Kompjuterska geometrija

Uvod

Kompjuterska geometrija je grana računarskih nauka koja se bavi istraživanjem algoritama za rešavanje geometrijskih problema. U modernom inženjerstvu i matematici, kompjuterska geometrija, između ostalog, ima primenu u kompjuterskoj grafici, robotici, dizajniranju procesora, kompjuterski podržanom projektovanju i statistici. Ulaz u problem kompjuterske geometrije je obično opis skupa geometrijskih objekata, kao što je skup tačaka, skup duži ili temena poligona. Izlaz je često odgovor na pitanje o objektima, kao što je pitanje da li se oni međusobno seku, novi geometrijski objekat i slično.

U ovom poglavlju ćemo razmatrati geometrijske algoritme u dve dimenzije, tj. u ravni. Svaki ulazni objekat je predstavljen skupom tačaka $\{p_1, p_2, p_3, ...\}$, pri čemu je $p_i = (x_i, y_i)$ i $x_i, y_i \in \mathbf{R}$. Na primer, n-ugaoni poligon P je predstavljen nizom temena $(p_0, p_1, ..., p_{n-1})$, po redosledu njihovog pojavljivanja na njegovom obodu. Kompjuterska geometrija se može primeniti i u tri dimenzije, ili čak u višedimenzionim prostorima, ali se njihova rešenja teško mogu vizuelizovati. Međutim, čak i u dve dimenzije možemo prikazati primere tehnika kompjuterske geometrije.

Osobine duži

Nekoliko algoritama kompjuterske geometrije u ovom poglavlju će zahtevati odgovore na pitanja o osobinama duži. **Konveksna kombinacija** dve različite tačke $p_1 = (x_1, y_1)$ i $p_2 = (x_2, y_2)$ je svaka tačka $p_3 = (x_3, y_3)$ takva da za neko α iz opsega $0 \le \alpha \le 1$, imamo da je $x_3 = \alpha x_1 + (1-\alpha) x_2$ i $y_3 = \alpha y_1 + (1-\alpha) y_2$. Takođe možemo napisati da je $p_3 = \alpha p_1 + (1-\alpha) p_2$. Jasno je da je p_3 bilo koja tačka koja leži na pravoj koja prolazi kroz tačke p_1 i p_2 i nalazi se na ili između tačaka p_1 i p_2. Za dve različite tačke p_1 i p_2, **duž** $\overline{p_1 p_2}$ predstavlja skup konveksnih kombinacija od p_1 i p_2. Tačke p_1 i p_2 nazivamo krajnjim tačkama duži $\overline{p_1 p_2}$. U nekim slučajevima je od značaja redosled tačaka p_1 i p_2 i tada govorimo o usmerenoj duži $\overrightarrow{p_1 p_2}$. Ukoliko je p_1 koordinatni početak, onda usmerenu duž $\overrightarrow{p_1 p_2}$ možemo posmatrati kao **vektor** p_2.

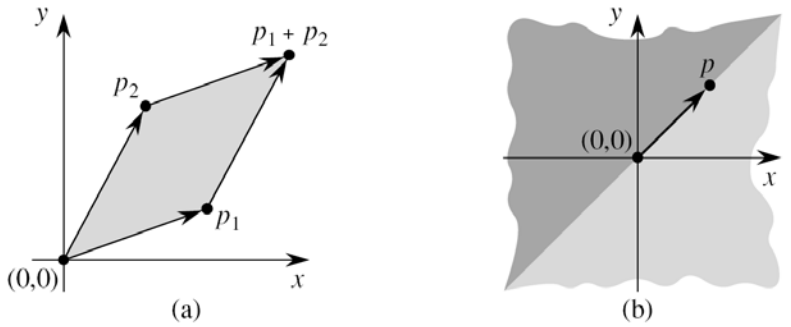
U ovoj sekciji razmatramo sledeće probleme:

- Za dve usmerene duži $\overrightarrow{p_0 p_1}$ i $\overrightarrow{p_0 p_2}$ odrediti da li se $\overrightarrow{p_0 p_1}$ nalazi u smeru kazaljke na satu u odnosu na $\overrightarrow{p_0 p_2}$, kada se rotacija vrši oko zajedničke tačke p_0.
- Za date dve duži $\overline{p_0 p_1}$ i $\overline{p_1 p_2}$ da li prilikom prolaska kroz $\overline{p_0 p_1}$, a zatim kroz $\overline{p_1 p_2}$ pravimo skretanje u levo u tački p_1?
- Da li se duži $\overline{p_1 p_2}$ i $\overline{p_3 p_4}$ međusobno seku?

