznu strukturu za status $\mathcal{T}$.
3: **dok god** red $\mathcal{Q}$ nije prazan
4: Uzmi sledeću tačku događaja P iz $\mathcal{Q}$ i obriši je.
5: ObradiDogađaj(P)

Slika 7.1: Algoritam OdrediPreseke

Opšti algoritam dat je na slici 7.1, a procedura za obradu događaja, koja pokriva i sve specijalne slučajeve, na slici 7.2. U koracima 8–16 se pretpostavlja da duži s_l i s_r postoje. Ako ne postoje, ti koraci očito ne treba da budu primenjeni.

[2] Alternativno, duži se mogu čuvati i u unutrašnjim čvorovima, što bi uštedelo nešto memorije. Izabrano rešenje je, ipak, pogodnije za jednostavniji opis algoritma.



Algoritam: ObradiDogađaj(P)

1: Neka je $U(P)$ skup duži čije gornje teme je P; ove duži se čuvaju zajedno sa tačkom događaja P. (Za horizontalne duži, za gornje teme se uzima levo teme.)

2: Odrediti sve duži iz $\mathcal{T}$ koje sadrže P; one su susedne u $\mathcal{T}$. Neka je $L(P)$ skup duži čije je donje teme P i neka je $C(P)$ skup pronađenih duži čija unutrašnjost sadrži P

3: **ako je** $L(P) \cup U(P) \cup C(P)$ sadrži više od jedne duži **onda**

4: Prijavi P kao presek zajedno sa $L(P)$, $U(P)$ i $C(P)$

5: Obriši sve duži iz skupa $L(P) \cup C(P)$ iz $\mathcal{T}$

6: Ubaci duži iz skupa $U(P) \cup C(P)$ u $\mathcal{T}$.
Poredak duži u $\mathcal{T}$ treba da odgovara poretku u kojem one seku brišuću pravu neposredno ispod tačke P. Ako postoji horizontalna duž, ona dolazi na kraj svih duži koje sadrže P.
{Brisanje i ponovno umetanje duži iz $C(p)$ obrće njihov poredak}

7: **ako je** $U(P) \cup C(P)$ prazan skup **onda**

8: Neka su s_l i s_r levi i desni sused tačke P u $\mathcal{T}$.

9: PronađiNoviDogađaj(s_l, s_r, P)

10: **inače**

11: Neka je s' najlevlja duž skupa $U(p) \cup C(p)$ u $\mathcal{T}$.

12: Neka je s_l levi sused duži s' u $\mathcal{T}$.

13: PronađiNoviDogađaj(s_l, s', P)

14: Neka je s'' najdesnija duž skupa $U(p) \cup C(p)$ u $\mathcal{T}$.

15: Neka je s_r desni sused duži s'' u $\mathcal{T}$.

16: PronađiNoviDogađaj(s'', s_r, P)

Slika 7.2: Algoritam OdrediPreseke

Algoritam: PronađiNoviDogađaj(s_l, s_r, P)

1: **ako je** s_l i s_r se seku ispod brišuće prave, ili na njoj, ali desno od tekuće tačke događaja P, i ako taj presek nije već prisutan kao događaj u $\mathcal{Q}$ **onda**
2: Ubaci presečnu tačku kao novi događaj u $\mathcal{Q}$

Slika 7.3: Algoritam PronađiNoviDogađaj

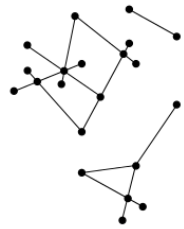
Procedura PronađiNoviDogađaj (prikazana na slici 7.3) je jednostavna: ona jednostavno proverava da li dve duži imaju presek. Treba jedino voditi računa o tome da li je taj presek već ranije pronađen. Kada su u pitanju horizontalne duži, konvencija je da se događaji sa istom y koordinatom obrađuju sleva nadesno. To znači da je potrebno voditi računa o tačaka preseka koje su desno od tekuće tačke događaja.

Svojstva

Teorema 7.1. *Algoritam OdrediPreseke je korektan, tj. ispravno određuje sve presečne tačke i duži koje ih sadrže. Njegovo vreme izvršavanja je* $O(n \log n + I \log n)$*, gde je* I *broj presečnih tačaka zadatih duži.*

Dokaz: Ispravnost: na osnovu leme 7.1.

Složenost: Inicijalizacija strukture $\mathcal{Q}$ je složenosti $O(n \log n)$, a inicijalizacija strukture $\mathcal{T}$ je složenosti $O(1)$. Prilikom obrade događaja, izvršavaju se operacije nad $\mathcal{T}$ od kojih svaka ima složenost $O(\log n)$. Takvih operacija ima $O(m(P))$, gde je $m(P)$ jednak broju elemenata skupa $L(P) \cup U(P) \cup C(P)$. Ako je m zbir svih vrednosti $m(P)$ za sve tačke događaja, onda je vreme izvršavanja algoritma $O(m \log n)$.



Dokažimo da je $m = O(n + I)$. Razmotrimo skup duži kao graf u ravni – čvorovi tog grafa su temena duži i presečne tačke. Tada je za čvor P

vrednost $m(P)$ ograničena njegovim stepenom u grafu. Odatle sledi da je vrednost m ograničena zbirom stepena svih čvorova u grafu. Neka E označava broj grana u grafu, V broj čvorova u grafu i F broj strana grafa. Svaka grana grafa doprinosi po 1 stepenu svakog svog temena, pa važi $m \leq 2E$. Trivijalno, važi i $V \leq 2n + I$. Svaka strana grafa je ograničena sa najmanje tri grane i svaka grana učestvuje u najviše dve strane. Zato važi $F \leq 2E/3$.

Ojlerova formula daje:

$$V - E + F \geq 2$$

pri čemu jednakost važi ako je graf povezan. Odatle sledi:

$$E \leq V + F - 2 \leq (2n + I) + 2E/3 - 2$$

i dalje:

$$E \leq 6n + 3I - 6$$

i

$$m \leq 12n + 6I - 12$$

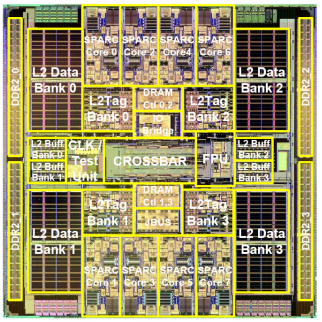
što dokazuje tvrđenje. □

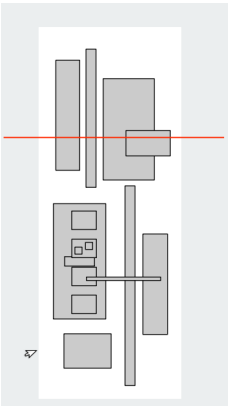
7.2 Određivanje preseka pravougaonika

Problem 7.2. *Za dati skup pravougaonika sa stranicama paralelnim osama, odrediti sve preseke.*

Jedna od najznačajnijih primena navedenog problema je u dizajnu mikroprocesora:

- neke komponente ne smeju da se seku
- između nekih komponenti mora da postoji određeni prostor
- ...

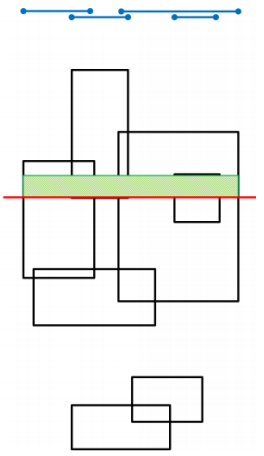




Upotrebom grube sile mogu se odrediti svi preseci n pravougaonika u vremenu $O(n^2)$.

Osnovna ideja naprednijeg algoritma:

- brišuća prava kreće se odozgo nadole (mada, naravno, može i u nekom drugom smeru);
- tačke događaja su gornje i donje tačke svih pravougaonika;
- potrebno je održavati skup duži koje seče brišuća prava;
- ključna operacija: provera da li zadata duž seče neku od duži iz zadatog skupa.



Za proveru da li zadata duž seče neku od duži iz zadatog skupa koristi se algorita opisan u prethodnoj glavi. Ukoliko je stablo pretrage duži balansirano – onda se u vremenu $O(\log n)$ može proveriti da li postoji presek neke duži sa

dužima koje se čuvaju u stablu. Dodatno, u jednoj tački događaja mogu se pronaći svi preseci u vremenu $O(I_y \log n)$ (gde je I_y broj preseka na brišućoj pravoj).

Obrada događaja:

- gornja stranica (L, R) pravougaonika: pretraga duži koje seku duž (L, R)
- donja stranica (L, R) pravougaonika: obriši duž (L, R)

Složenost: $O(n \log n + I \log n)$, gde je I broj preseka.

Postoji i rešenje složenosti $O(n \log n + I)$.

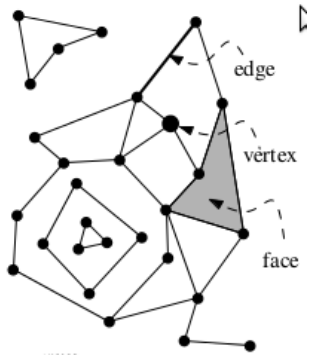
Varijacija: odrediti preseke pravougaonika koji su na rastojanju manjem od d, odrediti preseke pravougaonika u opštem položaju, odrediti preseke krugova,...

Glava 8

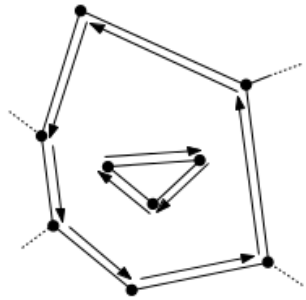
Struktura DCEL

Mape se obično razmatraju kao razlaganja ravni indukovana planarnim utapanjem grafa. Takvo razlaganje je povezano ako je odgovarajući graf povezan. Podrazmevaćemo u nastavku da je svaka ivica grafa deo prave. Strana razlaganja je maksimalni povezani podskup ravni koji ne sadrži tačku na ivici ili čvor. Dakle, strana je otvorena poligonalna oblast čiji rub formiraju ivice i čvorovi. Ako neka ivica čini rub strane, onda kažemo da je ona njemu *incidentna*. *Složenost* razlaganja definiše se kao zbir broja čvorova, broja ivica i broja strana.

Struktura dvostruko povezana lista ivica (eng. doubly-connected edge list – DCEL) sadrži opis svake strane, ivice i čvora razlaganja. Pored geometrijskih i topoloških informacija, taj opis može da sadrži i dodatne informacije ili *atribut* (npr. tip vegetacije za svaku stranu). Za svaku stranu, za svaku ivicu čuva se pokazivač na sledeću i prethodnu (kako bi strana mogla da se obilazi u oba smera). U okviru ivice čuvaju se pokazivači na strane koje ograničava. Pogodno je razmatrati dve različite orijentacije stranice kao par različitih polustranica (eng. half-edges), pa onda svaka polustranica ima jedinstvenu sledeću i jedinstvenu prethodnu polustranicu. Dodatno, polustranice ograničavaju samo jednu stranu (za razliku od stranica koje ograničavaju po dve). Dve polustranice koje odgovarajući jednoj stranici zovemo *blizanci* (eng. twins). Ako je sledeća polustranica definisana u skladu sa ccw orijentacijom, strana koju ograničava ostaje levo kada se ide duž polustranice. Početak jedne polustranice e je kraj suprotne polustranice $Twin(e)$, a njen kraj je početak ove druge. Prateći polustranice može se obići strana koju ograničavaju.



Ukoliko strana sadrži „rupu“, polustranice koje je ograničavaju usmerene su cw (u smeru kazaljke na satu), pa strana ostaje levo za svaku polustranicu koja je ograničava. Polustranice-blizanci uvek su usmerene suportno. Ako strana sadrži rupu, onda nije dovoljno čuvati samo jedan pokazivač iz strane na proizvoljnu stranicu iz koje se može doći do svih drugih stranica te strane - potreban je po jedan pokazivač za svaku komponentu.



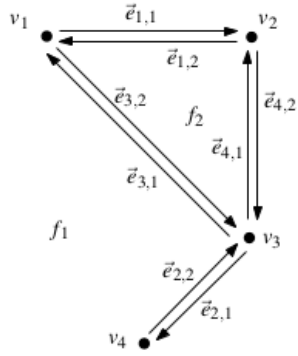
Struktura DCEL sastoji se od tri kolekcije zapisa: jedne za temena, jedne za strane i jedne za polustranice. Ovi zapisi čuvaju sledeće informacije:

- Zapis za čvor v čuva koordinate $Coordinates(v)$, kao i pokazivač $IncidentEdge(v)$ na proizvoljnu polustranicu čiji je v početak.
- Zapis za stranu f sadrži pokazivač $OuterComponent(f)$ na neku polustranicu svoje spoljašnje granice. Za neograničenu stranu, ovaj pokazivač je $NULL$. Sadrži i listu $InnerComponents(f)$ koja za svaku rupu u strani sadrži pokazivač na neku njenu polustranicu.
- Zapis za polustranicu e sadrži pokazivač $Origin(e)$ na njen početak, pokazivač $Twin(e)$ na njenu polustranicu-bliznakinju i pokazivač $IncidentFace(e)$ na stranu koju ograničava. Nema potrebe čuvati kraj polustranice, jer je on jednak $Origin(Twin(e))$. Početak je izabran tako da $IncidentFace(e)$

ostaje sleva prilikom kretanja od početka ka kraju polustranice. Ovaj zapis takođe sadrži pokazivače $Next(e)$ i $Prev(e)$ na sledeću i prethodnu polustranicu na rubu strane $IncidentFace(e)$.

Za svako teme i stranicu koristi se konstantan obim podataka. Strana može da zahteva promenljivu količinu podataka, jer lista $InnerComponents(f)$ ima elemenata koliko ima rupa na strani. Kako na proizvoljnu polustranicu ukazuje najviše jedna strana iz $InnerComponents(f)$, ukupan obim podataka je linearan u odnosu na složenost razlaganja.

Primer 8.1. *Jedan primer strukture DCEL za jedno jednostavno razlaganje je dat u nastavku. Dve polustranice koje odgovaraju stranici* e_i *označene su sa* $\vec{e}_{i,1}$ *i* $\vec{e}_{i,2}$.



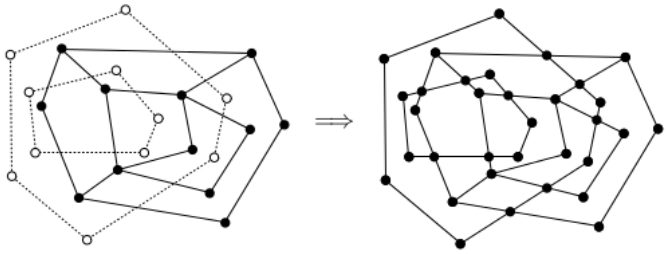
teme	*Coordinates*	*IncidentEdge*
v_1	$(0, 4)$	$e_{1,1}$
v_2	$(2, 4)$	$e_{4,2}$
v_3	$(2, 2)$	$e_{2,1}$
v_4	$(1, 1)$	$e_{2,2}$

strana	*OuterComponent*	*InnerComponents*
f_1	*nil*	$e_{1,1}$
f_2	$e_{4,1}$	*nil*

polustranica	*Origin*	*Twin*	*IncidentFace*	*Next*	*Prev*
$e_{1,1}$	v_1	$e_{1,2}$	f_1	$e_{4,2}$	$e_{3,1}$
$e_{1,2}$	v_2	$e_{1,1}$	f_2	$e_{3,2}$	$e_{4,1}$
$e_{2,1}$	v_3	$e_{2,2}$	f_1	$e_{2,2}$	$e_{4,2}$
$e_{2,2}$	v_4	$e_{2,1}$	f_1	$e_{3,1}$	$e_{2,1}$
$e_{3,1}$	v_3	$e_{3,2}$	f_1	$e_{1,1}$	$e_{2,2}$
$e_{3,2}$	v_1	$e_{3,1}$	f_2	$e_{4,1}$	$e_{1,2}$
$e_{4,1}$	v_3	$e_{4,2}$	f_2	$e_{1,2}$	$e_{3,2}$
$e_{4,2}$	v_2	$e_{4,1}$	f_1	$e_{2,1}$	$e_{1,1}$

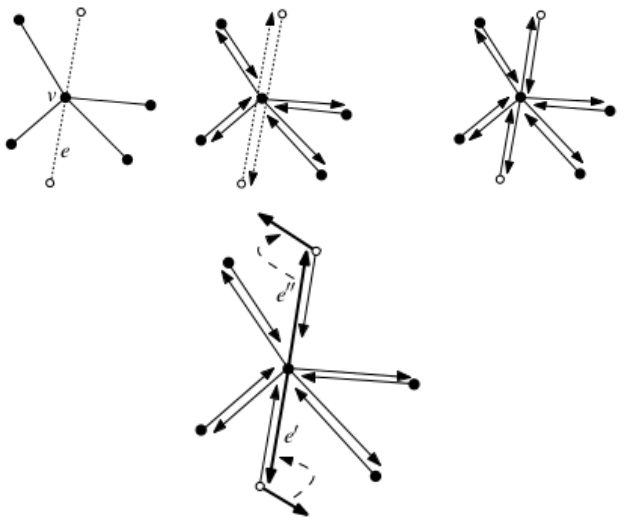
8.1 Izračunavanje preklapanja dva razlaganja

Preklapanje $\mathcal{O}(\mathcal{S}_1, \mathcal{S}_2)$ dva razlaganja $\mathcal{S}_1$ i $\mathcal{S}_2$ indukovano je stranicama ovih razlaganja. Važan problem je kako izračunati DCEL za $\mathcal{O}(\mathcal{S}_1, \mathcal{S}_2)$, ako su raspoložive strukture DCEL za $\mathcal{S}_1$ i $\mathcal{S}_2$. Svaka strana u $\mathcal{O}(\mathcal{S}_1, \mathcal{S}_2)$ treba da bude označena oznakama strana iz $\mathcal{S}_1$ i $\mathcal{S}_2$ koje je sadrže.



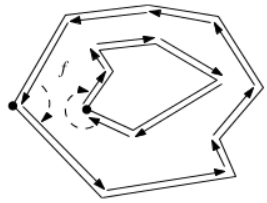
Prvi korak je kopiranje DCEL struktura za $\mathcal{S}_1$ i $\mathcal{S}_2$ u jednu novu, koja treba da bude korigovana. Sve stranice koje pripadaju samo jednom razlaganju postojaće i u rezultatu, dok preseci stranica iz dva razlaganja zahtevaju posebnu akciju. U prvoj fazi se ažuriraju samo liste stranica i temena, a u drugoj lista strana.

Prva faza, u kojoj je potrebno ažurirati spisak temena i stranica, zasnovan je na algoritmu za određivanje preseka duži. Ovaj algoritam modifikuje se tako što ne održava samo strukture $\mathcal{Q}$ i $\mathcal{T}$ već i traženu DCEL strukturu $\mathcal{D}$ (koja inicijalno sadrži liste za $\mathcal{S}_1$ i $\mathcal{S}_2$. Strukturu $\mathcal{D}$ ne treba ažurirati ako i samo ako tačka događaja pripada samo stranicama iz jednog razlaganja. Naredna slika ilustruje situaciju u kojoj stranica iz $\mathcal{S}_1$ prolazi kroz teme iz $\mathcal{S}_2$:



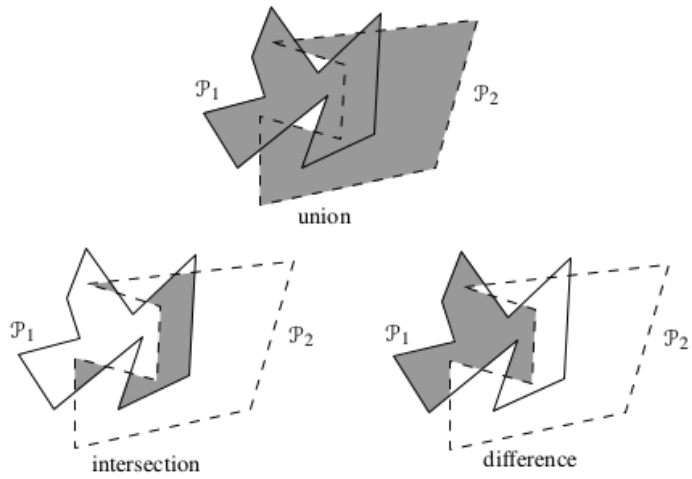
Kada su ažurirane liste temena i stranica, lista strana se ažurira na osnovu njih: osim neograničene strane, svaka strana ima jedinstvenu spoljašnji rub, pa je broj strana jednak broju spoljašnjih rubova plus jedan.

Iz izračunatih lista temena i stranica mogu se odrediti svi rubovi, ali je pitanje da li je ciklus stranica spoljašnji rub ili rub rupe strana. Ovo može biti određeno razmatranjem (najnižeg) najlevljeg temena v ciklusa – ako je ugao između stranica koje su incidentne sa v manji od opruženog, onda je rub spoljašnji, a inače je unutrašnji.



8.2 Bulovske operacije

Da bi se izračunale bulovske operacije nad dva razlaganja, potrebno je najpre izračunati preklapanje ta dva razlaganja. Svaka ograničena strana rezultujućeg razlaganja nosi oznaku $\mathcal{S}_1$ i/ili $\mathcal{S}_2$ u zavisnosti od toga da li je pripadala $\mathcal{S}_1$ i $\mathcal{S}_2$. Ukoliko je, onda, potrebno odrediti presek dva razlaganja – dovoljno je izdvojiti sve strane koje među oznakama imaju i $\mathcal{S}_1$ i $\mathcal{S}_2$. Za uniju je dovoljno izdvojiti one koji imaju bar jednu od oznaka $\mathcal{S}_1$ i $\mathcal{S}_2$. Za (jednu) razliku dovoljno je izdvojiti one koji imaju oznaku $\mathcal{S}_1$ a nemaju oznaku $\mathcal{S}_2$. Svaka od navedenih bulovskih operacija može se izvršiti u vremenu $O(n \log n + k \log n)$, gde je k složenost rezultata.



Glava 9

Triangulacija

Poligonska linija je konačan niz duži. Prost poligon čini zatvorena poligonska linija bez samopresecanja. Prosta poligonska linija deli ravan na unutrašnjost, spoljašnjost i rub. *Dijagonala poligona* je duž čija su temena dva nesusedna temena poligona i čije sve tačke pripadaju unutrašnjosti poligona.

Poligoni se koriste u mnogim geometrijskim aplikacijama (zbog jednostavne reprezentacija i efikasnih algoritama za obradu).

Triangulacija je podela prostog poligona P na nepresecajuće trouglove koristeći samo njegove stranice i dijagonale. Najvažnije primene: modelovanje terena, lociranje položaja, vidljivost, robotika, itd.

Pitanje je da li je triangulacija moguća za proizvoljan prost poligon, a važan problem je efikasno izračunavanje triangulacije (ako postoji).

9.1 Egzistencija

Teorema 9.1. *Za svaki prost poligon postoji triangulacija.*

Dokaz: Dokaz može biti izveden indukcijom.

Ako je $n = 3$, tvrđenje trivijalno važi.

Pretpostavimo da je $n > 3$ i da tvrđenje važi za svaki poligon sa manje od n temena. Neka je V teme poligona sa najmanjom vrednošću x koordinate, a ako ima više takvih, onda ono sa najmanjom vrednošću y koordinate. Neka su U i W susedna temena tog temena. Na osnovu izbora ovih tačaka, sigurno je da one nisu kolinearne, tj. sigurno je da one formiraju trougao.

Duž UW ili čitava pripada unutrašnjosti poligona ili ne. Pretpostavimo da čitava pripada. Tada je ona dijagonala i deli zadati poligon na trougao

Slika 9.1:

VUW i jedan poligon sa $n-1$, za koji, na osnovu induktivne hipoteze, važi da postoji triangulacija. Odatle sledi da i za polazni poligon postoji triangulacija.

Pretpostavimo da ne pripada čitava duž UW unutrašnjosti zadatog poligona. Tada u trouglu VUW (u njegovoj unutrašnjosti ili na rubu) postoji teme zadatog poligona (inače bi neka stranica poligona dva puta sekla duž UW, što je nemoguće). Neka je P takvo teme sa najmanjom vrednošću x koordinate. Tada je čitava duž VP u unutrašnjosti poligona (tj. duži VP je dijagonala), te ga ona razdvaja na dva poligona sa manjim brojem temena. Na osnovu induktivne pretpostavke, tvrđenje važi za njih, pa važi i za zadati poligon. □

Teorema 9.2. *Svaka triangulacija prostog poligona sa n temena ima tačno $n-2$ trouglova.*

Dokaz: Dokaz može biti izveden indukcijom.

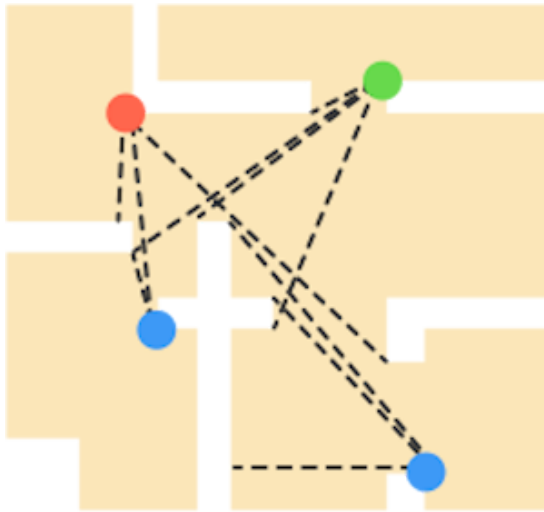
Ako je $n = 3$, tvrđenje trivijalno važi.

Pretpostavimo da je $n > 3$ i da tvrđenje važi za svaki poligon sa manje od n temena. Neka je d proizvoljna dijagonala u nekoj triangulaciji poligona. Ova dijagonala razdvaja poligon na dva poligona sa m_1 i m_2 temena. Krajevi dijagonale d se javljaju u oba ta poligona, pa važi $m_1+m_2 = n+2$. Na osnovu induktivne hipoteze, triangulacije poligona sa m_1 i m_2 temena imaju $m_1 - 2$ i $m_2 - 2$ trouglova, što znači da triangulacija zadatog poligona ima $(m_1 - 2) + (m_2 - 2) = m_1 + m_2 - 4 = n + 2 - 4 = n - 2$ trouglova, što je i trebalo dokazati. □

9.2 Vidljivost i problem umetničke galerije

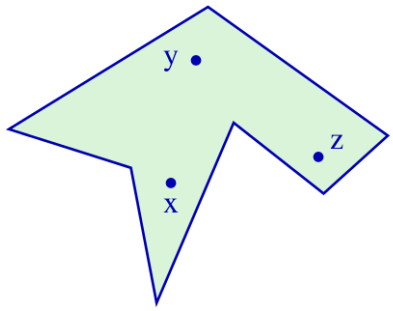
Problem 9.1 (Problem umetničke galerije (art gallery problem)). *Osnovu umetničke galerije čini prost poligon sa n temena. Zadatak je rasporediti čuvare tako da pokrivaju kompletnu galeriju (tj. da svaki deo galerije vidi barem jedan čuvar) i tako da ih ima što manje (svaki čuvar stoji na fiksiranom mestu, može da gleda u svim smerovima oko sebe i ne može da vidi kroz zidove).*

Ovaj problem može da se formuliše i kao problem video nadzora (tj. raspoređivanje sigurnosnih kamera) i relevantan je za mnoge probleme vidljivosti. Problem je formulisao Kli 1973. godine, prvi ga je rešio Čvatal nešto kasnije.



Slika 9.2:

Vidljivost u ravni može da se definiše u čisto geometrijskim terminima: tačka P je u okviru poligona $\mathcal{P}$ vidljiva iz Q ako duž PQ pripada poligonu $\mathcal{P}$. Vidljivost nije tranzitivna, na narednoj slici je poligon sa tačkama X, Y i Z takvim da su X i Y međusobno vidljive, Y i Z su međusobno vidljive, ali X i Z nisu međusobno vidljive.

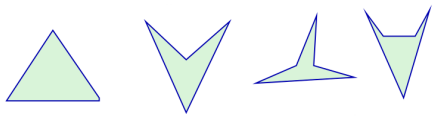


Neka $g(\mathcal{P})$ označava najmanji broj čuvara za konkretan poligon $\mathcal{P}$ i neka $G(n)$ označava maksimum svih vrednosti $g(\mathcal{P})$ za poligone $\mathcal{P}$ koji imaju n temena:

$$G(n) = \max_{|\mathcal{P}|=n} g(\mathcal{P})$$

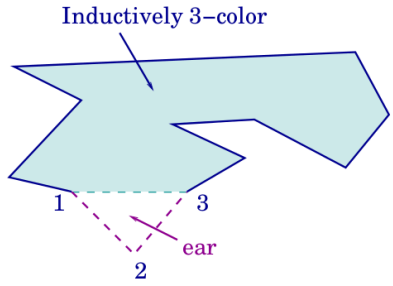
Vrednost $G(n)$ daje najmanji broj čuvara koji je dovoljan za bilo koji n-tougao.

Lako je naslutiti vrednost $G(n)$ za $n = 3, 4, 5$:



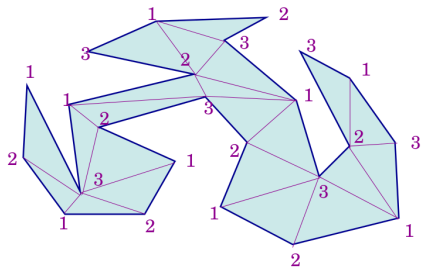
Teorema 9.3. $G(n) = \lfloor n/3 \rfloor$

Dokaz: (Fiskov dokaz:) Poligon može biti triangulisan. Štaviše, ta triangulacija može biti obojena u tri boje (1, 2, 3) tako da svaki trougao ima temena obojena različitim bojama. Ovo tvrđenje može da se dokaže lako indukcijom. Trivijalno važi za $n = 3$, a za $n > 3$ može se eliminisati jedno teme, može se primeniti induktivna pretpostavka i onda pogodno obojiti izbačeno teme.

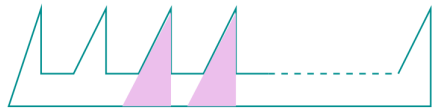


Sa triangulacijom koja je obojena na opisani način, odrediti boju temena koja se pojavljuje najmanji broj puta. Ona se pojavljuje najviše $\lfloor n/3 \rfloor$ puta.

Ako postavimo čuvare u temena obojena tom bojom, oni će videti čitavu galeriju i biće ih najviše $\lfloor n/3 \rfloor$.



Navedena granica ne može biti popravljena: postoje grafovi za koje je neophodno $\lfloor n/3 \rfloor$ čuvara:



□

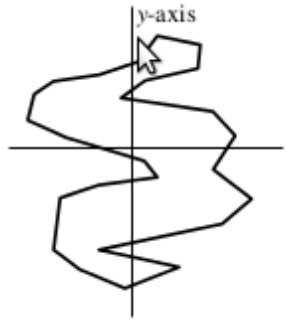
Teorema 9.4 (Teorema o umetničkoj galeriji)**.** *Za prost poligon sa n temena, $\lfloor n/3 \rfloor$ kamera je dovoljno, a u nekim slučajevima i potrebno za vizuelno pokrivanje svih tačaka poligona.*

Problem pronalaženja minimalnog broja kamera za dati poligon je NP-težak.

9.3 Efikasna triangulacija

Dokaz teoreme 9.2 vodi ka jednom pristupu za triangulaciju poligona. U svakom koraku se detektuje jedna dijagonala i problem se svodi na problem manje dimenzije. Ako poligon ima n temena, onda ovaj korak ima složenost $O(n)$. Nakon ovog koraka, problem može biti sveden na problem za $n-1$, pa je ukupna složenost $O(n^2)$. Postoje, međutim, i efikasniji algoritmi. Postoji više algoritama složenosti $O(n \log n)$. Čazel je 1991. godine razvio algoritam složenosti $O(n)$, ali je on „nerazumno komplikovan“.

U nekim specijalnim slučajevima, na primer, za konveksni poligon, triangulacija je trivijalna – za konveksni poligon, može biti izvršena u vremenu $O(n)$. Ideja, dakle, može da bude da se najpre razloži na konveksne delove, a da se onda triangulacija primeni na njih. Međutim, nije lako razložiti poligon na konveksne delove i, umesto toga, poligon će biti razložen na takozvane monotone delove. U nastavku će biti opisan algoritam složenosti $O(n \log n)$ za triangulaciju zasnovan na ovoj ideji.

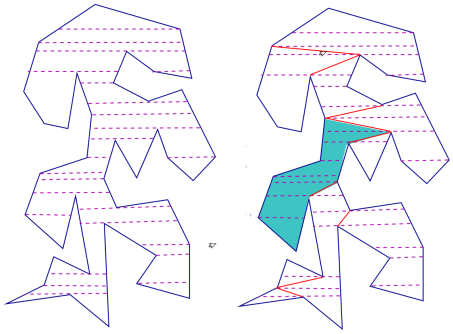


Za prost poligon se kaže da je *monoton* u odnosu na pravu l ako je presek poligona i svake prave normalne na l povezan skup (tj. jedna duž, tačka ili prazan skup). Poligon koji je monoton u odnosu na y-osu se naziva y-monoton poligon. Za y-monoton poligon važi: ako se krećemo od najvišeg do najnižeg temena duž levog lanca stranica poligona, uvek ćemo se kretati naniže ili horizontalno (nikada naviše). Ako to nije tako – poligon nije monoton.

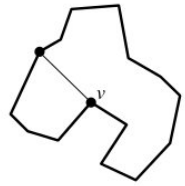
Triangulacija poligona može biti izvršena tako što će poligon biti podeljen na monotone delove, a onda će biti izvršena njihova triangulacija. Ukupna složenost će biti $O(n \log n)$.

9.3.1 Podela poligona na monotone delove

Osnovna ideja: konstruisati prave kroz temena poligona. Svaki član takvog razlaganja je trapez (koji može biti degenerisan u trougao). Ovo omogućava jednostavno eliminisanje temena koja narušavaju monotonost.

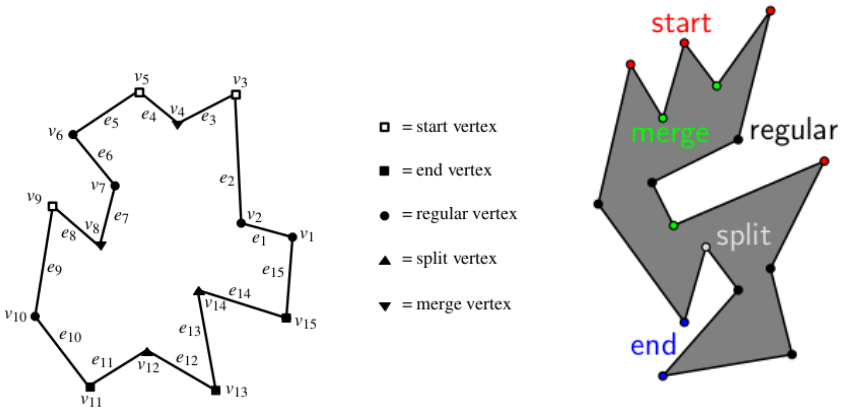


Pretpostavimo da se krećemo od najvišeg do najnižeg temena duž levog lanca stranica poligona. Teme u kojem se smer kretanje menja od naniže (ili horizontalno) na naviše (ili obratno) naziva se teme okreta (eng. turn vertex). Ovakva temena ne postoje u monotonim poligonima i ako želimo da poligon razložimo na monotone delove potrebno je da se rešimo temena okreta. To se može postići dodavanjem pogodnih dijagonala. Ako su u temenu okreta v obe stranice nadole, a poligon lokalno gore, onda je potrebno dodati neku dijagonalu nagore koja razdvaja poligon na dva dela. Tada su u oba poligona u temenu v po jedna stranica nagore i jedna nadole, pa ono više nije teme okreta.



Razlikujemo pet vrsta temena:

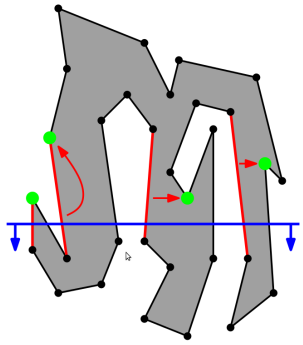
- "start" teme: ako su dva suseda dole i unutrašnji ugao je manji od opruženog;
- "split" teme: ako su dva suseda dole i unutrašnji ugao je veći od opruženog;
- "end" teme: ako su dva suseda gore i unutrašnji ugao je manji od opruženog;
- "merge" teme: ako su dva suseda gore i unutrašnji ugao je veći od opruženog;
- "regularno" teme: inače;



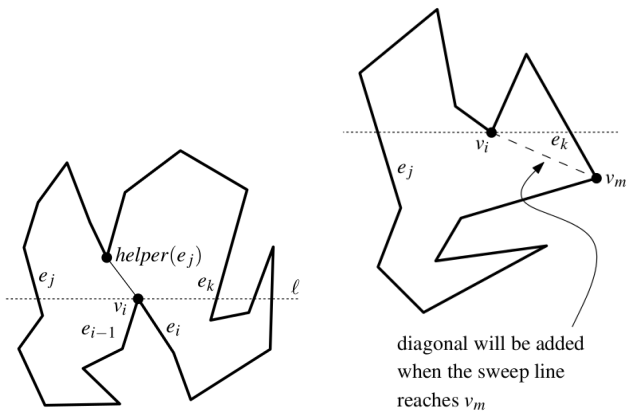
Lema 9.1. *Ako poligon nema split ni merge temena onda je on y-monoton.*

Eliminisanje temena okreta i, time, razlaganje poligona na y-monotone poligone može se realizovati metodom brišuće prave. Brišuća prava kreće se odozgo nadole a tačke događaja su samo temena poligona. Kada se naiđe na novo teme, potrebno je obraditi događaj. Temena mogu biti sortirana (po y koordinati, a ako ima više takvih - po x koordinati) na početku algoritma, pa se onda određivanje sledećeg događaja može uraditi u vremenu $O(1)$.

Kada se naiđe na split teme v, potrebno je spojiti ga novom dijagonalom sa pogodno odabranim temenom iznad dijagonale. To pogodno teme je tzv. *helper* za stranicu neposredno levo od temena v. Teme *helper* potrebno je određivati (i ažurirati) samo za stranice od kojih je poligon lokalno desno. Teme $helper(e)$ za stranicu e za koju je poligon lokalno desno, je najniže teme v iznad brišuće prave takvo da je horizontalna duž koja spaja v sa stranicom e čitava pripada unutrašnjosti poligona. Teme $helper(e)$ može da se menja kako se brišuća prava pomera. Primetimo da $helper(e)$ može da bude gornje teme stranice e. Može se dokazati da duž koja spaja split teme i odgovarajući *helper* čitava pripada unutrašnjosti poligona.



Kada se naiđe na split teme v, potrebno je naći stranicu e na brišućoj pravoj neposredno levo od v i spojiti v sa $helper(e)$, onda postaviti $helper(e)$ na v, i postaviti $helper$ za stranicu kojoj je v gornje teme na v.



obrada split temena i merge temena

Potrebno je na odgovarajući način obraditi druge vrste događaja (za druge vrste temena).

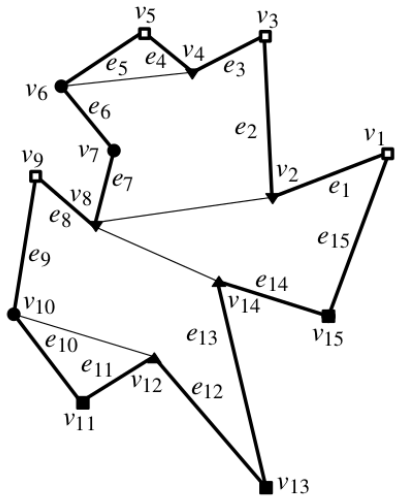
Merge teme v potrebno je spojiti sa najvišim temenom između stranica koje su neposredno oko v na brišućoj pravoj. Problem je, međutim, što to teme ne može biti poznato unapred, u tretutku kada se naiđe na v. Međutim, ono će biti pronađeno kasnije: kada naiđemo na teme v' koje zamenjuje v kao $helper$ stranice – to je upravo traženo teme. (Ovo nije slučajnost – merge temena su split temena ako se poligon razmatra odgozdo nagore.) Dakle, uvek kada se promeni $helper$ za neku stranicu, proveravamo da li je stari $helper$ merge teme i, ako jeste, dodajemo dijagonalu između starog i novog. Ta dijagonala se dodaje uvek ako je novi $helper$ split teme, pa se u tom slučaju jednom dijagonalom rešavamo i jednog merge i jednog split temena.

Strukture podataka

Red događaja $\mathcal{Q}$ može biti uređena lista temena.

Status $\mathcal{T}$ čuva stranice kojima je poligon lokalno desno, zajedno sa svojim *helper* temenom, sortirane sleva nadesno prema tački preseka sa brišućom pravom. Status može biti implementiran kao uređeno binarno stablo. Sadržaj ove strukture menja se dok se brišuća prava pomera: stranice mogu biti dodavane, izbacivane i njihov *helper* može biti menjan.

Podrazumeva se da je zadati poligon reprezentovan kao DCEL. Izabrane dijagonale (i indukovani potpoligoni) se dodaju u tu strukturu.



Algoritam

Algoritam: RazložiNaMonotonePoligone

Ulaz: Prost poligon $\mathcal{P}$ reprezentovan kao DCEL $\mathcal{D}$.

Izlaz: Razlaganje na monotone poligone, reprezentovano kao DCEL $\mathcal{D}$.

1: Inicijalizuj prazan red događaja $\mathcal{Q}$. Ubaci sva temena duži u $\mathcal{Q}$, sortirano po y koordinati, ako postoji više sa istom vrednošću, sortiraj ih po x.
2: Inicijalizuj praznu strukturu za status $\mathcal{T}$.
3: **dok god** red $\mathcal{Q}$ nije prazan
4: Uzmi sledeću tačku događaja P iz $\mathcal{Q}$ i obriši je.
5: ObradiDogađajPoTipuTemena(P)

Algoritam: ObradiSplitTeme(v)

1: Pronađi u $\mathcal{T}$ stranicu e neposredno levo od v.
2: Dodaj dijagonalu $v - helper(e)$ u $\mathcal{D}$.
3: Postavi $helper(e)$ na v.
4: Dodaj u $\mathcal{T}$ (desnu) stranicu čiji je gornje teme v i postavi njen $helper$ na v.

Algoritam: ObradiMergeTeme(v)

1: **ako je** e stranica čije je donje teme v a poligon joj je lokalno desno i $helper(e)$ je merge teme **onda**
2: dodaj dijagonalu $v - helper(e)$.
3: Izbaci e iz $\mathcal{T}$.
4: Pronađi u $\mathcal{T}$ stranicu e' neposredno levo od v.
5: **ako je** $helper(e')$ merge teme **onda**
6: dodaj dijagonalu $v - helper(e')$.
7: Postavi $helper(e')$ na v.

Algoritam: ObradiStartTeme(v)

1: Dodaj e u $\mathcal{T}$ i postavi $helper(e)$ na v (e je stranica levo).

Algoritam: ObradiEndTeme(v)

1: **ako je** $helper(e)$ merge teme (e je stranica gore levo) **onda**
2: Dodaj dijagonalu $v - helper(e)$.
3: Izbaci e iz $\mathcal{T}$

Algoritam: ObradiRegularnoTeme(v)

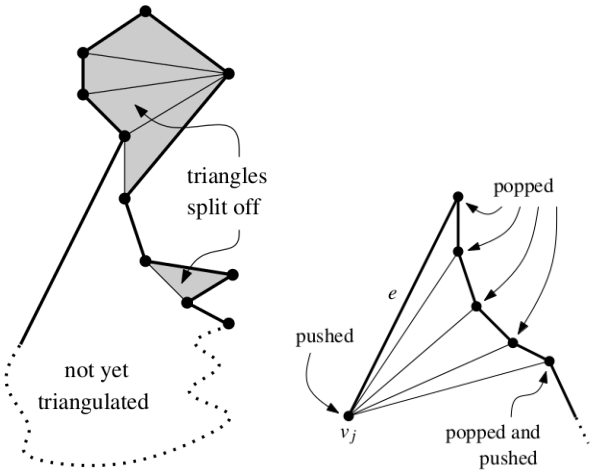
1: **ako je** poligon $\mathcal{P}$ je desno od v (e je stranica gore, e' stranica dole) **onda**
2: **ako je** $helper(e)$ je merge teme **onda**
3: Dodaj dijagonalu $v - helper(e)$.
4: Izbaci e iz $\mathcal{T}$.
5: Dodaj e' u $\mathcal{T}$ i postavi $helper(e')$ na v.
6: **inače**
7: Nađi u u $\mathcal{T}$ stranicu e'' neposredno levo od v.
8: **ako je** $helper(e“)$ je merge teme **onda**
9: Dodaj dijagonalu $v - helper(e“)$.
10: Postavi $helper(e'')$ na v

9.3.2 Triangulacija y-monotonih poligona

Triangulacija y-monotonog poligona vrši se metodom brišuće prave, brišuća prava ide odozgo nadole a događaji su temena poligona. U svakom događaju moguće je da u triangulaciju budu dodati novi trouglovi i oni ubuduće ne treba da budu razmatrani. Treba da bude razmatran samo tekući deo koji nije triangulisan. Primenom narednog algoritma, taj deo je uvek povezan i ima uvek istu formu, tzv. formu levka.

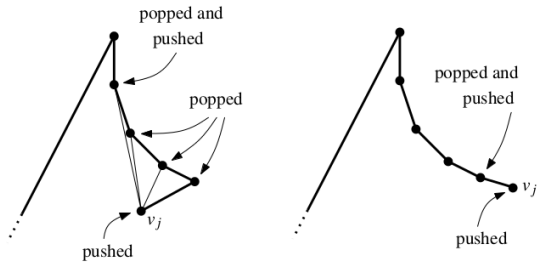
U algoritmu će biti korišćen stek i na njemu će biti temena koja su posećena ali koja će učestvovati u još dijagonala. Osnovna ideja je da se u svakom temenu dodaje što više dijagonala nagore (koristeći stek za posećivanje temena redom suprotnim od njihovog posećivanja).

Ivarijanta algoritma je da netriangulisan deo poligona ima formu levka: jedan njegov rub sastoji se od dela jedne stranice poligona, a drugi njegov rub sastoji se od lanca temena u kojima je unutrašnji ugao veći ili jednak opruženom uglu.