Odgovori na sva ova pitanja su kompleksnosti $O(1)$, što ne bi trebalo da začudi imajući u vidu da je ulaz za svako pitanje veličine $O(1)$. Štaviše, naše metode će koristiti samo sabiranje, oduzimanje, množenje i poređenje. Nisu nam potrebni ni deljenje ni trigonometrijske funkcije, koje mogu biti računski veoma skupe i mogu dovesti do problema sa zaokruživanjem. Na primer, metoda za određivanje da li se dve duži seku tako što izračunava jednačinu prave u obliku $y = mx + b$ za svaku duž (m je nagib, a b presek ose y), određuje tačku na preseku ove dve prave i na kraju proverava da li se dobijena tačka nalazi na ove dve duži, koristi deljenje za da bi odredila tačku preseka. Kada su duži približno paralelne, ova metoda je veoma osetljiva na preciznost operatora deljenja kod realnih računara. Metoda koja će biti predstavljena u ovoj sekciji i koja ne koristi deljenje je znatno preciznija.

Vektorski proizvod

Računanje vektorskog proizvoda je ključna stvar kod metoda za rad sa dužima. Posmatrajmo vektore p_1 i p_2, prikazane na Slici ###a. **Vektorski proizvod** $p_1 \times p_2$ se može posmatrati kao označena površina paralelograma koji formiraju tačke $(0,0)$, p_1, p_2 i $p_1 + p_2 = (x_1 + x_2, y_1 + y_2)$.



Slika ###. a) Vektorski proizvod vektora p_1 i p_2 je označena površina paralelograma. b) Svetla oblast sadrži vektore koji su u smeru kazaljke na satu u odnosu na vektor p, a tamna sadrži vektore koji su u suprotnom smeru od kazaljke na satu u odnosu na p.

Ekvivalentna, ali korisnija definicija vektorskog proizvoda prikazuje vektorski proizvod determinante matrice

$$p_1 \times p_2 = \begin{vmatrix} x_1 & y_1 \\ x_2 & y_2 \end{vmatrix} = x_1 y_2 - x_2 y_1 = -p_2 \times p_1$$

Ako je $p_1 \times p_2$ pozitivno, onda je p_1 u smeru kazaljke na satu u odnosu na p_2, kada se rotacija vrši oko tačke $(0,0)$. Suprotno, ukoliko je vektorski proizvod negativan, onda je p_1 u suprotnom smeru od kazaljke na satu u odnosu na p_2. Na Slici ###b su prikazane oblasti u smeru kazaljke i suprotno od kazaljke u odnosu na vektor p. Granični slučaj nastaje kada je vektorski proizvod 0. U tom slučaju vektori su **kolinearni** i okrenuti su u istom ili suprotnom smeru. Da bismo odredili da li je usmerena duž $\overrightarrow{p_0 p_1}$ u smeru kazaljke na satu u odnosu na usmerenu duž $\overrightarrow{p_0 p_2}$, kada se rotira oko zajedničke tačke p_0, vršimo jednostavnu translaciju da bismo iskoristili p_0 kao koordinatni početak. To praktično znači da $p_1 - p_0$ označimo kao vektor $p_1' = (x_1', y_1')$, gde je $x_1' = x_1 - x_0$ i $y_1' = y_1 - y_0$, as lično i vektor $p_2 - p_0$. Zatim izračunavamo vektorski proizvod

$$(p_1 - p_0) \times (p_2 - p_0) = (x_1 - x_0)(y_2 - y_0) - (x_2 - x_0)(y_1 - y_0)$$

Ako je vektorski proizvod pozitivan, onda je $\overrightarrow{p_0 p_1}$ u smeru kazaljke na satu u odnosu na $\overrightarrow{p_0 p_2}$. Suprotno, ukoliko je negativan onda je u smeru suprotnom od kazaljke na satu.