Kada se naiđe na novo teme v (tj. na novi događaj), postoje dve mogućnosti:

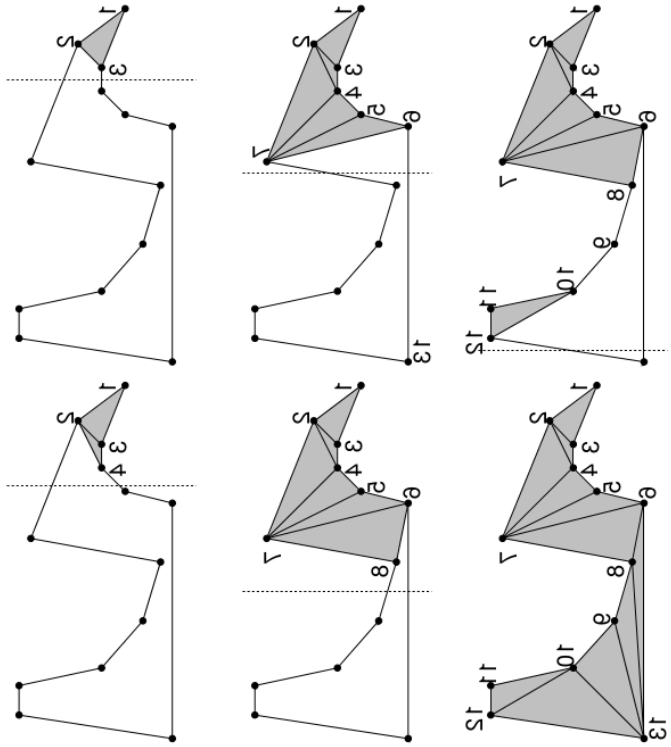
v se nalazi na suprotnom rubu od niza temena u levku: u tom slučaju, v mora biti donje teme stranice koja ograničava levak. Moguće je (i potrebno) dodati dijagonale od v do svih temena u levku osim poslednjeg (sa kojim je v već povezan). Sva temena levka se brišu sa steka i na njegov vrh se dodaju najniže teme levka i teme v. Netriangulisani deo ponovo ima oblik levka (ali na drugu stranu).



v se nalazi na rubu na kojem je niz temena u levku: dodaju se dijagonale iz v do temena iz levka tj. sa steka dok god je to moguće (pri čemu se preskače prvo, jer je već povezano sa v). Kada više nije moguće dodati dijagonalu, poslednje razmatrano teme vraća se na stek, a onda se na stek stavlja i v. I u ovom slučaju netriangulisani deo (i dalje) ima oblik levka.

Analiza složenosti Od dva niza temena, sortirana lista može se dobiti u vremenu $O(n)$ (slično kao u merge sortiranju). Prolazi se kroz n tačaka i u

svakom događaju dodaju se najviše dve tačke na stek. Zato ni skidanja sa steka ne može biti više nego $2n$, pa je ukupno vreme $O(n)$.

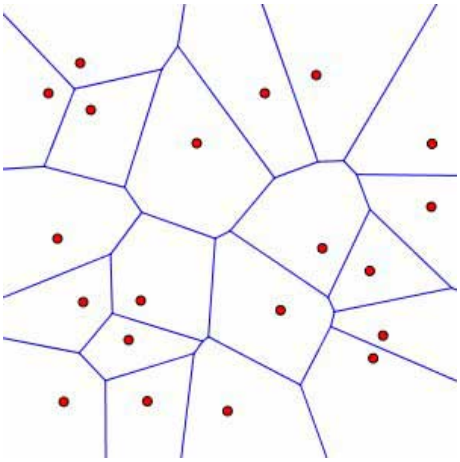


Glava 10

Voronoj dijagrami

Voronoj dijagrami su često korišćena geometrijska struktura u čijoj osnovi je pitanje bliskih tačaka datoj tački. Na primer, Voronoj dijagrami korisni su za sistem koji korisniku treba brzo da odgovori koja picerija mu je najbliža u tom trenutku. Mnoge prirodne i društvene pojave mogu se opisati i objasniti u terminima Voronoj dijagrama: u antropologiji, oblasti pod uticajem neke kulture, u kristalografiji, struktura kristala i metala, u ekologiji, nadmetanje između biljaka, u ekonomiji, modeli tržišta itd.

Knutova formulacija: postoji n pošta u gradu i svaki korisnik koristi poštu koja mu je najbliža (jednostavnosti radi, pretpostavlja se da je vreme putovanja do pošte proporcionalno euklidskom rastojanju). Pitanje je kako izgleda oblast koju opslužuje jedna pošta. Ako treba da se otvori nova pošta, gde bi bilo najbolje da se ona nalazi? Itd.



Problem 10.1. *Neka je dat skup tačaka u ravni: $\{P_1, P_2, \ldots, P_n\}$, koje zovemo "sedišta". Za svaku tačku ravni potrebno je odrediti koje joj je sedište najbliže. Iako se mogu koristiti i druge funkcije rastojanja, obično se podrazumeva euklidsko rastojanje između tačaka:*

$$d(P, Q) = \sqrt{(x_P - x_Q)^2 + (y_P - y_Q)^2}$$

Tačke kojima je jedno isto sedište najbliže čine jednu oblast. Opštije, relacija: "dve tačke su u relaciji ako i samo ako im je isto sedište najbliže" je relacija ekvivalencije i deli ravan na skup oblasti. Rub oblasti čine tačke koje imaju više nego jedno najbliže sedište.

Funkcija koja svakoj tački ravni pridružuje najbliže sedište naziva se funkcija Voronojove dodele.

Definicija 10.1 (Voronoj dijagram)**.** *Voronoj dijagram $Vor(\mathcal{P})$ skupa tačaka $\mathcal{P} = \{P_1, P_2, \ldots, P_n\}$ je razlaganje ravni na oblasti (regione, ćelije) takve da tačka X pripada oblasti tačke P_i ako i samo ako važi*

$$d(X, P_i) < d(X, P_j), \text{ za svako } i \neq j$$

Oblast tačke P_i naziva se Voronoj ćelija tačke P_i i označava sa $v(P_i)$.

Pod Voronoj dijagramom se ponekad smatra samo skup grana i temena razlaganja. Tada, kada se kaže da je Voronoj dijagram povezan – misli se na to da je skup grana povezan.

Voronoj dijagram pruža mnoštvo korisnih informacija. Na primer, ako su dve oblasti susedne, onda je očekivano da se odgovarajuća sedišta bore za klijente u graničnom regionu.

Voronoj dijagrami, Delone triangulacije i konveksni omotači su međusobno blisko povezani.

10.1 Svojstva Voronoj dijagrama

Važno pitanje je kakav oblik imaju Voronoj ćelije, koja geometrijska i kombinatorna svostva imaju, (da li su one konveksne, povezane, ...), koliko efikasno mogu biti izračunate, itd.

Jednostavna a važna činjenica je da se za dve date tačke P i Q simetrala duži PQ sastoji od tačaka koje su na jednakim rastojanjima od P i Q, sa jedne njene strane su tačke koje su bliže P a sa druge, tačke koje su bliže tački Q. Ta simetrala određuje poluravni $h(P, Q)$ i $h(Q, P)$. Neka je sa $h(P, Q)$ označena (otvorena) poluravan koja sadrži tačke koje su bliže tački P nego tački Q, a sa $h(Q, P)$ (otvorena) poluravan koja sadrži tačke koje su bliže tački Q nego tački P.

Svaka Voronoj ćelija je presek nekoliko poluravni. Preciznijie, ćelija $v(P_i)$ jednaka je preseku poluravni $h(P_i, P_j)$, za svako $j \neq i$, tj.

$$v(P_i) = \cap_{1 \leq j \leq n, j \neq i} h(P_i, P_j)$$

Zbog navedenog, svaka Voronoj ćelija je konveksna figura (pri čemu može biti neograničena). Za zadatih n sedišta, bilo koja Voronoj ćelija može imati najviše $n - 1$ stranica.

Teorema 10.1. *Pretpostavimo da skup $\mathcal{P}$ sastoji od n sedišta. Ako su sva sedišta kolinearna, onda se $Vor(\mathcal{P})$ sastoji od $n-1$ paralelni pravih. Inače, dijagram $Vor(\mathcal{P})$ je povezan (tj. skup grana je povezan) i svaka grana je ili duž ili poluprava.*

Teorema 10.2. *Broj temena Voronoj dijagrama za n sedišta je najviše $2n - 5$ a broj grana je najviše $3n - 6$.*

Dokaz: Ako su sva sedišta kolinearna, onda tvrđenje trivijalno važi. Inače, dijagram je povezan i ta činjenica biće korišćena u nastavku dokaza.

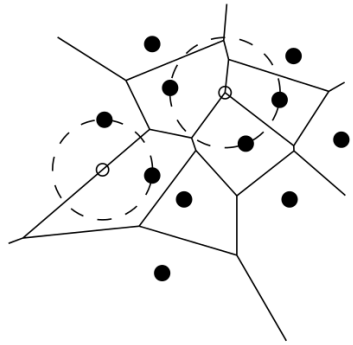
Neke od grana su poluprave, pa kako ne formiraju graf, dodajemo „beskonačno daleko teme“, kao kraj svih takvih grana. Neka je V broj temena Voronoj dijagrama, pa je broj temena u grafu sa beskonačno dalekim temenom jednak $V+1$. Neka je E broj grana i F broj oblasti. Za n sedišta, F je jednako n. Zbir stepenova svih temena je $2E$. Kako svako teme ima stepen barem 3, važi $2E \geq 3(V + 1)$, odakle sledi $V + 1 \leq 2E/3$. Na osnovu Ojlerove formule važi $(V + 1) - E + F = 2$ (graf je povezan!). Odatle dalje sledi $2 = (V + 1) - E + F \leq 2E/3 - E + F$ i $2 \leq -E/3 + n$ odakle je $E \leq 3n-6$. Dodatno, $V+1 \leq 2E/3 \leq 2n-4$, pa je $V \leq 2n-5$. □

Grane Voronoj dijagrama pripadaju simetralama duži određenih sedištima i temena dijagrama su preseci simetrala. Međutim, broj simetrala je kvadratan, a broj grana u dijagramu samo linearan u odnosu na broj sedišta. Zaista, ne određuju sve simetrale grane dijagrama, niti svi preseci simetrala određuju temena. Može se izvršiti karakterizacija simetrala i njihovih preseka koji definišu temena.

Za tačku Q, definišemo najveći prazan krug tačke Q u odnosu na skup $\mathcal{P}$, u oznaci $C_{\mathcal{P}}(Q)$, kao najveći krug sa središtem Q čija unutrašnjost ne sadrži nijedno sedište iz skupa $\mathcal{P}$.

Teorema 10.3. *Za Voronoj dijagram $Vor(\mathcal{P})$ skupa sedišta $\mathcal{P}$ važi:*

- *Tačka Q je teme dijagrama $Vor(\mathcal{P})$ ako i samo ako rub kruga $C_{\mathcal{P}}(Q)$ sadrži tri ili više sedišta.*
- *Simetrala duži P_iP_j za sedišta P_i i P_j definiše granu dijagrama $Vor(\mathcal{P})$ ako i samo ako postoji tačka Q na simetrali takva da rub kruga $C_{\mathcal{P}}(Q)$ sadrži P_i i P_j ali nijedno drugo sedište.*



Teorema 10.4. *Ćelija sedišta P je neograničena akko P pripada konveksnom omotaču datog skupa tačaka.*

Dokaz: Ako P pripada konveksnom omotaču, postoji prava p kroz P takva da su sva druga sedišta sa iste njene strane. Onda svaka tačka na polupravoj q koja je u P normalna na p (i sa druge strane prave p je u odnosu na druga sedišta) ima za najbliže sedište upravo tačku P. Dakle, ćelija za P je neograničena jer sadrži polupravu q. (Ovo važi i u degenerisanom slučaju kada postoje tri kolinearna sedišta na konveksnom omotaču).

Obratno, ako je ćelija za P neograničena, onda ona sadrži neku polupravu q sa temenom P. Neka je p prava koja je normalna na ovu polupravu. Svako sedište Q mora biti sa druge strane p u odnosu na q. Zaista, u suprotnom bi na polupravoj q postojala tačka koja je bliža tački Q nego tački P. Iz ovoga sledi da P mora da pripada konveksnom omotaču. □

10.2 Izračunavanje Voronoj dijagrama

Ulaz za algoritam izračunavanja Voronoj dijagrama je skup $\mathcal{P}$ koji čini n sedišta iz skupa tačaka $\mathbf{R}^2$, a izlaz je skup grana dijagrama.

Najjednostavnije rešenje za konstruisanje Voronoj dijagrama zasnovano je na gruboj sili i na razmatranju svih parova tačaka. Konstruisanje Voronoj dijagrama je jednostavno: za određivanje ćelije $v(P_i)$ dovoljno je odrediti presek svih poluravni $h(P_i, P_j)$, za svako j takvo da je $j \neq i$. To se može uraditi u vremenu $O(n \log n)$, a za sve ćelije u vremenu $O(n^2 \log n)$.

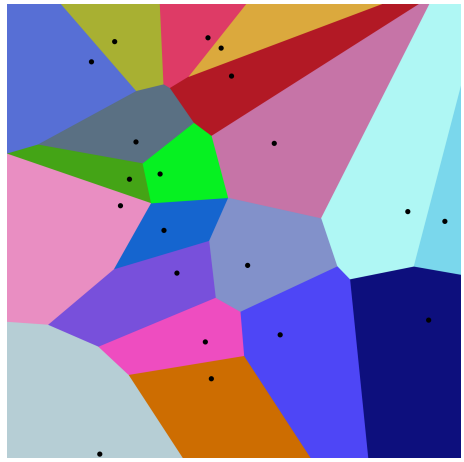
Navedeni pristup je jednostavan, ali nedovoljno efikasan jer uključuje mnogo nepotrebnog rada: sedišta koja su daleko od P_i nemaju uticaj na ćeliju $V(P_i)$. Zaista, za jednu Voronoj ćeliju obično je relevantan samo mali broj drugih središta.

Veoma elegantan i efikasan algoritam, zasnovan na *metodi brišuće prave* (engl. sweep paradigm), otkrio je Stiv Forčen (Steve Fortune) 1987. godine. Algoritam se često koristi u praksi. On računa dijagram u vremenu $O(n \log n)$. Problem sortiranja n realnih brojeva je svodiv na problem izračunavanja Voronoj dijagrama, pa svaki algoritam za izračunavanje Voronoj dijagrama mora da ima barem složenost $\Omega(n \log n)$. Odatle sledi da je Forčenov algoritam asimptotski optimalan.

10.3 Forčenov algoritam za konstruisanje Voronoj dijagrama

Forčenov algoritam zasnovan je na metodi „brišuće prave“. Horizontalna prava l pomera se odozgo nadole i usput konstruiše Voronoj dijagram.

Zamislimo da je svakom sedištu pridružena druga boja i da je cilj da se svaka tačka ravni oboji bojom najbližeg sedišta.



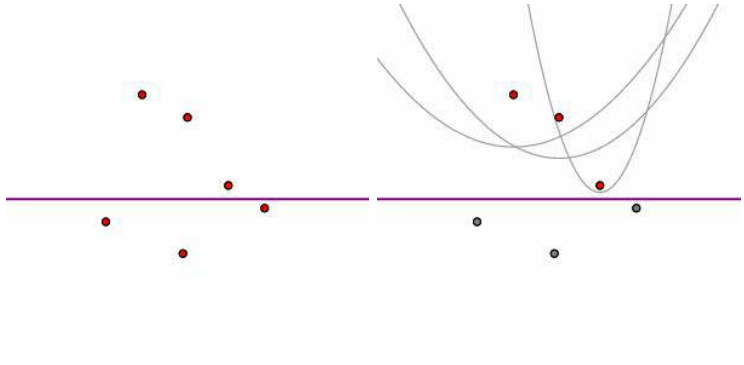
Algoritam ne garantuje da su ispravno obojene sve tačke iznad brišuće prave. Naime, u nekom trenutku, za tačku X na brišućoj pravoj još se uvek možda nije naišlo na najbliže sedište. Zbog toga pristup brisanja ne može da se koristi u jednostavnom obliku (neće se održavati preseci brišuće prave sa Voronoj dijagramom) i mora biti obogaćen dodatnim tehnikama. Ne garantuje se da su ispravno obojene sve tačke iznad brišuće prave, već slabija invarijanta:

ispravno je obojena svaka tačka X iznad brišuće prave l za koje je i najbliže sedište takođe iznad prave l.

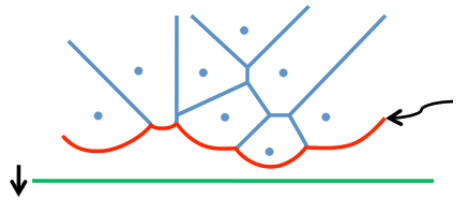
Za koje tačke X se može utvrditi da najbliže sedište nije ispod prave l? Ako važi $d(X,l) > d(X,P)$ za neko sedište P iznad l, onda nijedno sedište ispod l ne može biti najbliže sedište za X (jer je sedište P svakako bliže). Ova provera može se opisati u terminima parabole. Naime, parabola je skup tačaka sa svojstvom:

$$\Pi(P,l) = \{\ X\ \mid\ d(X,P) = d(X,l)\ \}$$

gde je P žiža i l je direktrisa parabole. Dakle, ako je tačka X „unutar“ parabole $\Pi(P,l)$ i Q je bilo koje sedište ispod l, onda nužno važi $d(X,Q) > d(X,P)$ i sve tačke unutar parabole mogu biti pouzdano obojene (ne nužno bojom sedišta P, već možda bojom nekog drugog sedišta na koje se već naišlo). Za tačke ispod parabole tek će biti određeno kako treba da budu obojene. Na primer, za skup sedišta i brišuću pravu kao na slici levo, biće razmatrane parabole prikazane na slici desno.

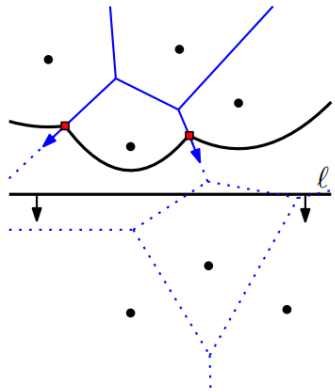


Svako sedište koje se nalazi iznad l ima svoju parabolu. Dok se brišuća prava pomera nadole, parabole se menjaju („otvaraju se“). *Paraboličkі front* ili *linija plaže* (engl. beach line) je rub menjajućih parabola $\Pi(\mathcal{P},l)$ za skup tačaka $\mathcal{P}$ koje su iznad l. Čitav front je uvek iznad prave l i menja se dok se pomera prava l. Frontu može pripadati više različitih lukova iste parabole. Sve tačke iznad fronta mogu biti pouzdano obojene tj. njihova boje neće se menjati bez obzira na tačke koje su ispod (trenutne) prave l. Algoritam, suštinski, prati menjanje fronta dok se brišuća prava kreće kroz skup sedišta.



Linija plaže može se opisati i na sledeći način: svako sedište iznad prave l određuje jednu parabolu, linija plaže je funkcija koja za svaku x koordinatu prolazi kroz najnižu tačku među svim tim parabolama. Dakle, na jednoj x koordinati postoji samo jedna tačka fronta.

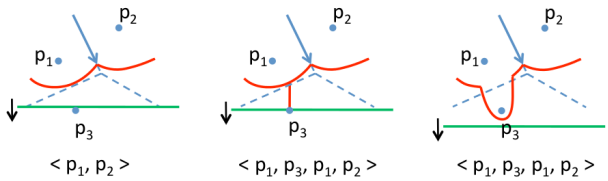
Preseci susednih paraboličkih lukova idu duž Voronoj ivica. Zaista, tačka preseka susednih parabola je na istim rastojanjima od dva sedišta, pa tačke na frontu koje su tačke preseka dva parabolička luka pripadaju ivicama Voronoj dijagrama. Dakle, da bi bio konstruisan Voronoj dijagram, dovoljno je pratiti tačke na frontu koje pripadaju po dvema parabolama.



Linija fronta se menja dok se pomera brišuća prava ali nikada se ne čuva eksplicitno (već se samo čuva lista sedišta koja odgovaraju paraboličkim lukovima na frontu). Parabole se menjaju neprekidno kako se kreće prava l, ali suštinske izmene (u frontu i kreiranom dijagramu) dešavaju se samo kada neka parabola nastaje (onda kada brišuća prava naiđe na novo sedište) ili nestaje (onda kada se njen luk pretvori u tačku i onda nestane, te više nije deo zajedničkog fronta). To su (jedina) dva tipa događaja (događaj sedišta i događaj kruga) koje treba obraditi tokom kretanja prave l.

Događaj sedišta: nastajanje nove parabole Nova parabola nastaje (jedino) kada prava l prođe novo sedište i odgovarajući novi parabolički luk potrebno je dodati u front. Ovo je lako detektovati, jer se lako može održavati lista sedišta uređena po y koordinatama.

Naredna ilustracija prikazuje dodavanje novog luka u front kada prava l naiđe na sedište P_3:



Pre nailaska, lista sedišta koje određuju front bila je $\ldots P_1, P_2 \ldots$, a posle $\ldots P_1, P_3, P_1, P_2 \ldots$.

Generalno, kada prava l prođe kroz novo sedište, kreira se nova parabola i njeni preseci sa frontom, tj. sa jednim paraboličkim lukom iz fronta. Isprva je parabolički luk degenerisan u duž, i dva preseka su identična, a onda se (kako se pomera prava l) kreću u suprotnim smerovima određujući istu buduću granu Voronoj dijagrama. Ta nova, menjajuća grana nije inicijalno povezana sa ostatkom tekućeg dijagrama, ali će postati.

Ukoliko normala kroz novo središte prolazi kroz presek dve parabole iz fronta, novi luk se umeće u luk koji odgovara bližem sedištu.

Može se dokazati da front dobija novi luk samo kada prava l prođe novo sedište.

Prilikom nailaska prave l na sedište uvodi se jedna parabola i najviše dodatna dva paraboličkа luka, te front može imati najviše $2m - 1$ lukova (za prvo sedište kreira se samo jedan luk), gde je m broj sedišta iznad prave l.

Kada se naiđe na sedište P_i određuje se presek vertikalne linije sa frontom da bi se odredio luk koji će biti razdvojen. Neka je P_j njegovo sedište. Kada se razdvoji luk, biće zamenjen nizom lukova P_j, P_i, P_j koji će biti umetnut u front umesto starog luka. Takođe, kreira se (nedovršena) grana između P_i i P_j. Neke trojke koje uključuju P_j će možda biti obrisane a neke će biti dodate. Na primer, pretpostavimo da je pre umetanja front bio sačinjen od lukova koji odgovaraju sedištima: P_1, P_2, P_j, P_3, P_4. Dodavanje luka p_i neka razdvaja luk p_j na dva dela, označena sa p'_j i p''_j. Ovo su različiti i razdvojeni lukovi iako imaju isto sedište. Novi niz lukova je onda $p_1, p_2, p'_j, p_i, p''_j, p_3, p_4$.

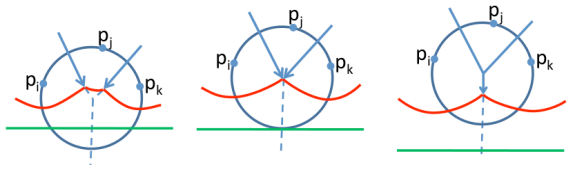
Svaki događaj pridružen trojki P_2, P_j, P_3 biće obrisan. Dodatno, biće razmotreno dodavanje događaja za trojke P_2, P'_j, P_i i P_i, P''_j, P_3. Trojka P'_j, P_i, P''_j uključuje dva ista sedišta i ne može da kreira novi događaj.

Događaj kruga: nestajanje jedne parabole Događaj drugog tipa je kada se neki postojeći parabolički luk pretvori u tačku i onda nestane iz fronta. Susedni lukovi tom luku ne mogu biti delovi istog paraboličkog luka, te se može smatrati da tri susedna luka odgovaraju sedištima P_i, P_j i P_k. U trenutku kada srednji luk nestane, parabole koje odgovaraju ovim sedištima imaju zajedničku tačku Q. Ta tačka je na jednakom rastojanju od tačaka P_i, P_j i P_k, pa postoji krug sa središtem Q koji prolazi kroz ove tri tačke. Štaviše, taj krug dodiruje brišuću pravu l (u tački kruga sa najmanjom y koordinatom). U unutrašnjosti tog kruga ne može biti sedišta (u suprotnom bi tački Q bilo bliže to sedište nego prava l, što je u suprotnosti sa činjenicom da Q pripada frontu). Odatle sledi da je Q teme Voronoj dijagrama. To je u skladu sa zapažanjem da preseci parabola idu duž grana Voronoj dijagrama. Dakle, kada jedan parabolički luk nestane

iz fronta i dve presečne tačke lukova se susretnu, tada se takođe susretnu i dve grane Voronoj dijagrama. Ovaj događaj, kada brišuća prava dodirne najnižu tačku opisanog kruga tri sedišta koja određuju susedne lukove u frontu zovemo *događaj kruga*.