Određivanje da li dve uzastopne duži skreću u levo ili desno

Sledeće pitanje je da li dve uzastopne duži $\overrightarrow{p_0 p_1}$ i $\overrightarrow{p_1 p_2}$ skreću u levo ili u desno u tački p_1. Drugim rečima, potrebna nam je metoda za određivanje smera u kome skreće ugao $\angle p_0 p_1 p_2$. Vektorski proizvod nam omogućava da odgovorimo na ovo pitanje bez izračunavanja ugla. Kao što je prikazano na Slici ###, jednostavno proveravamo da li je usmerena duž $\overrightarrow{p_0 p_2}$ u istom ili suprotnom smeru kao kazaljka na satu u odnosu na usmerenu duž $\overrightarrow{p_0 p_1}$. Da bismo ovo uradili izračunavamo vektorski proizvod $(p_2 - p_0) \times (p_1 - p_0)$. Ako je znak vektorskog proizvoda negativan, onda je $\overrightarrow{p_0 p_2}$ u suprotnom smeru od

kazaljke na satu u odnosu na $\overrightarrow{p_0 p_1}$, što znači da duži skreću u levo u tački p_1. Pozitivan vektorski proizvod govori da je usmerenje u smeru kazaljke na satu, pa samim tim i skretanje je u desno. Ukoliko je vektorski proizvod 0, to znači da su tačke p_0, p_1 i p_2 kolinearne.

Određivanje da li se dve duži seku

Da bismo odredili da li se dve duži seku, proveravamo da li svaka duž opkoračava pravu koja sadrži drugu duž. Duž $\overline{p_1 p_2}$ **opkoračava** pravu ukoliko tačka p_1 leži sa jedne strane prave, a tačka p_2 sa druge strane. Granični slučaj se javlja kada p_1 ili p_2 leže na pravoj. Dve duži se seku ako i samo ako je neki od sledeća dva uslova ispunjen:

1. Svaka duž opkoračava pravu koja sadrži drugu duž.
2. Neka krajnja tačka jedne duži leži na drugoj duži. (Ovaj uslov proizilazi iz graničnog slučaja.)

Sledeća procedura implementira ovu ideju. SEGMENTS_INTERSECT vraća TRUE ukoliko se duži $\overline{p_1 p_2}$ i $\overline{p_3 p_4}$ seku, a FALSE ukoliko se ne seku. Ona poziva proceduru DIRECTION, koja računa međusobnu orijentaciju korišćenjem vektorskog proizvoda na ranije opisan način i proceduru ON_SEGMENT, koja određuje da li tačka za koju se zna da je kolinearna sa duži leži na toj duži.