Može se dokazati da parabolički luk iz fronta može da nestane samo u ovom slučaju. Dakle, parabola nestaje (jedino) kada njen luk na frontu bude „progutan“ od strane dva susedna luka, tj. kada dužina jednog luka postane nula. Tada će na tom mestu biti kreirano novo teme Voronoj dijagrama.

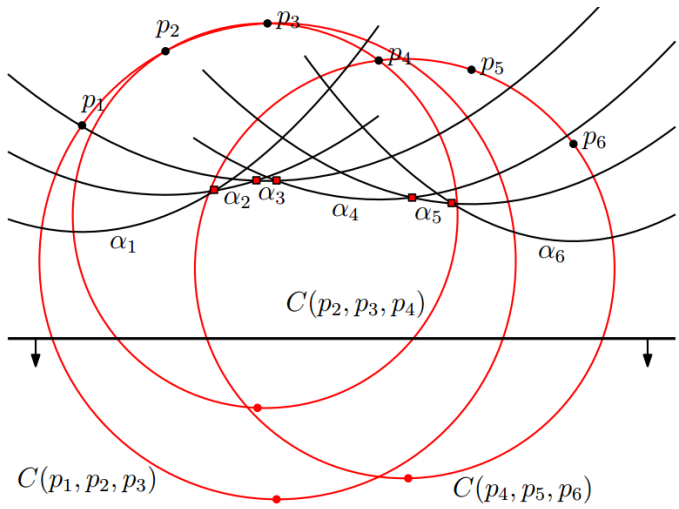
Na narednoj slici: levo: sedišta P_i i P_j i P_k čiji se lukovi pojavljuju redom na frontu; sredina: opisani krug nalazi se iznad brišuće linije, opisani krug je prazan i njegovo središte je na jednakim rastojanjima od P_i i P_j, P_k i l (ovo središte je teme Voronoj dijagrama); desno: parabolički luk P_j nestaje iz fronta.

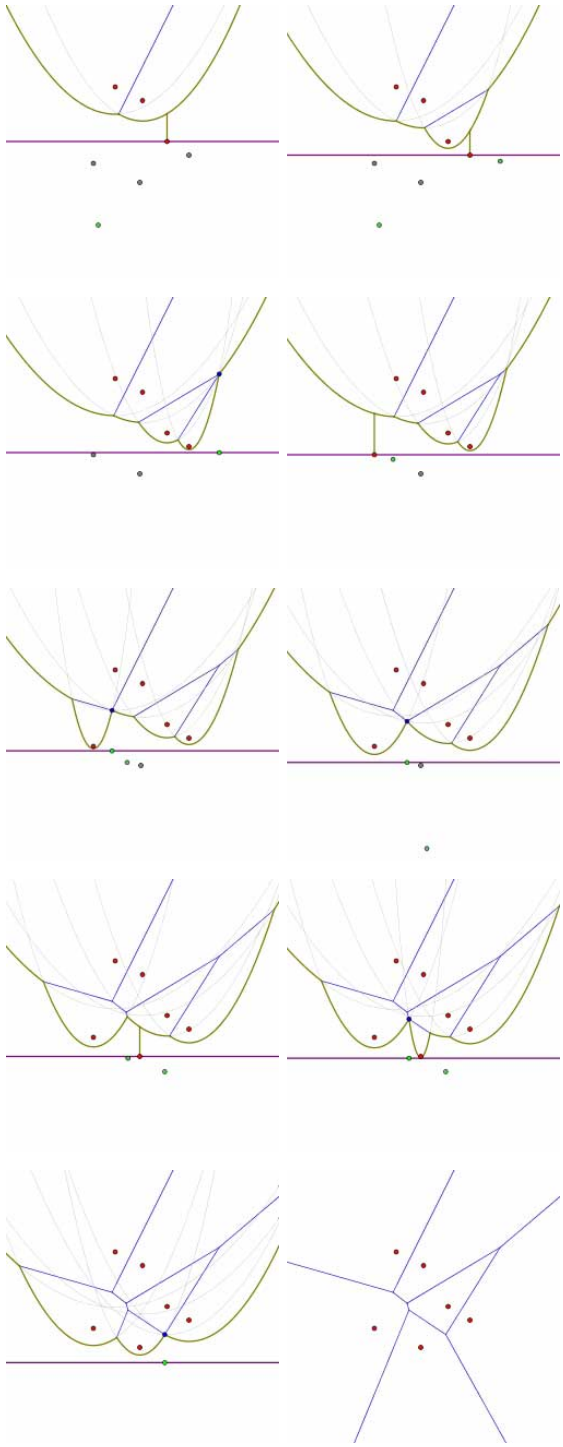


Ključno geometrijsko svojstvo koje ukazuje na događaj drugog tipa je da važi

$$d(Q, L) = d(Q, P_i) = d(Q, P_j) = d(Q, P_k)$$

Tada je prava l tangenta na krug opisan oko P_i, P_j, P_k.





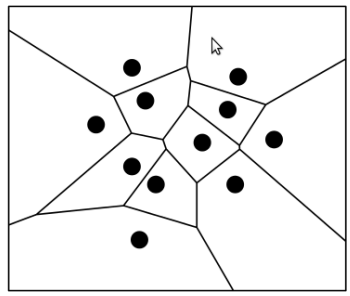
10.3.1 Implementacija

U slučaju događaja sedišta nova grana počinje da se formira, a u slučaju događaja kruga, dve rastuće grane formiraju novo teme. Potrebno je izabrati pogodne strukture podataka uz koje će opisani algoritam biti efikasan.

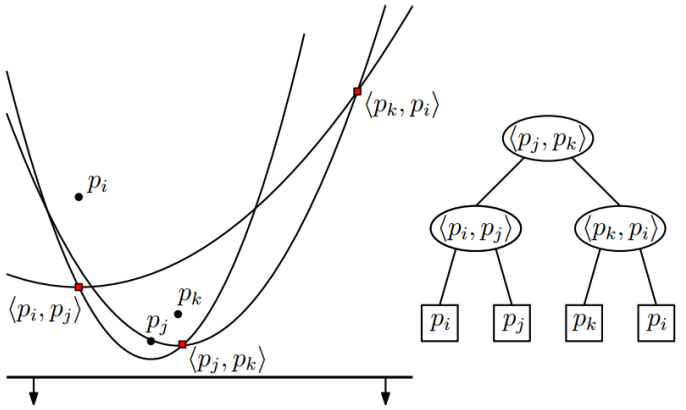
Biće održavan status brišuće prave, tj. struktura koja opisuje front: u slevanadesno poretku, skup sedišta koje određuju front (naglasimo da algoritam nikada ne čuva liniju fronta, već samo listu sedišta koje je određuju). Front je određen poretkom sleva nadesno sedišta koja formiraju front. Ova sedišta čuvaju se u strukturi $\mathcal{T}$ koja se implementira kao balansirano binarno stablo. Neka parabola može da učestvuje sa više lukova u frontu, pa se sedišta mogu pojavljivati više puta u listi.

Koriste se sledeće strukture podataka:

- Voronoj dijagram će biti reprezentovan u vidu DCEL. Problem je, međutim, u tome što sve grane dijagrama nisu duži, već su neke poluprave. To nije problem u toku rada algoritma, ali na kraju je potrebno kreirati ispravnu listu. To se postiže tako što se kreira okvir dovoljno veliki da sadrži sva temena dijagrama, te će sve grane biti duži.



- Status ili front je reprezentovan u vidu balansiranog binarnog stabla pretrage $\mathcal{T}$. Njegovi listovi odgovaraju lukovima fronta (a čuvaju zapravo sedišta tih lukova). Najlevlji list odgovara najlevljem luku, sledeći list sledećem luku itd. Svaki list sadrži sedište koje mu odgovara. Unutrašnji čvorovi stabla odgovaraju tačkama preseka susednih lukova. Svaki presek izračunava se po potrebi (u konstantnom vremenu). Kada se traži parabolički luk koji odgovara novom događaju sedišta – potrebno je izračunati $O(\log n)$ preseka.



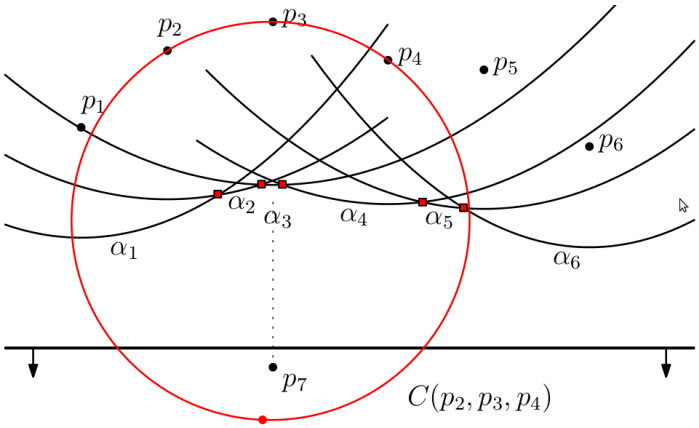
Sa ovakvom reprezentacijom fronta, u vremenu $O(\log n)$ može se pronaći luk iznad novog sedišta na brišućoj pravoj. U unutrašnjem čvoru, poredimo x koordinatu novog sedišta sa x koordinatom presečne tačke. Kao što je ranije rečeno, lukovi i parabole se nikad ne čuvaju.

U stablu $\mathcal{T}$ čuvaju se i pokazivači na druge dve strukture koje se koriste tokom rada algoritma. Svaki list stabla, koji reprezentuje neki luk α, čuva pokazivač na element reda događaja, element koji reprezentuje događaj kruga u kojem će luk α nestati. Ovaj pokazivač je NULL ako ne postoji događaj kruga u kojem će luk α nestati ili ako ovaj događaj još uvek nije otkriven. Konačno, svaki unutrašnji čvor ν sadrži pokazivač na granu u strukturi DCEL, i to na onu granu koju obilazi presečna tačka koju opisuje čvor ν.

- Red događaja $\mathcal{Q}$ se implementira kao red sa prioritetima, gde je prioritet događaja njegova y koordinata. $\mathcal{Q}$ čuva događaje koji su već poznati. Za događaj sedišta, čuva se samo sedište. Za događaj kruga, kao tačka događaja čuva se najniža tačka kruga, sa pokazivačem na list u stablu $\mathcal{T}$ koji odgovara luku koji će nestati.

Svi događaju sedišta poznati su unapred, ali događaji kruga nisu. Njih je potrebno pametno obraditi. Tokom pomeranja brišuće prave, front menja svoja topološka svostva prilikom svakog događaja. Događaj može proizvesti nove trojke uzastopnih lukova i može učiniti da neke trojke nestanu. Za svaka tri susedna luka na frontu potencijalni događaj kruga biće pohranjen u listu $\mathcal{Q}$. U ovome postoje dve stvari koje zahtevaju dodatnu pažnju. Prvo, mogu da postoje trojke čije se presečne tačke lukova ne približavaju (na primer, za tačke P, Q, R presečne tačke lukova se približavaju ako je presek simetrala duži PQ i QR *ispod* tačke Q. U ovom slučaju, trojka ne definiše potencijalni događaj kruga. Drugo, čak i ako trojka određuje preseke koji se približavaju, odgovarajući događaj kruga ne mora nužno da se desi: može da se desi da ta trojka nestane pre (na primer, ako se novo sedište pojavi na frontu). Takva

situacija naziva se *lažna uzbuna* i može se desiti zbog novog događaja kruga ili zbog novog događaja sedišta koji će se desiti u međuvremenu:



Zato se radi sledeće. U svakom događaju, proveravaju se sve trojke lukova koje nastaju. Na primer, u događaju sedišta, mogu se dobiti tri nove trojke: trojka u kojoj je novi luk levi element trojke, srednji element trojke i desni element trojke. Ako takva trojka ima približavajuće preseke, onda se odgovarajući događaj umeće u $\mathcal{Q}$. Napomenimo da u slučaju događaja sedišta, trojka u kojoj je novi luk srednji element trojke ne može kreirati novi događaj.

Ukupan broj nastajanja parabola je n i svaki troši $O(\log n)$ vremena.

Algoritam: Algoritam za izračunavanje Voronoj dijagrama

Ulaz: Skup sedišta u ravni $P = \{P_1, P_2, \ldots, P_n\}$

Izlaz: Voronoj dijagram $Vor(P)$ dat unutar okvira u vidu DCEL $\mathcal{D}$.

1: Inicijalizuj red događaja $\mathcal{Q}$ svim događajima sedišta, inicijalizuj praznu strukturu statusa $\mathcal{T}$ i prazan DCEL $\mathcal{D}$.
2: **dok god** red $\mathcal{Q}$ nije prazan
3: Uzmi događaj sa najvećom y koordinatom iz $\mathcal{Q}$.
4: **ako je** Ako je to događaj sedišta koji se javlja u sedištu P_i **onda**
5: obradi taj događaj
6: **inače**
7: obradi događaj kruga za parabolički luk γ, gde je γ list strukture $\mathcal{T}$
8: Unutrašnji čvorovi još uvek prisutni u $\mathcal{T}$ odgovaraju polupravama koje su grane Voronoj dijagrama. Izračunaj okvir kojem pripadaju sva temena Voronoj dijagrama i pridruži neograničene grane okviru ažuriranjem $\mathcal{D}$ na odgovarajući način.

Algoritam: Obrađivanje događaja sedišta

1: **ako je** stablo $\mathcal{T}$ je prazno **onda**
2: ubaci P_i u njega (tako da se $\mathcal{T}$ sastoji samo od lista koji sadrži P_i.
3: **inače**
4: Nađi u $\mathcal{T}$ luk α koji je vertikalno iznad P_i (može se uraditi u vremenu $O(\log n)$. Ako list koji reprezentuje α sadrži pokazivač na događaj kruga u $\mathcal{Q}$, onda je taj događaj lažna uzbuna i treba da bude izbrisan iz $\mathcal{Q}$.
5: Zameni list stabla $\mathcal{T}$ koji reprezentuje luk α podstablom koje ima tri lista: srednji čuva novo sedište P_i a preostala dva čuvaju sedište koje je prethodno bilo pridruženo luku α. Sačuvaj parove P_j, P_i' i P_i', P_j koji opisuju nove tačke preseka u nova dva unutrašnja čvora. Primeni operaciju balansiranja stabla $\mathcal{T}$ ako je potrebno.
6: Kreiraj novi zapis za granu u Voronoj dijagramu za granu između temena P_i and P_j, i duž koje će ići dve nove presečne tačke.
7: Proveri trojku uzastopnih lukova gde je novi luk za P_i levi luk kako bi se utvrdilo da li se presečne tačke primiču. Ako da, umetni događaj kruga u $\mathcal{Q}$ i dodaj pokazivače između čvora u $\mathcal{T}$ i čvora $\mathcal{Q}$. Uradi isto za trojku u kojoj je novi luk desni luk.

Algoritam: Obrađivanje događaja kruga

1: Obriši list γ koji reprezentuje luk α koji će nestati iz $\mathcal{T}$. Ažuriraj parove koji reprezentuju presečne tačke u unutrašnjim čvorovima. Primeni operaciju balansiranja stabla $\mathcal{T}$ ako je potrebno. Obriši iz $\mathcal{Q}$ sve događaje kruga koji uključuju α; oni mogu biti pronađeni korišćenjem pokazivača od prethodnika i sledbenika od γ u $\mathcal{T}$.
2: Dodaj središte kruga koji je proizveo događaj kruga kao teme u $\mathcal{D}$. Dovrši i kreiraj u strukturi $\mathcal{D}$ dve grane koje se završavaju u središtu kruga. Kreiraj novu menjajuću granu iz središta kruga.
3: Proveri novu trojku uzastopnih lukova koja je imala ranijeg levog suseda luka α kao svoj srednji luk, radi provere da li se dve presečne tačke primiču međusobno. Ako da, umetni odgovarajući događaj kruga u $\mathcal{Q}$ i postavi pokazivače između novog događaja kruga u $\mathcal{Q}$ i odgovarajućeg lista u $\mathcal{T}$. Uradi isto za trojkou u kojoj je raniji desni sused srednji luk.

Specijalni (degenerisani) slučajevi zahtevaju dodatnu pažnju. Na primer, ako se događaj sedišta javlja u odnosu na dva sedišta sa istom y koordinatom, onda se bira jedan od dva luka i uvodi se jedan presečni luk dužine nula. Može da se pokaže da je takav postupak ispravan.

10.3.2 Složenost

Oba tipa događaja mogu se obraditi u vremenu $O(\log n)$

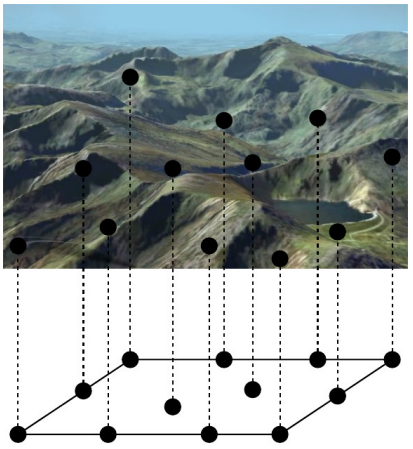
Teorema 10.5. *Algoritam za izračunavanje Voronoj dijagrama (implementiran na opisani način) ima vremensku složenost $O(n \log n)$ i prostornu složenost $O(n)$.*

Glava 11

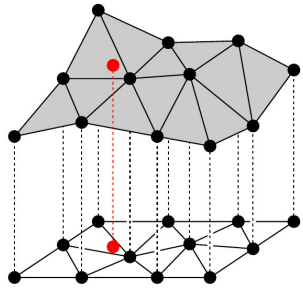
Delone triangulacija

Triangulacija ima brojne primene u mnogim oblastima nauke i tehnike. Mnoge skupove tačaka moguće je triangulisati na više načina. Jedan od njih naziva se Delone triangulacija, ona je jedinstvena i ima mnoga zanimljiva svojstva. Delone triangulacija može da se uopšti na prostore dimenzije veće od dva.

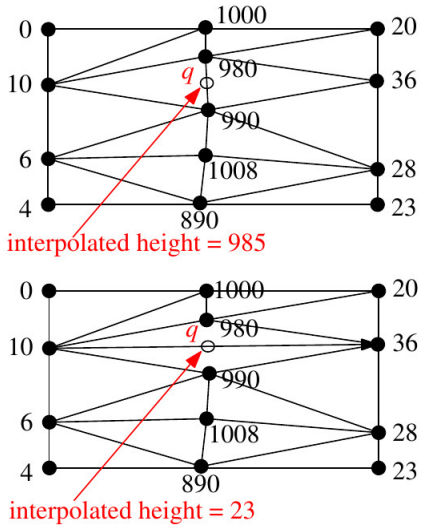
Razmotrimo sledeći primer. Teren je opisan funkcijom $f : A \subset R^2 \to R$. Poznate su vrednosti za nadmorske visine (ili temperaturu ili vazdušni pritisak) samo za neke tačke u kojima je izvršeno merenje. Pitanje je kako proceniti (interpolirati) visine u svim preostalim tačkama.



Jedno moguće rešenje je: koristeći triangulaciju, kao što je to ilustrovano na sledećoj slici:



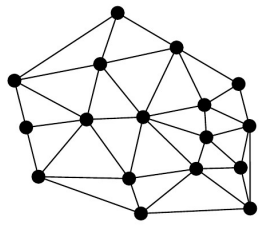
Triangulacija za skup tačaka može da bude mnogo i postavlja se pitanje koju izabrati. Na primer, u mnogim primenama prednost može imati triangulacija u kojoj trouglovi nisu „izduženi“ (već su bliži pravilnim trouglovima), kao što je ilustrovani sledećim primerom.



11.1 Triangulacija

Definicija 11.1 (Triangulacija)**.** *Neka je* $\mathcal{P} = \{P_1, P_2, \ldots, P_n\}$ *skup tačaka u ravni.* Triangulacija *skupa* $\mathcal{P}$ *je maksimalno ravansko razlaganje sa skupom temena* $\mathcal{P}$*.*

Razlaganje je maksimalno ako se dodavanjem stranice koja spaja bilo koja dva temena skupa $\mathcal{P}$ narušava planarnost razlaganja.



Teorema 11.1. *Triangulacija skupa $\mathcal{P}$ od n tačaka ima $2n-k-2$ trouglova i $3n-k-3$ ivica, gde je k broj temena u konveksnom omotaču skupa $\mathcal{P}$.*

Dokaz: Triangulacija određuje povezani planarni graf za koji važi $V-E+F=2$. Broj temena V jednak je n, a broj strana F jednak je traženom broju trouglova t u triangulaciji uvećanom za jedan: $F=t+1$ (zbog spoljne, neograničene oblasti u grafu).

Svaki od t trouglova ima tri ivice, a spoljna, neograničena oblast ima k ivica. Svaka ivica pripada tačno dvema oblastima, pa za ukupan broj ivica E važi $2E=3t+k$. Iz Ojlerove formule sada dobijamo:

$$n-(3t+k)/2+(t+1)=n-3t/2-k/2+t+1=n-t/2-k/2+1=2,$$

odakle je $t=2n-k-2$. Odatle neposredno sledi da je E jednako $3n-k-3$. □

11.1.1 Vektor uglova triangulacije

Definicija 11.2 (Vektor uglova triangulacije)**.** *Neka je $\mathcal{T}$ triangulacija skupa koja ima m trouglova (i $3m$ unutrašnjih uglova).* Vektor uglova *triangulacije $\mathcal{T}$ je $A(\mathcal{T})=(\alpha_1,\alpha_2,\ldots,\alpha_{3m})$ gde su α_1, α_2, …, α_{3m} unutrašnji uglovi trouglova triangulacije $\mathcal{T}$ sortirani neopadajuće.*

Definicija 11.3 (Poredak nad vektorima uglova)**.** *Neka su $\mathcal{T}=(\alpha_1,\alpha_2,\ldots,\alpha_{3m})$ i $\mathcal{T}'=(\alpha'_1,\alpha'_2,\ldots,\alpha'_{3m})$ dve triangulacije skupa tačaka $\mathcal{P}$ od m tačaka.*

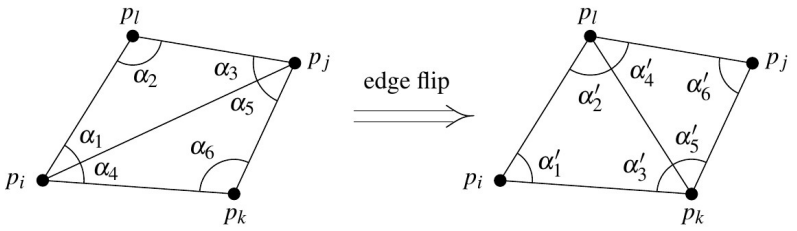
Kažemo da je vektor $A(\mathcal{T})$ veći *od vektora $A(\mathcal{T}')$ i pišemo $A(\mathcal{T})>A(\mathcal{T}')$ ako je $A(\mathcal{T})$ leksikografski veće od $A(\mathcal{T}')$, tj. ako postoji indeks i, $1\le i\le 3m$ takav da važi $\alpha_j=\alpha'_j$ za sve $j<i$ i $\alpha_i>\alpha'_i$.*

Triangulacija $A(\mathcal{T})$ skupa tačaka $\mathcal{P}$ je ugaono optimalna *ako važi $A(\mathcal{T})\ge A(\mathcal{T}')$ za svaku triangulaciju $A(\mathcal{T}')$.*

Ukoliko u nekoj primeni nije dobro da u triangulaciji ima „izduženih“ trouglova, onda od dve triangulacije ima prednost ona koja je veća, a najpoželjnija je ugaono optimalna triangulacija. Ugaono optimalna triangulacija ne samo da maksimizuje najmanji ugao u triangulaciji nego i više od toga.

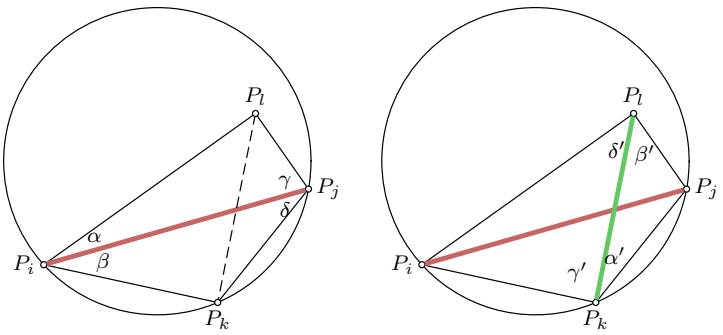
11.1.2 Obrtanje stranica

Obrtanjem jedne ivice (*edge flip*) menja se šest unutrašnjih uglova: vektor $(\alpha_1, \alpha_2, \ldots, \alpha_6)$ se zamenjuje sa $(\alpha'_1, \alpha'_2, \ldots, \alpha'_6)$.



Definicija 11.4 (Nelegalna stranica)**.** *Stranica $e = P_iP_j$ je „nelegalna“ ako je vrednost $min(\alpha_1, \alpha_2, \ldots, \alpha_6)$ manja od vrednosti $min(\alpha'_1, \alpha'_2, \ldots, \alpha'_6)$.*

Nelegalna stranica je „loša“, u smislu da se njenim obrtanjem (strogo) povećava vektor uglova.



Lema 11.1. *Stranica P_iP_j je nelegalna ako i samo ako P_l pripada unutrašnjosti kruga C opisanog oko $P_iP_kP_j$.*

Dokaz: Pretpostavimo da P_l pripada unutrašnjosti kruga C. Ugao $P_iP_lP_j$ nije minimalni unutrašnji ugao jer $\angle P_iP_lP_j > \angle P_kP_lP_j > \beta$. Analogno, ugao $P_iP_kP_j$ nije minimalni unutrašnji ugao jer $\angle P_iP_kP_j > \angle P_iP_kP_l > \gamma$. Dakle, minimalni ugao je jedan od α, β, γ, δ. Ako bi se stranica P_iP_j obrnula, onda bi unutrašnji uglovi bili α', β', γ', δ', $\angle P_kP_iP_l$ i $\angle P_kP_jP_l$. Minimum ovih uglova je sigurno veći nego za polaznu triangulaciju jer važi $\alpha < \alpha'$, $\beta < \beta'$, $\gamma < \gamma'$, $\delta < \delta'$ (jer za jedan krug, veći je onaj periferijski ugao koji je nad dužim lukom).

Primetimo da tačka P_l pripada unutrašnjosti kruga C ako i samo ako je zbir unutrašnjih uglova kod P_l i P_k, veći od 180°, pa samim tim veći i od zbira unutrašnjih uglova kod P_i i P_j.

Pretpostavimo da je stranica P_iP_j nelegalna. Potrebno je dokazati da onda tačka P_l pripada unutrašnjosti kruga C opisanog oko $P_iP_kP_j$, tj. da je zbir unutrašnjih uglova kod P_l i P_k, veći od zbira unutrašnjih uglova kod P_i i P_j. Pretpostavimo suprotno — da je zbir unutrašnjih uglova kod P_i i P_j veći od zbira unutrašnjih uglova kod P_l i P_k. Tada P_i pripada unutrašnjosti kruga opisanog oko trougla $P_kP_jP_l$, pa na osnovu prvog smera važi da je stranica P_kP_l nelegalna. Ali to je kontradikcija, jer ne mogu i stranica P_iP_j i stranica P_kP_l da budu nelegalne. □

Ako konveksni četvorougao $P_jP_kP_jP_l$ nije tetivan onda je tačno jedna od njegovih dijagonala nelegalna stranica.

Definicija 11.5 (Legalna triangulacija)**.** *Triangulacija je legalna ako ne sadrži nijednu nelegalnu stranicu.*

Na osnovu naredne teoreme može se kreirati algoritam koji vraća legalnu triangulaciju skupa tačaka.

Algoritam: LegalTriangulation($\mathcal{T}$)

Ulaz: Proizvoljna triangulacija $\mathcal{T}$ skupa $\mathcal{P}$.

Izlaz: Legalna triangulacija skupa $\mathcal{P}$.

1: **dok god** $\mathcal{T}$ sadrži nelegalnu stranicu P_iP_j
2: // obrni stranicu P_iP_j
3: Neka su $P_iP_jP_k$ i $P_iP_jP_l$ dva trougla sa stranicom P_iP_j
4: Izbaci P_iP_j iz $\mathcal{T}$, i u $\mathcal{T}$ dodaj stranicu P_kP_l.
5: vrati $\mathcal{T}$

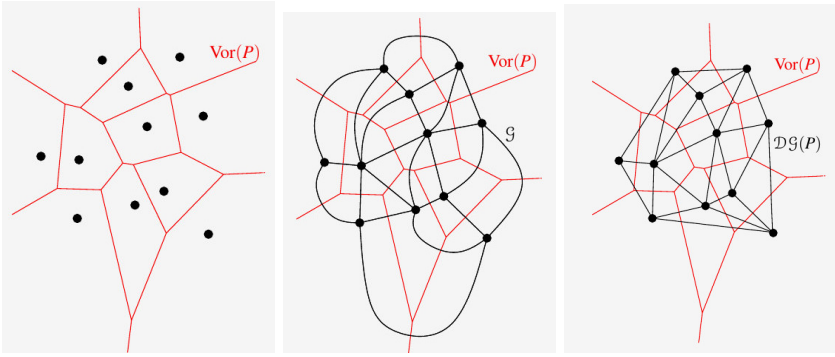
Navedeni algoritam se zaustavlja. Zaista, za fiksan skup tačaka, postoji konačan broj mogućih triangulacija, pa kako se eliminisanjem nelegalne stranice

povećava vektor uglova, sledi da se algoritam zaustavlja. Kada se algoritam zaustavi, dobijena je legalna triangulacija. Dakle, navedeni algoritam je korektan i daje legalnu triangulaciju, ali je suviše neefikasan za praktične primene.

Primetimo da trivijalno važi naredno tvrđenje.

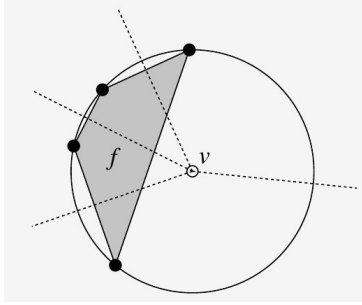
Lema 11.2. *Ako je triangulacija ugaono optimalna, onda je ona legalna.*

11.1.3 Voronoj dijagram i Delone triangulacija



Neka je $\mathcal{P}$ skup od n tačaka u ravni. Neka je $Vor(\mathcal{P})$ Voronoj dijagram skupa $\mathcal{P}$ i neka je G dualni graf razlaganja $Vor(\mathcal{P})$: čvorovi grafa su sedišta Voronoj dijagrama, a granama grafa spojene su susedne ćelije Voronoj razlaganja. Delone graf $DG(\mathcal{P})$ je pravolinijsko utapanje grafa G. Narednu teoremu navodimo bez dokaza.

Teorema 11.2. *Delone graf je planaran (tj. njegove grane se ne seku).*



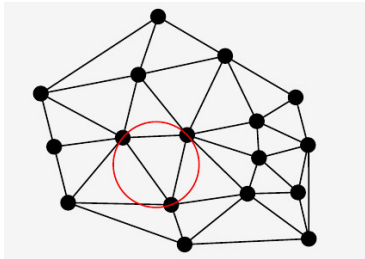
Ako za Voronoj dijagram $Vor(\mathcal{P})$ postoji k ćelija koje sadrže teme v, onda odgovarajuća strana Delone grafa $DG(\mathcal{P})$ ima k stranica i, štaviše, ta strana grafa $DG(\mathcal{P})$ je konveksni tetivni poligon.

Razmotrimo slučaj da nikoje četiri tačke iz zadatog skupa ne pripadaju jednom krugu i zovimo taj slučaj „opšti slučaj“. U tom, „opštem“ slučaju, Delone graf čini triangulaciju.

Teorema 10.3 može biti formulisana u terminima Delone grafova:

Teorema 11.3. *Neka je $\mathcal{P}$ skup tačaka u ravni.*

- *Tri tačke P_i, P_j i P_k skupa $\mathcal{P}$ su temena iste oblasti Delone grafa skupa $\mathcal{P}$ akko unutrašnjost kruga kroz P_i, P_j i P_k ne sadrži nijednu tačku skupa $\mathcal{P}$.*
- *Dve tačke P_i i P_j skupa $\mathcal{P}$ čine granu Delone grafa za $\mathcal{P}$ akko postoji krug koji prolazi kroz P_i i P_j čija unutrašnjost ne sadrži nijednu tačku skupa $\mathcal{P}$.*



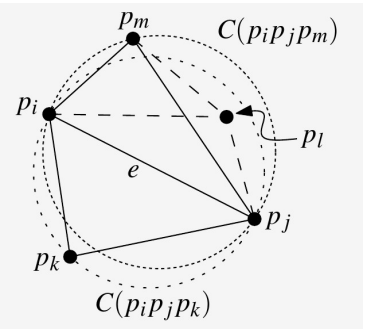
Iz navedene teoreme neposredno sledi naredna karakterizacija Delone triangulacije.

Teorema 11.4. *Neka je $\mathcal{T}$ triangulacija skupa tačaka $\mathcal{P}$ u ravni. $\mathcal{T}$ je Delone triangulacija za $\mathcal{P}$ akko unutrašnjost nijednog opisanog kruga oko trougla iz $\mathcal{T}$ ne sadrži neku tačku iz $\mathcal{P}$.*

Teorema 11.5. *Triangulacija $\mathcal{T}$ skupa tačaka u ravni je legalna triangulacija akko je Delone triangulaija.*

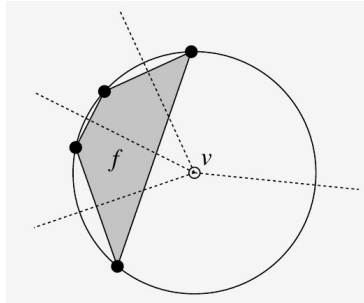
Dokaz: Jednostavno sledi da je svaka Delone triangulacija legalna.

Dokažimo pobijanjem da je svaka legalna triangulacija istovremeno i Delone triangulacija. Pretpostavimo da je $\mathcal{T}$ legalna triangulacija skupa tačaka $\mathcal{P}$ i da ona nije Delone triangulacija. Na osnovu teoreme 11.3, ovo znači da postoji trougao takav da njegov opisani krug sadrži tačku P_l iz skupa $\mathcal{P}$ u svojoj unutrašnjosti. Neka je e P_iP_j stranica tog trougla $P_iP_jP_l$ takva da su P_k i P_l sa njenih raznih strana. Od svih takvih parova $(P_iP_jP_k, P_l)$ iz $\mathcal{T}$ izaberimo onaj koji maksimizuje ugao $\angle P_iP_lP_j$.



Razmotrimo trougao triangulacije $P_iP_jP_m$ koji sadrži stranicu e. Kako on pripada triangulaciji, u njegovoj unutrašnjosti nema drugih tačaka iz skupa $\mathcal{P}$. Kako je $\mathcal{T}$ legalna triangulacija, stranica e je legalna, pa na osnovu leme 11.1, tačka P_m ne pripada unutrašnjosti kruga $C(P_iP_jP_k)$. Odatle sledi $\angle P_iP_mP_j < \angle P_iP_lP_j$, pa tačka P_l pripada unutrašnjosti kruga $C(P_iP_jP_m)$. Tačke P_l i P_m su sa iste strane prave P_iP_j, pa važi da ili P_lP_i seče duž P_mP_j ili da duž P_lP_j seče duž P_mP_i. Pretpostavimo da duž P_lP_i seče duž P_mP_j (kontradikcija se analogno dobija i u drugom slučaju), tj. da su P_l i P_i sa raznih strana prave P_jP_m. Tada važi da je $\angle P_jP_lP_m$ veći od ugla $\angle P_iP_lP_j$, što je suprotno pretpostavci da je ugao $\angle P_iP_lP_j$ maksimalan takav ugao. □

Ako skup tačaka P nije u opštem položaju, neke strane Delone grafa su n-touglovi sa više od tri stranice. Međutim, oni su konveksni i tetivni (oko njih se može opisati krug) i mogu se trivijalno triangulisati. Štaviše, npr. u slučaju četvorougla svejedno je koja će dijagonala biti izabrana da ga triangluiše (jer su dobijeni uglovi u oba slučaja jednaki). I za poligone sa više uglova može se pokazati da je najmanji ugao za bilo koju triangulaciju isti.



Ugaono optimalna triangulacija je legalna triangulacija (lema 11.2), tj Delone triangulacija. U opštem slučaju (kada nikoje četiri tačke ne pripadaju krugu), Delone graf je jedinstven, pa je jedinstvena i triangulacija koju on čini. Dakle, u opštem položaju, legalna triangulacija je jedinstvena, pa onda ona

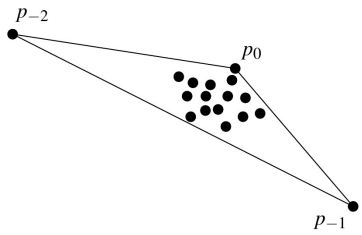
mora bude ugaono optimalna. Ukoliko skup tačaka P nije u opštem položaju, Delone triangulacija nije jedinstvena, ali, kao što je gore objašnjeno, svaka ima jednake uglove. Iz navedenog razmatranja sledi naredna teorema.

Teorema 11.6. *Ugaono optimalna triangulacija skupa tačaka u ravni je Delone triangulacija i ona maksimizuje minimalni ugao nad svim mogućim triangulacijama tog skupa.*

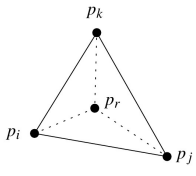
11.2 Izračunavanje Delone triangulacije

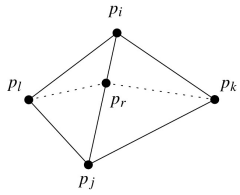
Na osnovu veze između Voronoj dijagrama i Delone grafa, ukoliko je raspoloživ algoritam za izračunavanje Voronoj dijagrama, lako se može dobiti Delone triangulacija. U nastavku je opisan jedan drugačiji način za izračunavanje Delone triangulacije.

Prvi korak je dodavanje dve tačke P_{-1} i P_{-2} tako da sve date tačke pripadaju jednom trouglu. Na kraju će te dve tačke biti izbačene:



Algoritam je randomizovano inkrementalan, što znači da se tačke datog skupa obrađuju jedna po jedna, u slučajnom poretku. Potrebno je najpre odrediti trougao u tekućoj triangulaciji kojem pripada nova tačka P_r i onda dodati nove grane u razlaganje ravni, moguća su dva slučaja:





Algoritam: DelaunayTriangulation($\mathcal{P}$)

Ulaz: Skup $\mathcal{P}$ $n+1$ tačaka u ravni.

Izlaz: Delone triangulacija za $\mathcal{P}$

1: Neka je P_0 tačka iz $\mathcal{P}$ sa najvećom y koordinatom, a ako ima više takvih - ona među njima koja ima najveću x koordinatu.

2: Neka su P_{-1} i P_{-2} dve tačke ravni dovoljno deleke da su sve tačke iz $\mathcal{P}$ sadržane u trouglu $P_0P_{-1}P_{-2}$.

3: Inicijalizuj $\mathcal{T}$ kao triangulaciju koja se sastoji od jednog trougla $P_0P_{-1}P_{-2}$.

4: Izaberi slučajnu permutaciju P_1, P_2, ... P_n skupa tačaka $\mathcal{P} \setminus \{P_0\}$.

5: **za** r od 1 do n

6: Odrediti trougao $P_iP_jP_k$ iz $\mathcal{T}$ koji sadrži P_r.

7: **ako je** P_r pripada unutrašnjosti trougla $P_iP_jP_k$ **onda**

8: Dodaj stranice od P_r do P_i, P_j, P_k, podelivši time trougao $P_iP_jP_k$ na tri trougla.

9: LegalizeEdge(P_r , P_iP_j , $\mathcal{T}$)

10: LegalizeEdge(P_r , P_jP_k , $\mathcal{T}$)

11: LegalizeEdge(P_r, pk pi , $\mathcal{T}$)

12: **inače**

13: // P_r pripada stranici P_iPj trougla $P_iP_jP_k$

14: Dodaj stranicu od P_r do P_k i do trećeg temena P_l drugog trougla kojem pripada stranica P_iP_j, podelivši time dva trougla sa stranicom P_iP_j na četiri trougla.

15: LegalizeEdge(P_r , P_iP_l , $\mathcal{T}$)

16: LegalizeEdge(P_r , P_lP_j , $\mathcal{T}$)

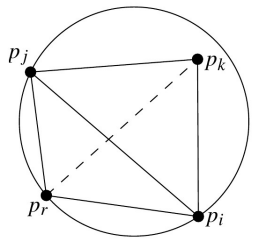
17: LegalizeEdge(P_r , P_jP_k , $\mathcal{T}$)

18: LegalizeEdge(P_r , P_kP_i , $\mathcal{T}$)

19: Izbaci iz $\mathcal{T}$ tačke P_{-1} i P_{-2} sa svim stranicama kojima pripadaju.

20: Vrati $\mathcal{T}$

Zbog dodavanja novih grana, triangulacija možda više nije legalna, i treba da se „legalizuje“.

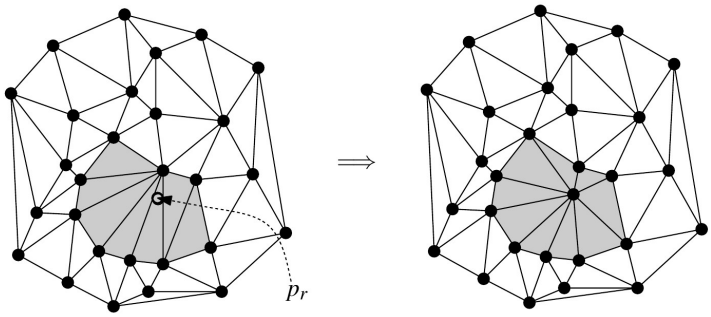


Algoritam: LegalizeEdge(P_r, P_iP_j , $\mathcal{T}$)

1: // Tačka koja je dodata je P_r a P_iP_j je stranica triangulacije $\mathcal{T}$ koja možda treba da bude obrnuta

2: **ako je** P_iP_j je nelegalna **onda**

3: Neka je $P_iP_jP_k$ (drugi) trougao koji sadrži stranicu P_iP_j
4: // Obrni P_iP_j :
5: Zameni P_iP_j sa P_rP_k
6: LegalizeEdge(P_r, P_iP_k, $\mathcal{T}$)
7: LegalizeEdge(P_r, P_kP_j, $\mathcal{T}$)



Može se dokazati da je navedeni algoritam korektan. Bez dokaza navodimo i narednu teoremu.

Teorema 11.7. *Delone triangulacija skupa n tačaka u ravni može se izračunati u očekivanom vremenu $O(n \log n)$ i u prostoru očekivane veličine $O(n)$*

Glava 12

Lociranje tačke

Slajovi preuzeti sa strane: http://www.cs.uu.nl/docs/vakken/ga/2022/

Point location

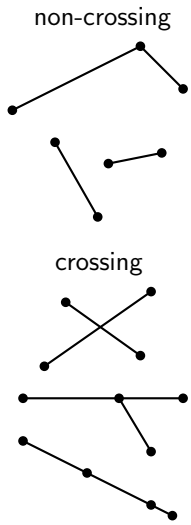
Point location problem: Preprocess a planar subdivision such that for any query point q, the face of the subdivision containing q can be given quickly (name of the face)

- From GPS coordinates, find the region on a map where you are located
- Subroutine for many other geometric problems (Chapter 13: motion planning, or shortest path computation)

Point location

Planar subdivision: Partition of the plane by a set of non-crossing line segments into vertices, edges, and faces

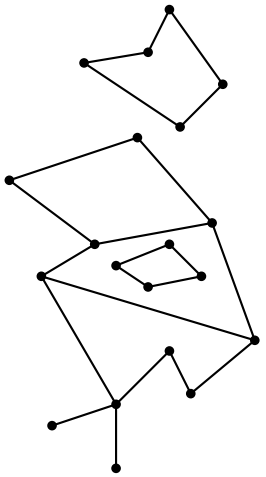
non-crossing: disjoint, or at most a shared endpoint



Point location

Data structuring question, so interest in query time, storage requirements, and preprocessing time

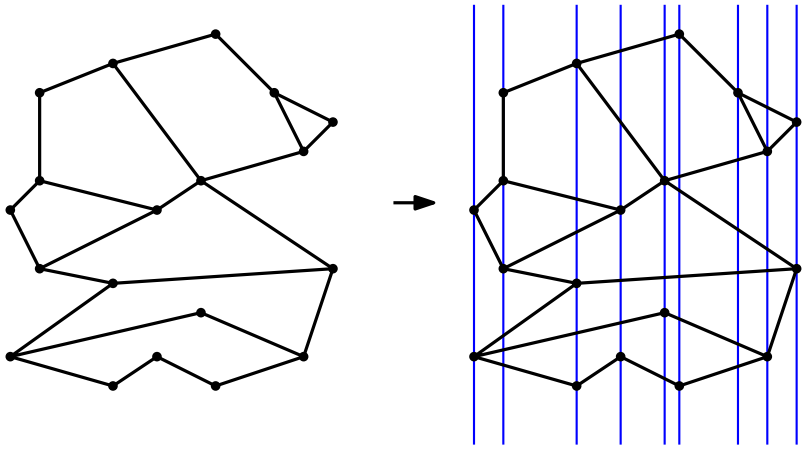
To store: set of n non-crossing line segments and the subdivision they induce



First solution

Idea: Draw vertical lines through all vertices, then do something for every vertical strip that appears

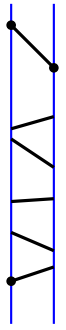
First solution



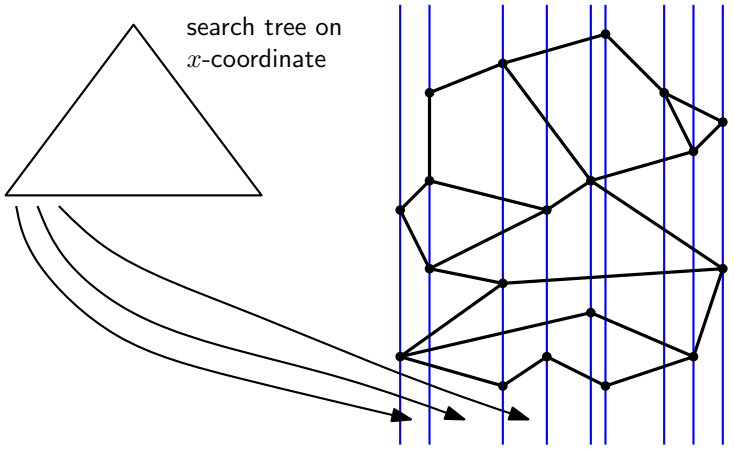
In one strip

Inside a single strip, there is a well-defined bottom-to-top order on the line segments

Use this for a balanced binary search tree that is valid if the query point is in this strip (knowing between which edges we are is knowing in which face we are)



Solution with strips

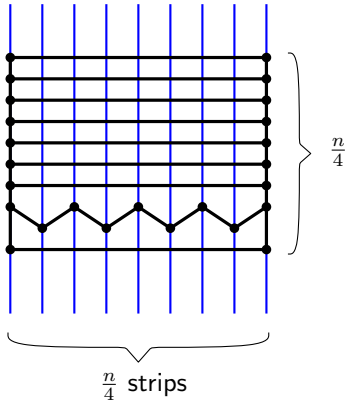


Solution with strips

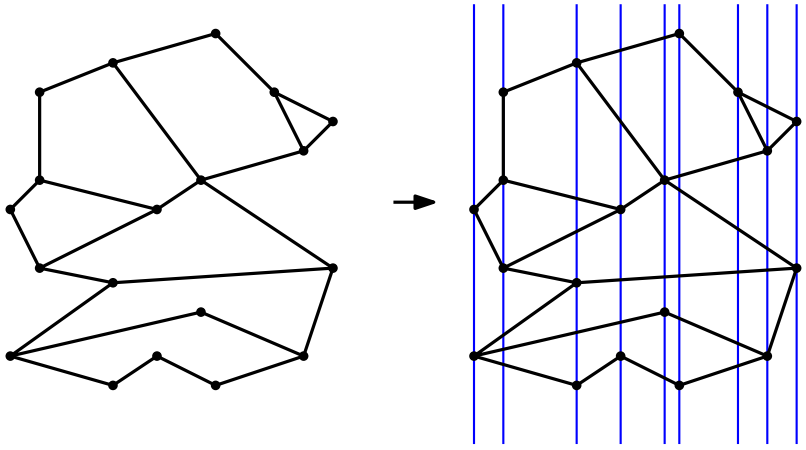
To answer a query with $q = (q_x, q_y)$, search in the main tree with q_x to find a leaf, then follow the pointer to search in the tree that is correct for the strip that contains q_x

Question: What are the storage requirements and what is the query time of this structure?