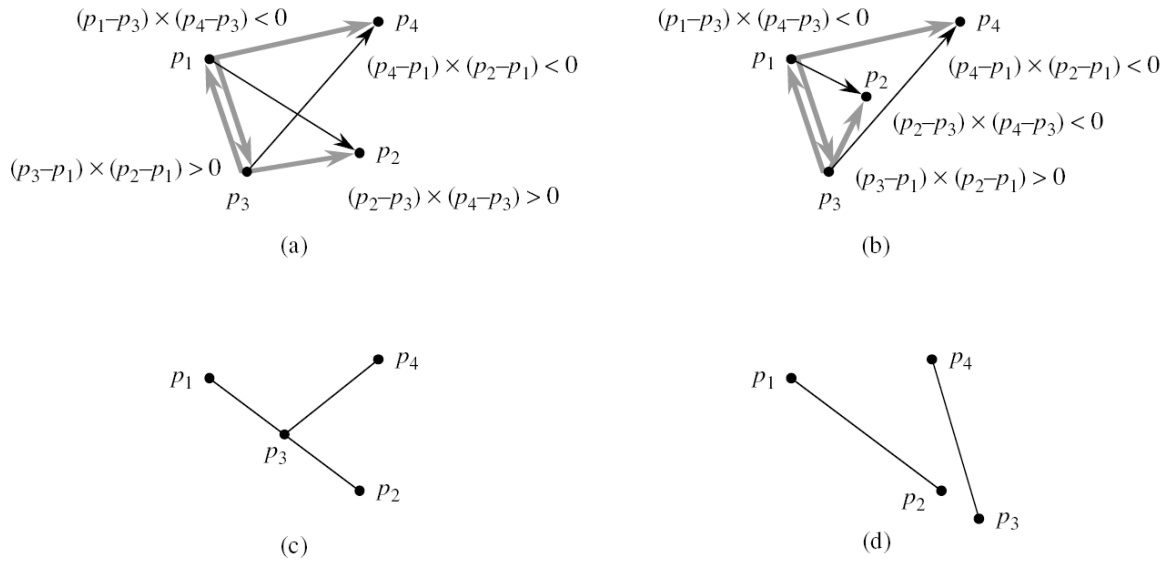
```
SEGMENTS-INTERSECT( p1 , p2 , p3 , p4 )
1     d1 = DIRECTION( p3 , p4 , p1 )
2     d2 = DIRECTION( p3 , p4 , p2 )
3     d3 = DIRECTION( p1 , p2 , p3 )
4     d4 = DIRECTION( p1 , p2 , p4 )
5     if (( d1 >0 and d2 <0) or ( d1 <0 and d2 >0)) and
         (( d3 >0 and d4 <0) or ( d3 <0 and d4 >0))
6        then return TRUE
7     elseif d1 =0 and ON-SEGMENT( p3 , p4 , p1 )
8        then return TRUE
9     elseif d2 =0 and ON-SEGMENT( p3 , p4 , p2 )
10       then return TRUE
11    elseif d3 =0 and ON-SEGMENT( p1 , p2 , p3 )
12       then return TRUE
13    elseif d4 = 0 and ON-SEGMENT( p1 , p2 , p4 )
14       then return TRUE
15    else return FALSE

DIRECTION( pi , pj , pk )
1  return (pk − pi) × (pj − pi)

ON-SEGMENT( pi , pj , pk )
1  if min( xi , xj ) ≤ xk ≤ max( xi , xj ) and min( yi , yj ) ≤ yk ≤ max( yi , yj )
2     then return TRUE
3  else return FALSE
```

Procedura SEGMENTS_INTERSECT funkcioniše na sledeći način. Linije 1-4 izračunavaju relativnu orijentaciju d_i svake tačke p_i u odnosu na drugu duž. Ako su sve relativne orijentacije različite od nule, onda na jednostavan način možemo odrediti da li se duži $\overline{p_1 p_2}$ i $\overline{p_3 p_4}$ seku na sledeći način. Duž $\overline{p_1 p_2}$ opkoračuje

pravu koja sadrži duž $\overline{p_3p_4}$ ukoliko usmerene duži $\overrightarrow{p_3p_1}$ i $\overrightarrow{p_3p_2}$ imaju suprotnu orijentaciju u odnosu na $\overrightarrow{p_3p_4}$. U tom slučaju se znaci promenljivih d_1 i d_2 razlikuju. Ako je uslov u liniji 5 tačan, onda se duži međusobno opkoračavaju i SEGMENTS_INTERSECT vraća TRUE. Slika ###a prikazuje ovaj slučaj. U suprotnom se duži ne opkoračavaju međusobno, i može se ispitati granični slučaj. Ako su sve relativne orijentacije različite od nule, ne radi se o graničnom slučaju. U tom slučaju svi uslovi od 7-13 su netačni i procedura SEGMENTS_INTERSECT vraća FALSE u liniji 15. Slika ###b prikazuje ovaj slučaj.