Solution with strips



Solution with strips



Different idea

The vertical strips idea gave a *refinement* of the original subdivision, but the number of faces went up from linear in n to quadratic in n

Is there a different refinement whose size remains linear, but in which we can still do point location queries easily?

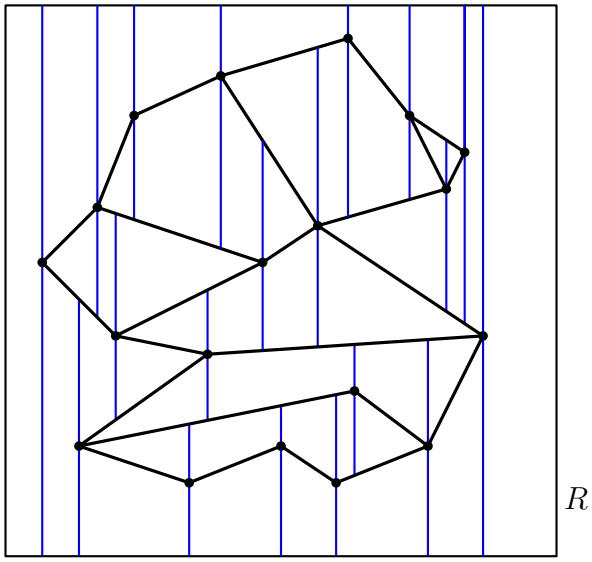
Vertical decomposition

Suppose we draw vertical extensions from every vertex up and down, *but only until the next line segment*

- Assume the input line segments are not vertical
- Assume every vertex has a distinct x-coordinate
- Assume we have a bounding box R that encloses all line segments that define the subdivision

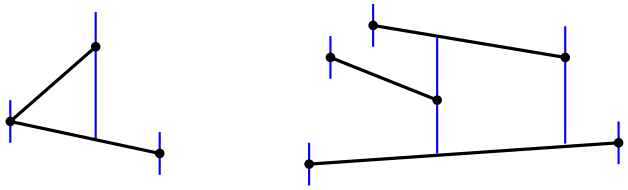
This is called the vertical decomposition or trapezoidal decomposition

Vertical decomposition



Vertical decomposition faces

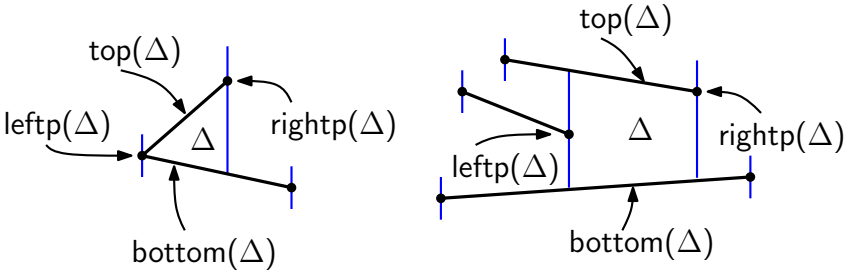
The vertical decomposition has triangular and trapezoidal faces



Vertical decomposition faces

Every face has a vertical left and/or right side that is a vertical extension, and is bounded from above and below by some line segment of the input

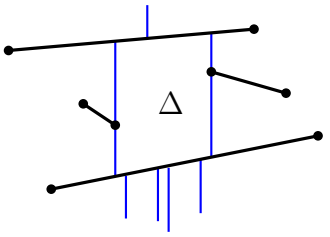
The left and right sides are defined by some endpoint of a line segment



Vertical decomposition faces

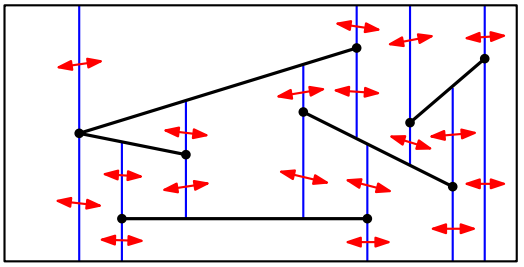
Every face is defined by no more than four line segments

For any face, we ignore vertical extensions that end on top(Δ) and bottom(Δ)



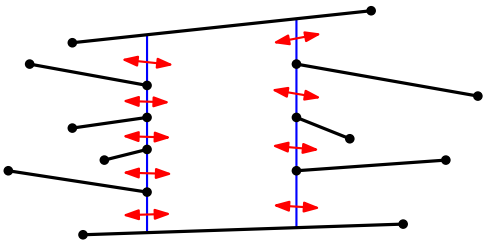
Neighbors of faces

Two trapezoids (including triangles) are *neighbors* if they share a vertical side



Each trapezoid has 1, 2, 3, or 4 neighbors

Neighbors of faces



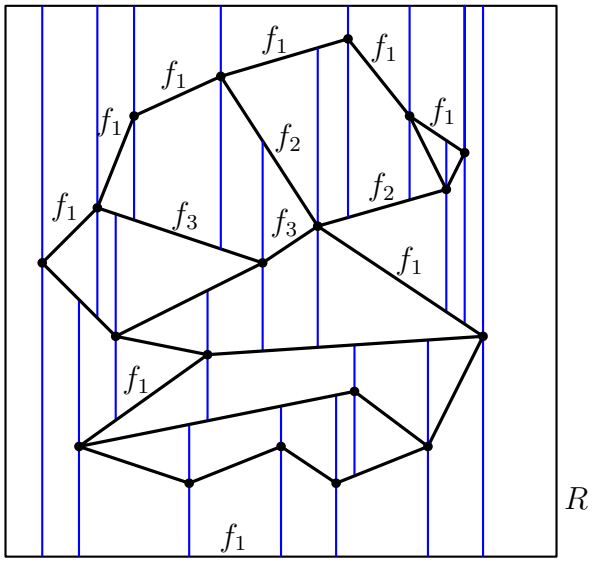
A trapezoid could have many neighbors if vertices had the same x-coordinate

Representation

We could use a DCEL to represent a vertical decomposition, but we use a more direct & convenient structure

- Every face Δ is an object; it has fields for top(Δ), bottom(Δ), leftp(Δ), and rightp(Δ) (two line segments and two vertices)
- Every face has fields to access its up to four neighbors
- Every line segment is an object and has fields for its endpoints (vertices) and the name of the face in the original subdivision directly above it
- Each vertex stores its coordinates

Representation

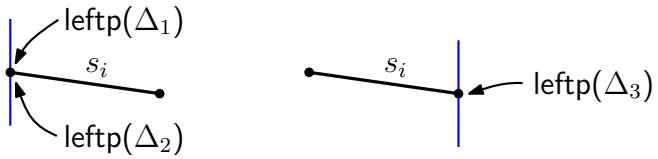


Representation

Any trapezoid Δ can find out the name of the face it is part of via bottom(Δ) and the stored name of the face

Complexity

A vertical decomposition of n non-crossing line segments inside a bounding box R, seen as a proper planar subdivision, has at most $6n+4$ vertices and at most $3n+1$ trapezoids



Point location preprocessing

The input to planar point location is a planar subdivision, for example in DCEL format

First, store with each edge the name of the face above it (our structure will find the edge below any query point)

Then extract the edges to define a set S of non-crossing line segments; ignore the DCEL otherwise

Point location solution

We will use *randomized incremental construction* to build, for a set S of non-crossing line segments,

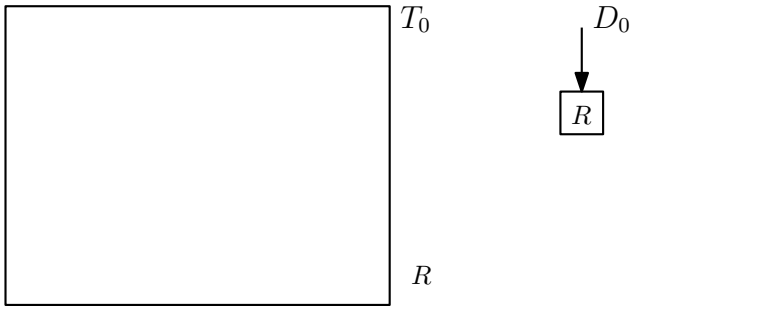
- a vertical decomposition T of S and R
- a search structure D whose leaves correspond to the trapezoids of T

The simple idea: Start with R, then add the line segments in random order and maintain T and D

Point location solution

Let $s_1, \ldots, s_n$ be the n line segments in random order

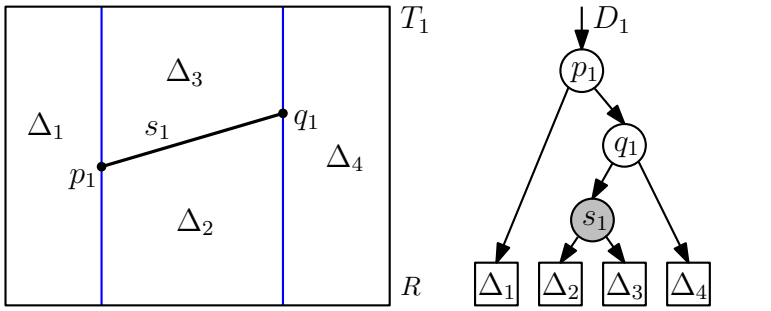
Let T_i be the vertical decomposition of R and $s_1, \ldots, s_i$, and let D_i be the search structure obtained by inserting $s_1, \ldots, s_i$ in this order



Point location solution

Let $s_1, \ldots, s_n$ be the n line segments in random order

Let T_i be the vertical decomposition of R and $s_1, \ldots, s_i$, and let D_i be the search structure obtained by inserting $s_1, \ldots, s_i$ in this order



Point location solution

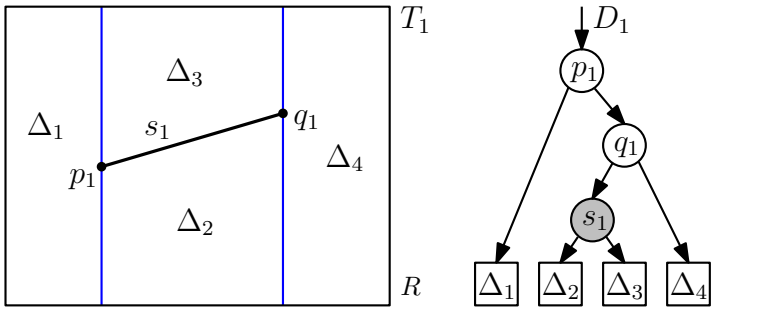
The search structure D has x-nodes, which store an endpoint, and y-nodes, which store a line segment s_j

For any query point t, we only test at an x-node: Is t left or right of the vertical line through the stored point?

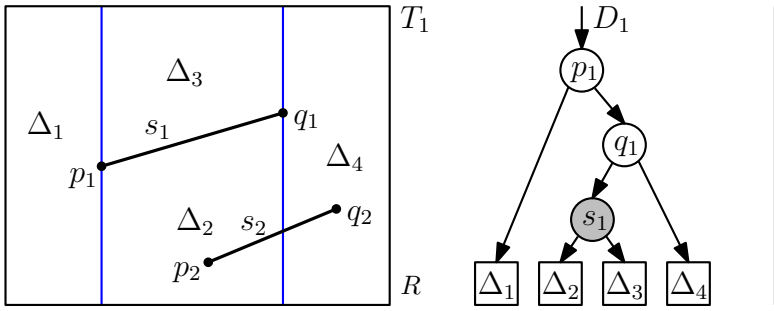
For any query point t, we only test at an y-node: Is t below or above the stored line segment?

We will guarantee that the question at a y-node is only asked if the query point t is between the vertical lines through p_j and q_j, if line segment $s_j = \overline{p_j q_j}$ is stored

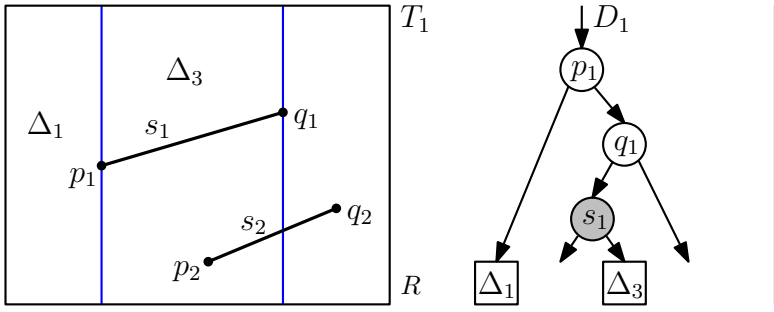
Point location solution



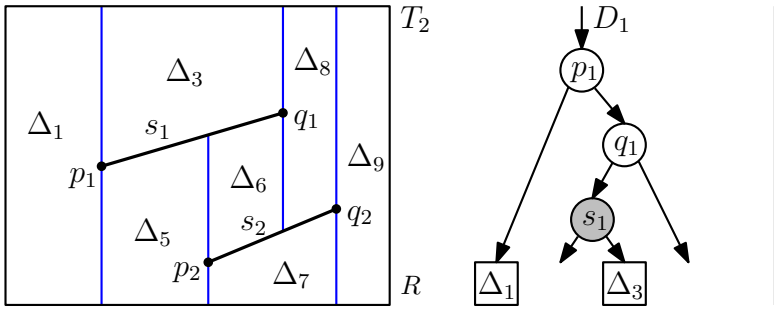
Point location solution



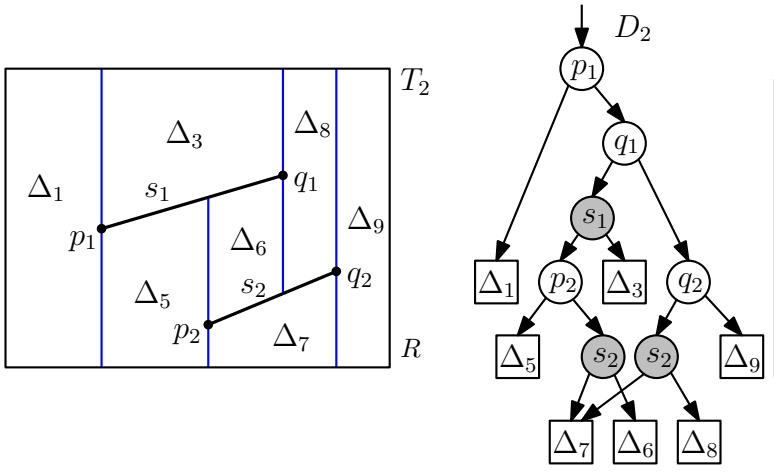
Point location solution



Point location solution



Point location solution

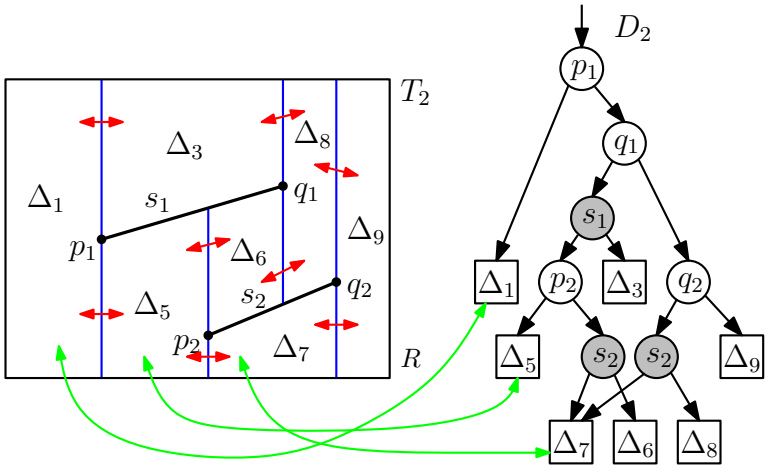


Point location solution

We want only one leaf in D to correspond to each trapezoid; this means we get a search *graph* instead of a search *tree*

It is a directed acyclic graph, or DAG, hence the name D

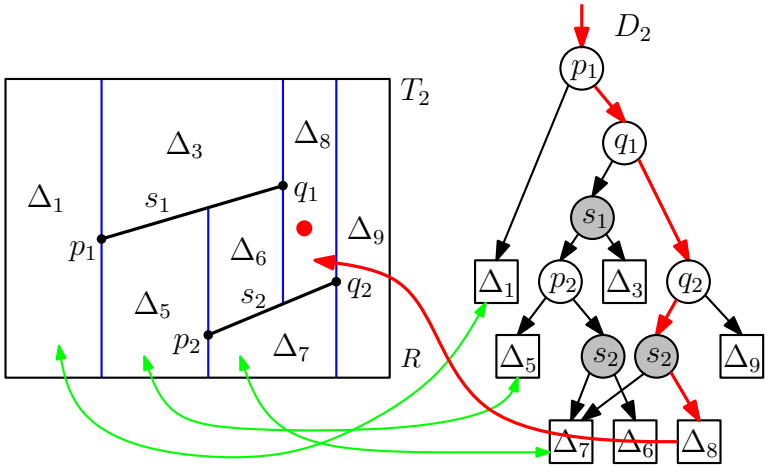
Point location solution



Point location query

A point location query is done by following a path in the search structure D to a leaf, then following its pointer to a trapezoid of T, then accessing bottom(..) of this trapezoid, and reporting the name of the face stored with it

Point location query



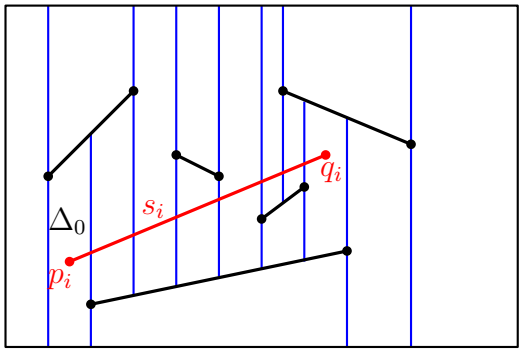
The incremental step

Suppose we have D_{i-1} and T_{i-1}, how do we add s_i?

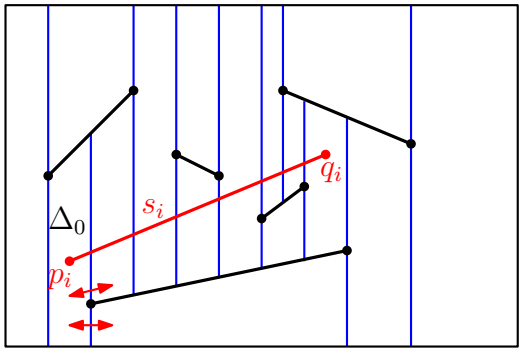
Because D_{i-1} is a *valid point location structure* for $s_1, \ldots, s_{i-1}$, we can use it to find the trapezoid of T_{i-1} that contains p_i, the left endpoint of s_i

Then we use T_{i-1} to find all other trapezoids that intersect s_i

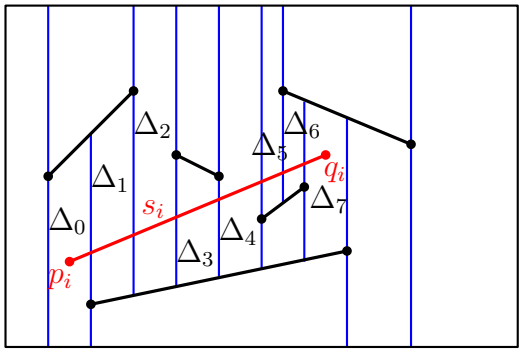
Find intersected trapezoids



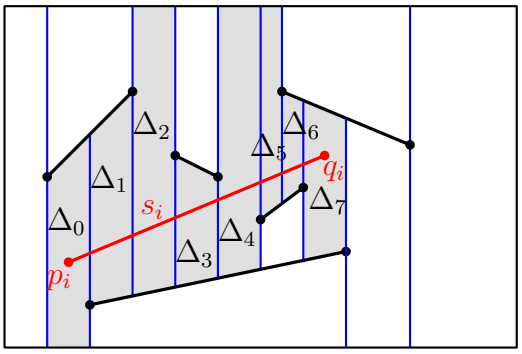
Find intersected trapezoids



Find intersected trapezoids



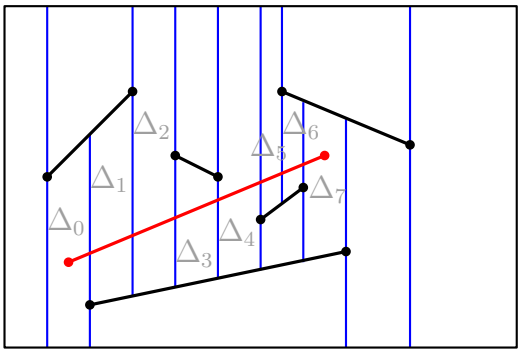
Find intersected trapezoids



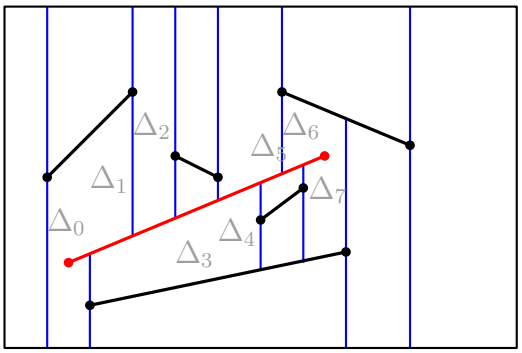
Find intersected trapezoids

After locating the trapezoid that contains p_i, we can determine all k trapezoids that intersect s_i in $O(k)$ time by traversing T_{i-1}

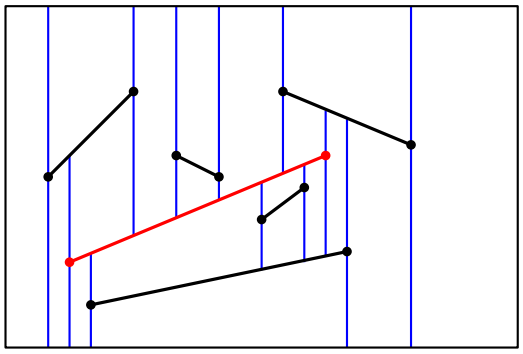
Updating the vertical decomposition



Updating the vertical decomposition



Updating the vertical decomposition



Updating the vertical decomposition

We can update the vertical decomposition in $O(k)$ time as well

Updating the search structure

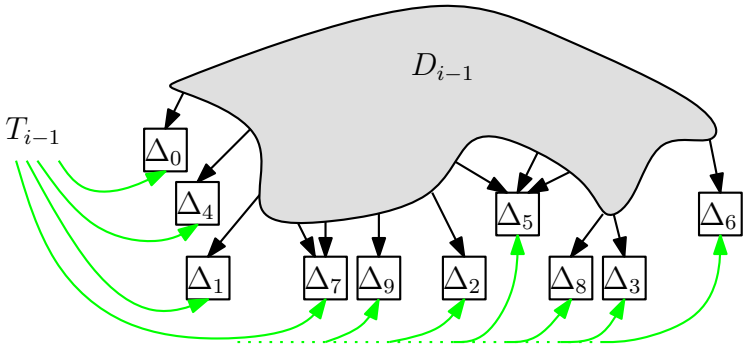
The search structure has k leaves that are no longer valid as leaves; they become internal nodes

We find these using the pointers from T_{i-1} to D_{i-1}

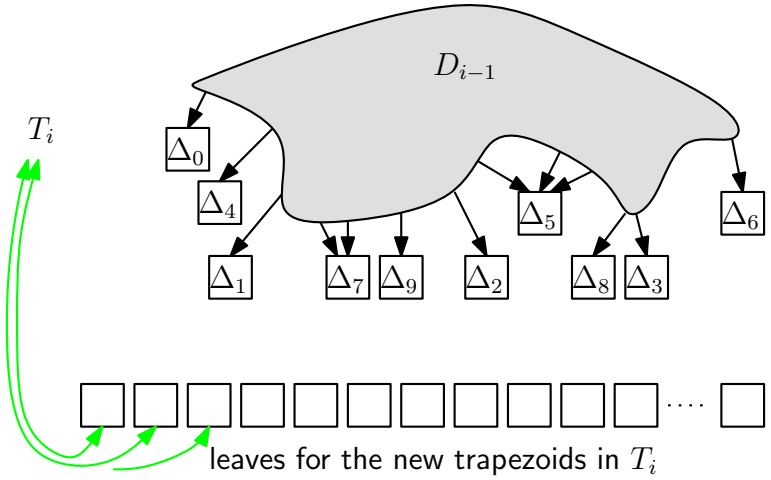
From the update of the vertical decomposition T_{i-1} into T_i we know what new leaves we must make for D_i

All new nodes besides the leaves are x-nodes with p_i and q_i and y-nodes with s_i

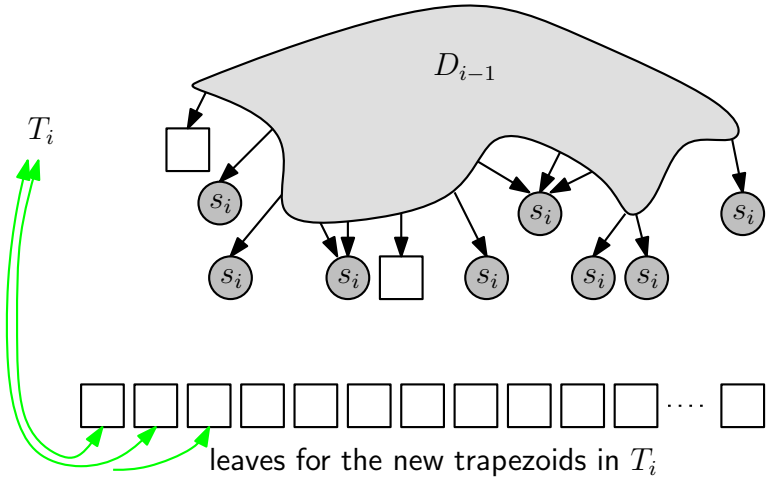
Updating the search structure



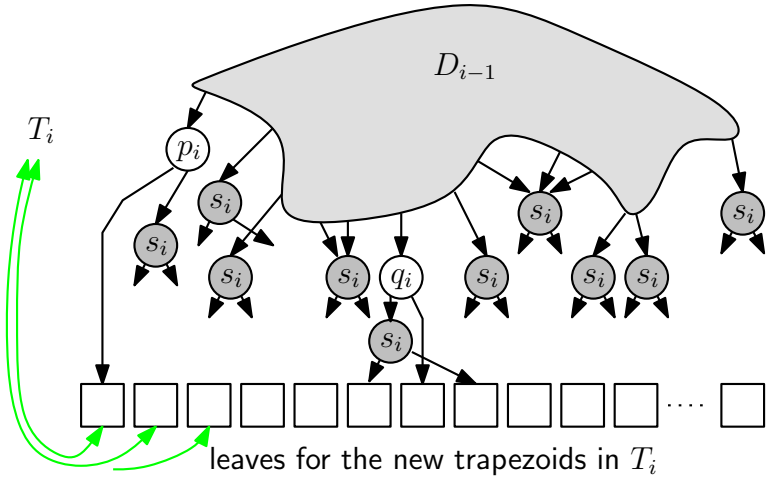
Updating the search structure



Updating the search structure



Updating the search structure



Observations

For a single update step, adding s_i and updating T_{i-1} and D_{i-1}, we observe:

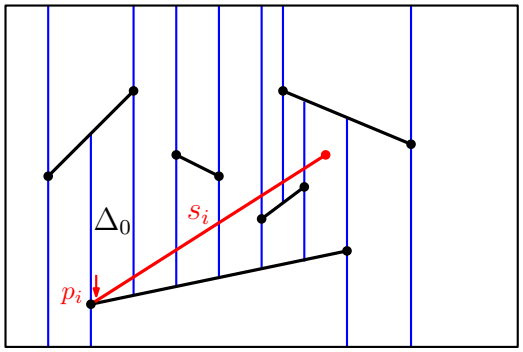
- If s_i intersects k_i trapezoids of T_{i-1}, then we will create $O(k_i)$ new trapezoids in T_i
- We find the k_i trapezoids in time linear in the search path of p_i in D_{i-1}, plus $O(k_i)$ time
- We update by replacing k_i leaves by $O(k_i)$ new internal nodes and $O(k_i)$ new leaves
- The maximum depth increase is three nodes

Questions

Question: In what case is the depth increase three nodes?

Question: We noticed that the directed acyclic graph D is binary in its out-degree, what is the maximum in-degree?

A common but special update



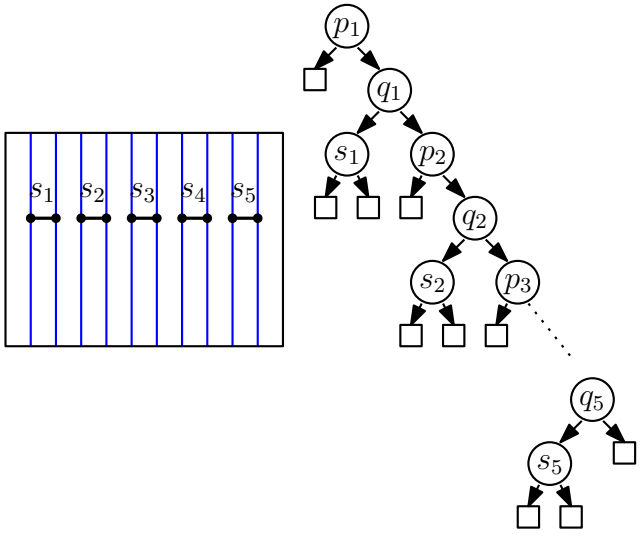
If p_i was already an existing vertex, we search in D_{i-1} with a point a fraction to the right of p_i on s_i

Randomized incremental construction

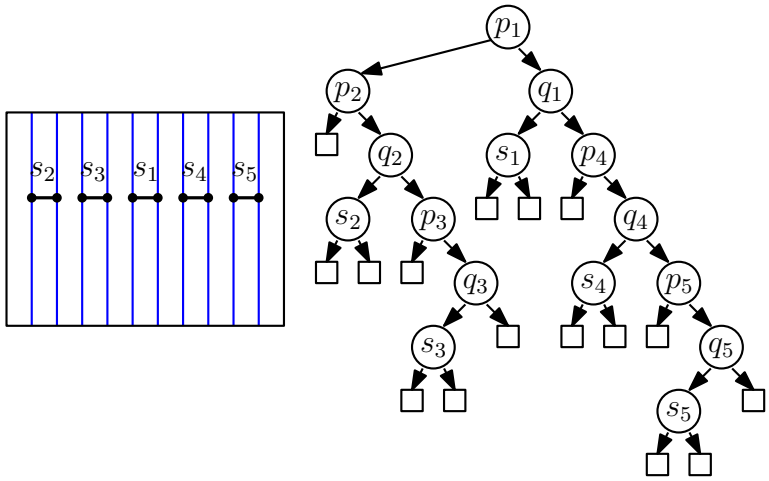
Randomized incremental construction, where does it matter?

- The vertical decomposition T_i is independent of the insertion order among $s_1, \ldots, s_i$
- The search structure D_i can be different for many orders of $s_1, \ldots, s_i$
- The *number of nodes in* D_i depends on the order
- The *depth of search paths in* D_i depends on the order

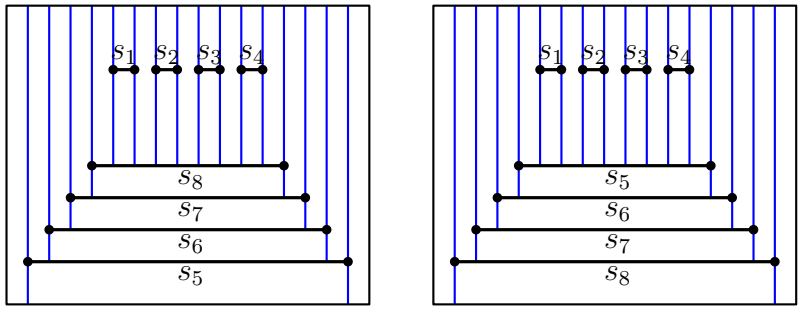
Randomized incremental construction



Randomized incremental construction



Randomized incremental construction



Storage of the structure

The vertical decomposition structure T always uses linear storage

The search structure D can use anything between linear and quadratic storage

We analyse the expected number of new nodes when adding s_i, using *backwards analysis* (of course)

Result

Theorem: Given a planar subdivision defined by a set of n non-crossing line segments in the plane, we can preprocess it for planar point location queries in $O(n \log n)$ expected time, the structure uses $O(n)$ expected storage, and the expected query time is $O(\log n)$

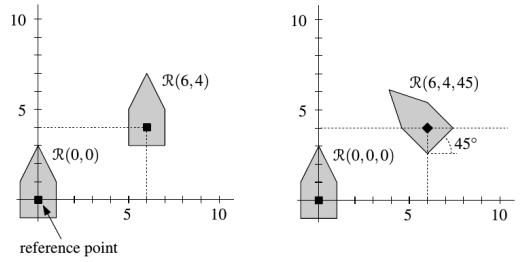
Glava 13

Planiranje kretanja robota

Planiranje kretanja važno je u mnogim kontekstima, a kretanje robota je jedan od njih. Da bi mogao da se autonomno da se kreće, robot mora da poseduje znanje o svom okruženju: treba, na primer, da ima informacije o preprekama ili da ima senzore pomoću kojih može da ih detektuje itd. Prilikom kretanja robot treba da se kreće bez sudaranja. Razmatraćemo pojednostavljen problem – kretanje u ravni. Razmatraćemo uglavnom samo robote koji ne menjaju svoju orijentaciju tokom kretanja, tj. kojima je na raspolaganju samo translaciono kretanje. Smatraćemo da je robot prost poligon ili, specijalno, samo jedna tačka.

13.1 Radni prostor i konfiguracioni prostor

Neka je $\mathcal{R}$ robot koji se kreće u Dekartovoj ravni, ili u *radnom prostoru* u kojem postoji skup prepreka $S = \{P_1, \ldots, P_t\}$. Ukoliko se robot kreće samo translaciono, svaki položaj robota može se opisati vektorom translacije. Neka $\mathcal{R}(x, y)$ označava robota $\mathcal{R}$ koji je transliran za vektor (x, y) u odnosu na koordinatni početak.



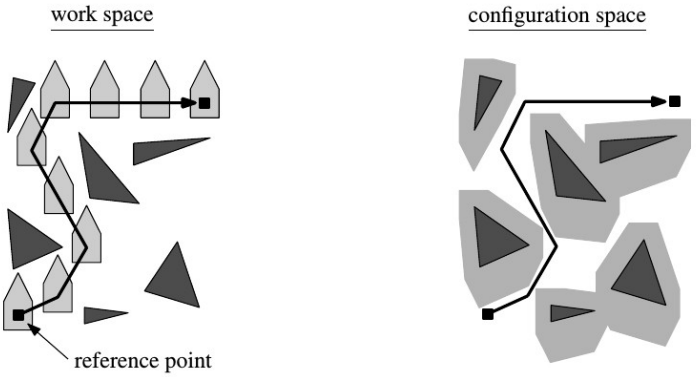
Opisana notacija može se objasniti i u terminima referentne tačke. *Referentna tačka* je neka izabrana tačka unutrašnjosti robota (mada čak nije nužno

da bude tačka unutrašnjosti). U osnovnom položaju robota, referentna tačka ima koordinate $(0, 0)$. Položaj robota potpuno je određen položajem referentne tačke. Sa $\mathcal{R}(x, y)$ onda označavamo položaj robota u kojem se referentna tačka nalazi u tački (x, y).

Ukoliko robot može da menja svoju orijentaciju rotacijama (na primer, oko svoje referentne tačke), onda je potreban još jedan parametar – φ – da opiše položaj robota. Tada sa $\mathcal{R}(x, y, \varphi)$ označavamo robot kojem se referentna tačka nalazi u tački (x, y) i koji je rotiran za ugao φ.

Prostor parametara koji opisuju položaj robota naziva se konfiguracioni prostor i označav sa $\mathcal{C}(\mathcal{R})$. U slučaju robota, u zavisnosti da li može da se rotira, taj prostor je dvodimenzioni ili trodimenzioni.

Podskup konfiguracionog prostora za čije elemente robot seče neku prepreku iz skupa $\mathcal{S}$ naziva se *zabranjeni prostor* i označava sa $\mathcal{C}_{forb}(\mathcal{R}, \mathcal{S})$. Ostatak konfiguracionog prostora naziva se *slobodan prostor* i označava sa $\mathcal{C}_{free}(\mathcal{R}, \mathcal{S})$. Svaka putanja kretanja robota bez sudaranja preslikava se u neku krivu u slobodnom prostoru. Smatra se da su prepreke otvoreni skupovi i da robot može da ih dodiruje tokom kretanja.



13.2 Tačkasti robot

Tačkasti robot je robot u vidu tačke, te su za njega radni prostor i konfiguracioni prostor jednaki. Pretpostavljamo da je radni prostor ograničen okvirom koji sadrži sve prepreke.

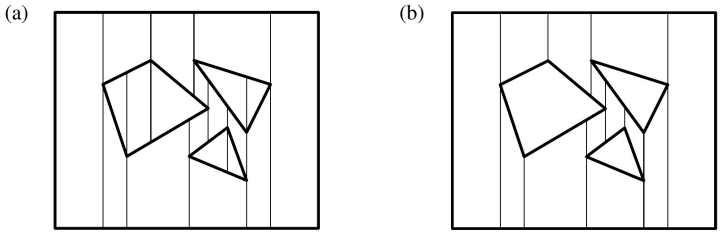
Neka je n ukupan broj stranica svih prepreka. Prvi korak u određivanju putanja za tačkasti robot je pogodno razlaganje radnog prostora. Za to se koristi trapezno razlaganje (videti glavu 12) i naredni algoritam:

Algoritam: ComputeFreeSpaces($\mathcal{S}$)

Ulaz: Skup $\mathcal{S}$ disjunktnih poligona

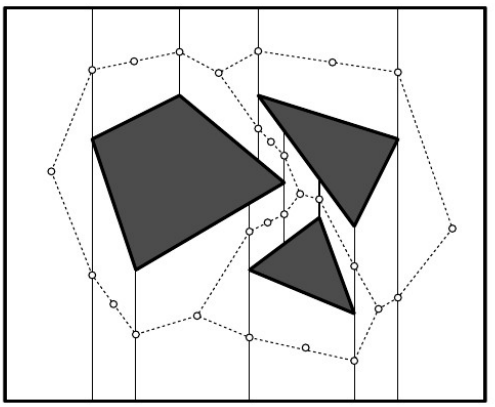
Izlaz: Trapezno razlaganje prostora $\mathcal{C}_{free}(\mathcal{R}, \mathcal{S})$ za tačkastog robota $\mathcal{R}$.

1: Neka je E skup stranica poligona iz skupa $\mathcal{S}$.
2: Izračunaj trapezno razlaganje $\mathcal{T}(E)$;
3: Ukloni trapeze koji pripadaju unutrašnjosti prepreka i vrati dobijeno razlaganje.



Lema 13.1. *Trapezno razlaganje slobodnog konfiguracionog prostora za tačkasti robot sa disjunktnim preprekama koje su poligoni sa ukupno n stranica, može se izračunati randomizovanim algoritmom sa očekivanim vremenom izvršavanja $O(n \log n)$.*

Sledeći korak je kreiranje *mape puteva*. To je graf koji je utopljen u slobodni prostor. Sa izuzetkom delova na početku i na kraju, traženi put za robota će uvek slediti grane mape puteve. Čvorovi mape puteva su određeni na sledeći način: čvorovi mape su središte svakog trapeza, kao i središta vertikalnih produžetaka. Mapa puta može da se konstruiše u vremenu $O(n)$ obilaskom DCEL strukture za slobodan prostor.



Algoritam: ComputePath($\mathcal{C}_{free}, \mathcal{G}_{road}, P_{start}, P_{goal}$)

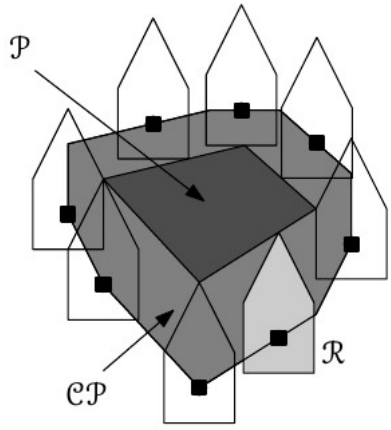
Ulaz: Trapezno razlaganje slobodnog prostora, mapa puta $\mathcal{G}_{road}$, polazna P_{start} i ciljna tačka P_{goal}

Izlaz: Putanja između P_{start} i P_{goal}, ako postoji. Ako ne postoji, to se prijavljuje.

1: Odredi trapez Δ_{start} koji sadrži tačku P_{start} i trapez Δ_{goal} koji sadrži tačku P_{goal}
2: **ako je** Δ_{start} ili Δ_{goal} ne postoje **onda**
3: Prijavi da je polazna ili ciljna tačka u zabranjenom prostoru.
4: **inače**
5: Neka je V_{start} središte trapeza Δ_{start} i neka je neka je V_{goal} središte trapeza Δ_{goalt}
6: Izračunaj put $\mathcal{G}_{road}$ od V_{start} to V_{goal} koristeći BFS.
7: **ako je** ne postoji takva putanja **onda**
8: Prijavi da ne postoji put od P_{start} do P_{goal}
9: **inače**
10: Vrati putanju koja se sastoji od duži koja spaja P_{start} i V_{start}, nađenu putanju iz $\mathcal{G}_{road}$ i duž od V_{goal} do P_{goal}.

13.3 Roboti koji nisu tačkasti i suma Minkovskog

Ukoliko robot nije tačkasti, problem pronalaženja puta je komplikovaniji.

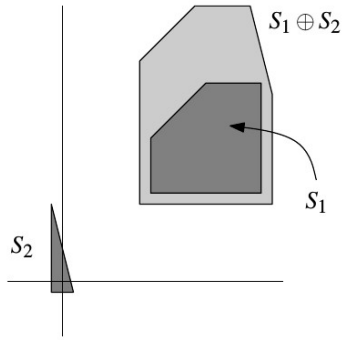


Prepreka konfiguracionog prostora za prepreku $\mathcal{P}$ i robota $\mathcal{R}$ definiše se kao skup tačaka $\mathcal{CP}$ u konfiguracionom prostoru takvih da odgovarajući položaj robota $\mathcal{R}$ seče $\mathcal{P}$, tj.

$$\mathcal{CP} = \{(x, y) \ : \ \mathcal{R}(x, y) \cap \mathcal{P} \neq 0\}.$$

Intuitivno, ova prepreka može se dobiti tako što se $\mathcal{R}$ vuče duž ruba oblasti $\mathcal{P}$.

Definicija 13.1 (Suma Minkovskog)**.** *Suma Minkovskog dva skupa $S_1 \subset \mathbf{R}^2$ i $S_2 \subset \mathbf{R}^2$, u oznaci $S_1 \oplus S_2$, definiše se na sledeći način: $S_1 \oplus S_2 = \{p + q : p \in S_1, q \in S_2\}$, gde $p + q$ označava vektorski zbir vektora p i q, tj. ako je $p = (px, py)$ i $q = (qx, qy)$, onda je $p + q = (px + qx, py + qy)$.*



Teorema 13.1. *Neka je $\mathcal{R}$ planarni robot koji se kreće translaciono i neka je $\mathcal{P}$ prepreka. Tada važi*

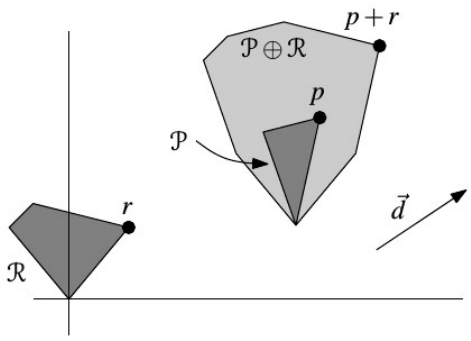
$$\mathcal{CP} = \mathcal{P} \oplus (-\mathcal{R}(0,0))$$

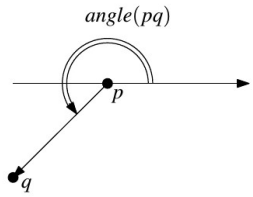
Dokaz: Treba da dokažemo da $\mathcal{R}(x, y)$ seče $\mathcal{P}$ ako i samo ako (x, y) pripada skupu $\mathcal{P} \oplus -\mathcal{R}(0, 0)$.

Pretpostavimo da $\mathcal{R}(x, y)$ seče $\mathcal{P}$ i neka je $Q = (qx, qy)$ tačka preseka. Iz $Q \in \mathcal{R}(x, y)$ sledi $(qx - x, qy - y) \in \mathcal{R}(0, 0)$, tj. $(-qx + x, -qy + y) \in -\mathcal{R}(0, 0)$. Kako važi i $Q \in \mathcal{P}$, sledi da (x, y) pripada skupu $\mathcal{P} \oplus -\mathcal{R}(0, 0)$.

Obratno, neka (x, y) pripada skupu $\mathcal{P} \oplus -\mathcal{R}(0, 0)$. Tada postoje tačke $(rx, ry) \in \mathcal{R}(0, 0)$ i $(px, py) \in \mathcal{P}$ takve da je $(x, y) = (px - rx, py - ry)$ tj. takve da je $px = rx + x$ i $py = ry + y$, što znači da $\mathcal{R}(x, y)$ seče $\mathcal{P}$. □

Teorema 13.2. *Neka su $\mathcal{P}$ i $\mathcal{R}$ dva konveksna poligona sa n i m stranica. Tada je suma Minkovskog $\mathcal{P} \oplus \mathcal{R}$ konveksni poligon sa najviše $n + m$ stranica. Ako $\mathcal{P}$ i $\mathcal{R}$ nemaju paralelnih stranica, onda je broj stranica tačno $n + m$.*





Algoritam: MinkowskiSum($\mathcal{P}$, $\mathcal{R}$)

Ulaz: Konveksni poligon $\mathcal{P}$ sa temenima $V_1, \ldots, V_n$, i konveksni poligon $\mathcal{R}$ sa temenima $W_1, \ldots, W_n$. Temena su poređana suprotno kretanju kazaljke na satu, a V_1 i W_1 su temea sa najmanjim y koordinatama (i, ako ima više takvih, najmanjim x koordinatama.

Izlaz: Suma Minkovskog $\mathcal{P} \oplus \mathcal{R}$

```
1: i := 1; j := 1;
2: V_{n+1} := V_1; V_{n+2} := V_2; W_{m+1} := W_1; W_{m+2} := W_2;
3: repeat
4:     Dodaj V_i + W_j kao teme u P ⊕ R.
5:     ako je je ugao (V_i V_{i+1}) manji od ugla W_j W_{j+1} onda
6:         i := (i + 1);
7:     inače
8:         ako je je ugao (V_i V_{i+1}) veći od ugla W_j W_{j+1} onda
9:             j := (j + 1)
10:        inače
11:            i := (i + 1); j := ( j + 1)
12: until i == n + 1 i j == m + 1
```

Teorema 13.3. *Suma Minkovskog dva konveksna poligona sa n i m temena može se izračunati u vremenu $O(n+m)$.*

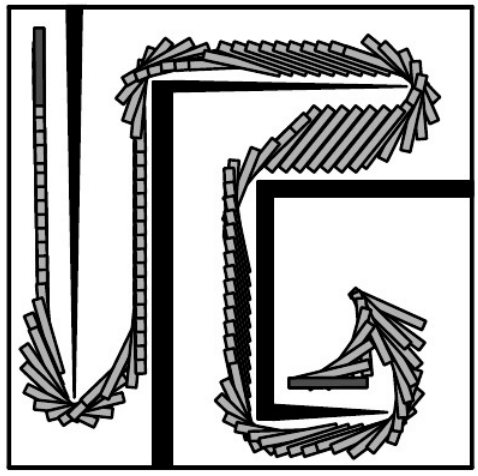
Kako se računa suma Minkovskog ako poligoini nisu konveksni? Nekonveksni poligoni se mogu razložiti na konveksne i suma se može izračunati na osnovu naredne jednakosti

$$S_1 \oplus (S_2 \cup S_3) = (S_1 \oplus S_2) \cup (S_1 \oplus S_3)$$

Kada se prepreke zamene proširenim preprekama (izračunatim kao sume Minkovskog), problem translacionog kretanja svodi se na problem kretanja tačkastog robota.

Teorema 13.4. *Neka je $\mathcal{R}$ konveksan robot sa translatornim kretanjem i $\mathcal{S}$ skup disjunktnih poligonskih prepreka sa ukupno n stranica. Tada se skup prepreka može obraditi za očekivano vreme $O(n \log^2 n)$, tako da se putanja između dve tačke, ako ona postoji, može naći u vremenu $O(n)$.*

13.4 Planiranje translacionog i rotacionog kretanja



Glava 14

Globalni pozicioni sistem

Globalni pozicioni sistem (engl. Global Positioning System - GPS) je satelitski sistem za navigaciju. Iako ima i drugih sličnih sistema, to je jedini takav sistem koji je potpuno funkcionalan i upotrebljiv u čitavom svetu i trenutno ima oko milijardu uređaja koji mogu da ga koriste. Sistem je razvila vojska SAD za svoje potrebe i potpuno je funkcionalan od 1995. godine, a jednom trenutku učinjen je dostupnim za slobodnu upotrebu, kao javno dobro. Dodatno, 2000. godine američka vlada prekinula je sa praksom namernog slabljenja GPS signala za javnu upotrebu, što je omogućilo deset puta veću preciznost nego do tada. GPS je kompleksan i skup sistem, ali osnovni principi i procedure na kojima funkcioniše su relativno jednostavni.

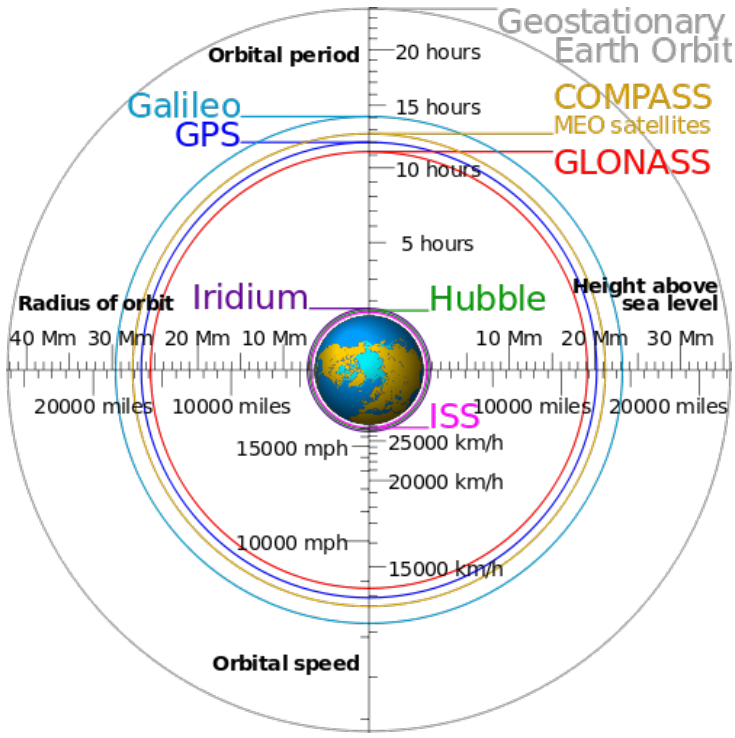
Pored američkog, još uvek najpopularnijeg navigacionog sistema GPS, postoji još nekoliko globahlnih navigacionih sistema (GNSS), od kojih su neki već kompletirani i potpuno funkcionalni a neki treba uskoro da postanu: GLONASS (Rusija), Galileo (EU), BeiDou (Kina), QZSS (Japan), IRNSS — NAVIC (Indija).

14.1 Sateliti, orbite i napajanje

Trenutno u Zemljinoj orbiti ima oko 6000 satelita, od kojih radi oko 40%.

Orbite male visine (Low-Earth orbits = LEO) Naučni sateliti su obično sasvim blizu Zemlji, na svega nekoliko stotina kilometara, kreću se po skoro kružnoj putanji i obilaze Zemlju za svega sat i po (neki od njih po tzv. polarnoj orbiti koja prolazi kroz Južni i Severni pol).

Orbite srednje visine (Medium-Earth orbits – MEO) MEO orbite su na oko 10 puta većim visinama od LEO orbita i MEO satelitima obično treba 12h da obiđu Zemlju čime su u isto vreme uvek iznad istog mesta na Zemlji.



Orbite velike visine (High-Earth orbits – HEO) Mnogi sateliti kreću se na visini 36000 km iznad površine Zemlje. Ovo je tzv. geostacionarna orbita i satelitu treba tačno jedan dan da obiđe Zemlju. Kako i Zemlja rotira, satelit može da uvek bude iznad iste tačke na Zemljinoj površini. Komunikacioni i sateliti često se kreću geostacionarnim putanjama kako bi funkcionisali uređaji koji su uvek na isti način usmereni ka njima. Slično, meteorološki sateliti često se kreću geostacionarnim putanjama kako bi uvek prikupljali podatke sa istog dela Zemlje (nasuprot LEO satelitima koji prikupljaju podatke sa raznih delova Zemlje, ali za jedan deo nemaju previše vremena na raspolaganju).

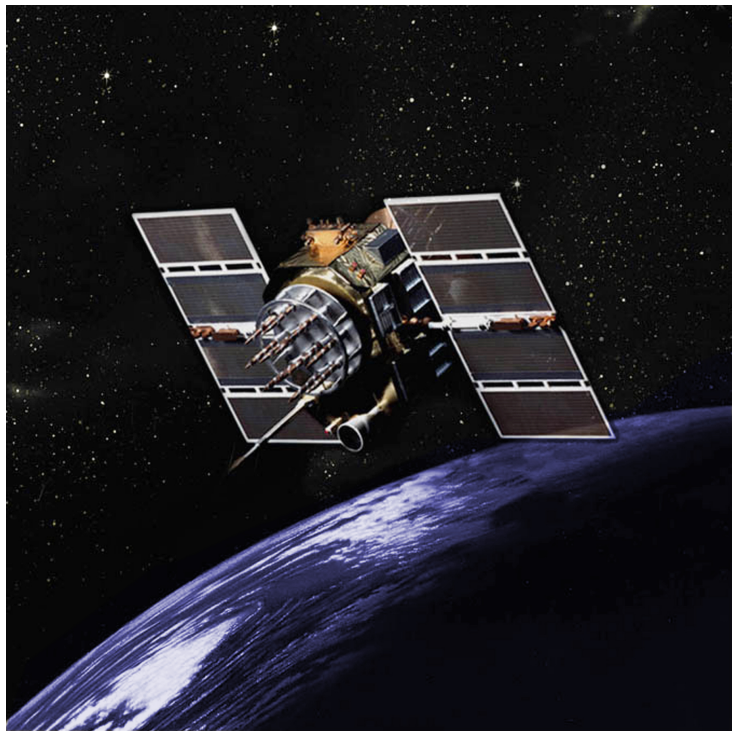
Sateliti koriste sunčevu energiju, koja se – za periode dok je satelit u Zemljinoj senci – čuva u baterijama. Za manevre se koriste dodatni izvori energije (i po nekoliko stotina litara specijalnog goriva zahvaljujući kojem satelit može da manevriše i ostaje u orbiti i više od 10 godina).

Kada se završi radni vek satelita, on se podigne (korišćenjem sačuvanog energenta) u orbitu oko 300 km višu, prepusti se laganom povratku na Zemlju koje može da traje godinama, ili kontrolisanom padu. Prilikom pada, najveći deo satelita bude spržen prolaskom kroz atmosferu.

Postoji ogroman broj komada svemirskog otpada u orbiti – procenjuje se oko 13000.

14.2 GPS sateliti

GPS je zasnovan na sistemu, konstelaciji satelita koji kruže oko Zemlje. Za punu konstelaciju potrebno je barem 24 funkcionalna satelita, pri čemu je taj broj obično i veći (oko 30). Nakon nekoliko godina (ili decenija) neki sateliti izlaze iz upotrebe i zamenjuju ih novi (najstariji GPS satelit koji je bio u upotrebi 2016. godine bio je lansiran 1997. godine). Uvek postoji i nekoliko satelita koji su u rezervi, za slučaj da neki od aktivnih satelita prestane da funkcioniše.



Svaki od ovih satelita (teških 1500 do 2000kg) kruži oko Zemlje na visini od oko 20000 km (u MEO orbiti) i napravi dve potpune rotacije u toku jednog dana. Orbite su definisane tako da u svakom trenutku, za svaku tački na Zemlji postoje barem pet „vidljivih“ satelita[1]. Iz jednog satelita na visini od 20000 km vidljivo je oko 38% Zemljine površine. Sateliti su stacionirani na ovoj visini (mnogo iznad Međunarodne svemirske stanice (visina oko 400 km) ili teleskopa

[1] U stvarnosti, za većinu tačaka na Zemlji, u svakom trenutku vidljivo je barem 6 satelita.

Habl) iz više razloga, pri čemu je najvažniji veća pokrivenost Zemlje nego sa manjih visina. Nije nužno da se sateliti nalaze u orbiti gde je vreme kruženja 12h (ili neki drugi celobrojni količnik od 24h), ali to olakšava izračunavanja (i posebno je bilo važno u ranim godinama upotrebe sistema).

Do sada je lansirano ukupno 50 GPS satelita, a trenutno je u funkciji njih 30. Najnovija grupa satelita Block IIF (proizvođača Boing) lansirana je od 2010. godine. Najnoviji sateliti iz te serije koštaju po oko 130 miliona dolara.

14.3 Lokacija GPS satelita

Lociranje GPS prijemnika zasnovano je na poznavanju lokacija GPS satelita. Zato je potrebno da su lokacije satelita određene veoma precizno i GPS uređaj mora da ih zna.

Orbitalno kretanje satelita u velikoj meri je veoma pravilno i predvidivo. Opis putanje i merenja sa oko 400 fiksnih mesta na kopnu daju izuzetno precizne podatke o lokaciji satelita. Na osnovu podataka sa 25 kopnenih stanica lokacija satelita se računa sa odstupanjem od nekoliko decimetara. Na osnovu tih merenja, američko Ministarstvo odbrane redovno satelitu šalje podatke o njegovoj egzaktnoj lokaciji.

GPS sateliti kreću se po ustaljenim orbitama, pri čemu se opisi tih orbita čuvaju na svakom GPS uređaju. Sunce i Mesec utiču vrlo malo na te putanje, a potrebna podešavanja opisa orbita, kao deo signala satelita, redovno šalje takođe američko Ministarstvo odbrane.

14.4 GPS signal, almanah i efemeris

Svaki GPS satelit emituje signal koji se prenosi radio talasima frekvencije oko 1600 MHz (FM radio talasi se emituju na frekvencijama između 87.5 i 108.0 MHz, dok WiFi koristi frekvencije između 2400 MHz i 5000 MHz). Preciznije, svi sateliti emituju signal L1 na 1575.42 MHz i signal L2 na 1227.6 MHz. GPS signal sadrži vreme u nedelji (na osnovu atomskog časovnika koji se nalazi na satelitu), GPS broj nedelje i izveštaj o stanju satelita (kako bi mogao da bude isključen ako je u kvaru). Svaki prenos traje 30 sekundi i prenosi 1500 bitova podataka. Podaci su kriptovani pseudo-random nizom (PRN) koji se razlikuje za svaki satelit. GPS prijemnici znaju PRN kodove za svaki satelit, pa ne samo da mogu da dekodiraju signal, već i da razlikuju različite satelite.

Sateliti neprekidno emituju dve vrste informacija potrebne prijemniku da odredi tačnu lokaciju GPS satelita: almanah (eng. almanac) i efemeris (eng. ephemeris). Almanah sadrži informaciju o statusu satelita i približnoj orbiti kretanja, dok efemeris daje veoma precizne informacije o orbiti satelita kojim se dopunjuje almanah. Efemeris se ažurira svaka dva sata i obično važi četiri sata.

14.5 GPS prijemnik i startovanje

GPS uređaj (zaseban ili u vidu pametnog telefona) je samo prijemnik, pasivan uređaj koji ne emituje nikakve signale već samo može da ih prima.

Prijemnik obično ima u sebi kompletan almanah i koristi ove informacije da izračuna koji sateliti su mu trenutno vidljivi. Ako je prijemnik nov i nema ugrađen almanah, ili nije korišćen neko vreme, ili je prethodno korišćen na udaljenoj lokaciji, može mu biti potrebno i 15 minuta da primi tekući almanah. GPS uređaju treba ispravni almanah, inicijalna lokacija, tačno vreme i efemeris podaci. Za prijem podataka (posebno inicijalnih) dobro je da prijemnik bude na otvorenom prostoru, bez prepreka okolo. U principu, moguće su sledeće situacije:

Hladni start: Ako GPS uređaj nije bio korišćen neko vreme ili ako je bio korišćen nekoliko stotina kilometara dalje, trebaće mu neko vreme (nekoliko ili čak desetak minuta) za početak rada. U ovakvom stanju, prijemnik nema tekući almanah, efemeris, inicijalnu poziciju, ni tačno vreme.

Ukoliko je uređaj korišćen nekoliko stotina kilometara dalje, pretpostavka o lokaciji koju uređaj ima je pogrešna i moraće da traži satelite. U takvim situacijama, neki uređaji daju korisniku mogućnost da unese približnu lokaciju.

Topli start: Tekući almanah, inicijalna pozicija i tačno vreme su poznati. Efemeris podaci su neispravni ili samo delom ispravnim. Za početak rada može biti potrebno nekoliko desetina sekundi.

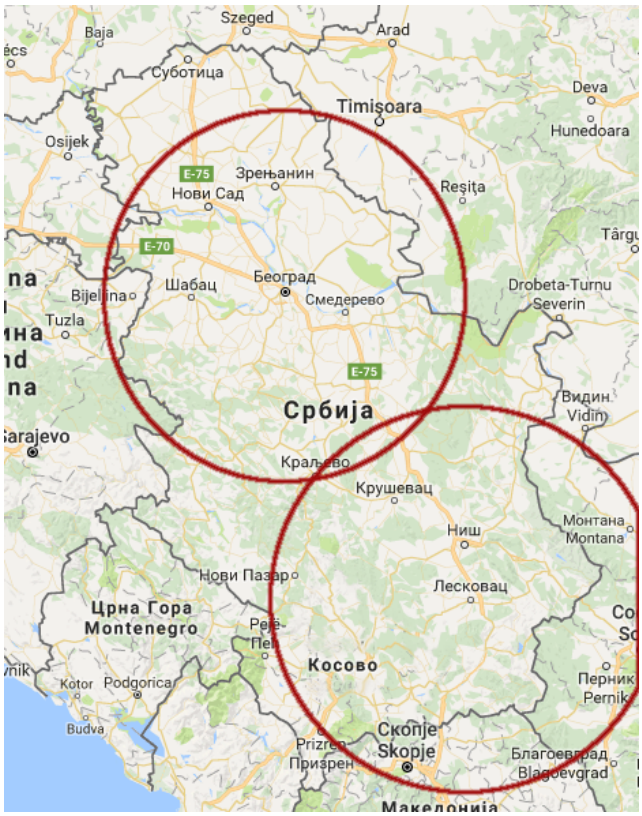
Vrući start: Ako je prijemnik bio isključen manje od, na primer, jednog sata i ako nije bila promenjena njegova lokacija, on može početi sa radom za nekoliko ili desetak sekundi.

14.6 Trilateracija

Zadatak GPS prijemnika je da locira četiri GPS satelita, odredi rastojanja do njih i, na osnovu toga, svoju tačnu lokaciju, korišćenjem postupka *trilateracije*.

Ilustrujmo najpre postupak trilateracije u ravni. Pretpostavimo da se nalazimo negde u Srbiji i da je potrebno odrediti gde je to. Pretpostavimo da je poznato da je rastojanje (vazdušnim putem) do Beograda — 120 km. To, naravno, ne omogućava lociranje položaja jer isto važi za sve tačke na krugu sa središtem u Beogradu, poluprečnika 120 km. Pretpostavimo, međutim, da je poznato i rastojanje do Leskovca – 130 km. Sa ovom informacijom, postoje samo dve moguće lokacije. Ukoliko je poznato i rastojanje do Novog Pazara – 70 km, onda se može eliminisati jedna od te dve mogućnosti, čime se locira položaj – nalazimo se u Kraljevu.

Analogni postupak primenjuje se u prostoru, pri čemu se koriste sfere umesto krugova. Ako znamo da se nalazimo na rastojanju od 21000 km od tačke



(tj. satelita) A u svemiru, mogli bismo da budemo bilo gde na ogromnoj sferi poluprečnika 21000 km. Ako znamo i da se nalazimo 22000 km od tačke B negde u svemiru, onda se svakako nalazimo na preseku sfere poliprečnika 21000 km i središtem A i sfere poluprečnika 22000 km i središtem B. Taj presek je krug. Ako znamo i rastojanje od treće tačke, onda odgovarajuća sfera seče taj krug u dve tačke. To će biti dovoljno za lociranje, jer Zemlja može da ima ulogu četvrte sfere: samo jedna od te dve tačke može da bude na površini Zemlje (druga je hiljadama kilometara iznad površine).

Da bi odredio tačnu lokaciju, GPS prijemnik treba da zna sledeće:

- Lokaciju barem tri satelita koja su „vidljiva“;
- Tačno rastojanje od prijemnika do svakog od tih satelita.

GPS uređaj do navedenih informacija dolazi analiziranjem signala sa GPS satelita. Bolji GPS uređaji imaju više prijemnika, pa mogu da primaju signal sa nekoliko satelita u isto vreme. Radio signal putuje brzinom svetlosti (oko 299792 km/s u vakuumu). GPS prijemnik može da zaključi koliko je satelit

udaljen ako može da zaključi koliko dugo je signal putovao (a putuje veoma kratko, tek oko 0,06s).

U određeno vreme, recimo – u ponoć, satelit počinje da emituje dug digitalni obrazac, tzv. preudo-slučajan kod (pseudo-random code). Prijemnik počinje da izračunava isti digitalni obrazac tačno u ponoć. Kada signal sa satelita stigne do prijemnika, on će kasniti za obrascem koji se generiše na prijemniku. To kašnjenje je upravo jednako trajanju putovanja signala, pa je rastojanje do satelita jednako tom trajanju pomnoženom sa brzinom svetlosti.

14.7 Ažuriranje tačnog vremena

GPS uređaj može da odredi svoju lokaciju ukoliko su poznata egzaktna rastojanja od tri satelita i njihove egzaktne lokacije. Problem je, međutim, u tome da li je informacija o tačnom vremenu sinhronizovana na satelitima i na prijemniku. Pokazaće se da su, zbog toga, za lociranje potrebna barem četiri vidljiva satelita.

Da bi trilateracija funkcionisala, i prijemnik i satelit moraju da imaju časovnike koji su sinhronizovani potpuno precizno. To bi moglo da se osigura atomskim časovnicima i na satelitu i na prijemniku, ali to nije realistično rešenje jer su atomski časovnici preskupi za ugradnju u navigacione uređaje (najeftiniji i najneprecizniji atomski časovnici koštaju oko 1500 dolara, dok vrhunski atomski časnovnici koštaju preko 50000 dolara).

Ovaj problem rešava se na sledeći način. Svaki satelit ima vrhunski atomski časovnik, ali prijemnici imaju samo jednostavne i jeftine kvarcne časovnike na

kojima se vreme podešava i koriguje neprestano.

Ukoliko se određuje rastojanje do tri satelita – presek tri sfere sigurno postoji. Međutim, ako se razmatraju rastojanja do četiri satelita – presek četiri sfere ne mora da postoji. Tada su sva izračunata rastojanja netačna zbog netačnog časovnika na prijemniku i to – zbog istog odstupanja. Prijemnik može lako da odredi potrebno podešavanje časovnika uz koje će sve četiri sfere da se seku. Na osnovu tog izračunavanja, prijemnik takođe podešava svog interni sat (tj. sinhronizuje ga sa časovnicima na satelitima).

14.8 GPS i teorija relativnosti

Kako posmatrač, tj. prijemnik vidi satelit u pokretu (relativno u odnosu na njega), specijalna teorija relativnosti predviđa da će, zbog dilatacije vremena, posmatrač sa Zemlje videti da časovnici na satelitu rade sporije nego časovnici na Zemlji, i to dodatnih 7 mikrosekundi svakog dana (jedna mikrosekunda je milioniti deo sekunde).

Dodatno, sateliti su u orbiti daleko iznad Zemlje, gde je zakrivljenost prostor-vreme zbog Zemljine mase manja nego na Zemljinoj površini. Opšta teorija relativnosti kaže da će se činiti da časovnik bliži masivnom objektu kuca sporije nego časovnik koji je dalji, pa kada se gledaju sa Zemlje, čini se da časovnici na satelitima kucaju brže nego na površini Zemlje. Izračunavanje na osnovu opšte teorije relativnosti kaže da će se činiti da časovnik na GPS satelitu ide ispred časnovnika na Zemlji 45 mikrosekundi svakog dana.

U kombinaciji, ova dva relativistička efekta daju da će posmatraču sa Zemlje čini da časovnici na GPS satelitu idu brže nego časovnici na Zemlji, i to 45 mikrosekundi -7 mikrosekundi = 38 mikrosekundi svakog dana. Ovo se čini kao malo odstupanje, ali sa ovakvim vremenskim odsupanjem dnevno bi se akumulirala greška u lociranju od čak 10km.

Opšti relativistički efekat se obrađuje (ili - poništava) tako što su atomski časovnici na satelitima usporeni do potrebnog nivoa. Što se tiče specijalnog relativističkog efekta, njega uzimaju u obzir prijemnici prilikom postupka trilateracije.

14.9 Očekivana preciznost i mere za popravljanje preciznosti

Nominalna preciznost GPS sistema je oko 15m, ali u idealnim uslovima (i sa najboljim prijemnikom) može biti i 5m ili čak 3m. Obično se smatra da je sa ručnim GPS uređajem preciznost 10m u 95% pozicija.

Preciznost GPS navigacije zavisi od mnogih faktora, kao što je pozicija satelita na nebu, stanja atomskih časovnika na satelitima, vremenski faktori (signal ipak ne putuje brzinom svetlosti, nego nešto manjom, posebno u nekim delovima atmosfere), greške u efemerskim podacima, okolina sa visokim objektima (pa satelitski signal stiže nakon odbijanja od njih), itd.

Neke od mera za povećavanje preciznosti:

- korišćenje mreže fiksniranih, kopnenih referentnih stanica (DGPS) može da popravi nominalnu GPS preciznost sa 15m na 10cm u idealnim uslovima.
- korišćenjem kombinacije GPS i ruskih GLONASS navigacionih satelita.