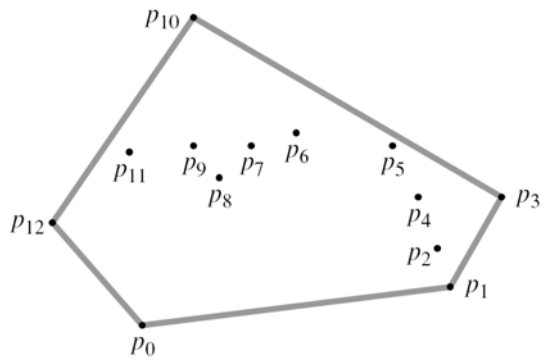
Slika ###. Slučajevi u proceduri SEGMENTS_INTERSECT.

Granični slučaj se javlja ukoliko je bilo koja relativna orijentacija d_k jednaka nuli. Tada znamo da je p_k kolinearno sa drugom duži. Ova tačka leži na drugoj duži ako i samo ako se nalazi između krajnjih tačaka druge duži. Procedura ON_SEGMENT vraća da li je p_k između krajnjih tačaka duži $\overline{p_ip_j}$, tj. druge duži u pozivima u linijama 7-13. Procedura podrazumeva je p_k kolinearno sa duži $\overline{p_ip_j}$. Slike ###c i ###d prikazuju slučajeve sa kolinearnim tačkama. Na Slici ###c, p_3 leži na p_1p_2, tako da SEGMENTS_INTERSECT vraća TRUE u liniji 12. Na Slici ###d ni jedna krajnja tačka ne leži na drugoj duži, pa SEGMENTS_INTERSECT vraća FALSE u liniji 15.

Određivanje konveksnog omotača

Konveksni omotač (eng. *convex hull*) skupa tačaka Q je najmanji konveksni poligon P takav da svaka tačka iz skupa Q leži u unutrašnjosti poligona ili na njegovoj granici. Prema engleskom nazivu, konveksni omotač skupa Q često označavamo sa $\mathrm{CH}(Q)$. Drugačije gledano, ako zamislimo da svaka tačka predstavlja ekser koji viri iz daske, onda bi konveksni omotač bila figura koju formira zategnuta gumica koja obuhvata sve eksere. Na Slici ### je prikazan skup tačaka i njihov konveksni omotač.

U ovoj sekciji ćemo predstaviti Grahamov algoritam (Grahamova pretraga) za određivanje konveksnog omotača za skup od n tačaka. Algoritam određuje vraćaju temena konveksnog omotača u smeru suprotnom od kazaljke na satu i ima vreme izvršavanja $O(n\log_2 n)$. Kao što se može videti sa Slike ###, svako teme poligona $\mathrm{CH}(Q)$ je tačka iz skupa Q. Algoritam koristi ovu osobinu, određujući koje tačke iz skupa Q da zadrži kao teme konveksnog omotača, a koje tačke da odbaci.



Slika ###. Skup tačaka i njihov konveksni omotač

Postoji nekoliko metoda za određivanje konveksnog omotača u vremenu $O(n\log_2 n)$. Grahamov algoritam koristi tehniku zvanu "rotacioni prolazak", koja obrađuje tačke po redosledu polarnih uglova koje one formiraju u odnosu na referentnu tačku.

Određivanje konveksnog omotača skupa tačaka je interesantan problem sam po sebi. Međutim, algoritmi za neke druge probleme kompjuterske geometrije počinju određivanjem konveksnog omotača. Posmatrajmo, na primer, dvodimenzioni problem određivanja para najudeljenijih tačaka: za dati skup od n tačaka u ravni potrebno je odrediti dve tačke koje su međusobno najviše udaljene. Može se dokazati da te dve tačke pripadaju konveksnom omotaču. Iako to ovde nećemo dokazivati, najudaljenije tačke n-ugaonog konveksnog poligona se mogu odrediti u vremenu $O(n)$. Odatle, određivanjem konveksnog omotača n ulaznih tačaka u vremenu $O(n\log_2 n)$, a zatim određivanjem para najudaljenijih tačaka rezultujućeg konveksnog poligona, možemo pronaći najudaljeniji par tačaka u bilo kom skupu od n tačaka u vremenu $O(n\log_2 n)$.

Grahamova pretraga

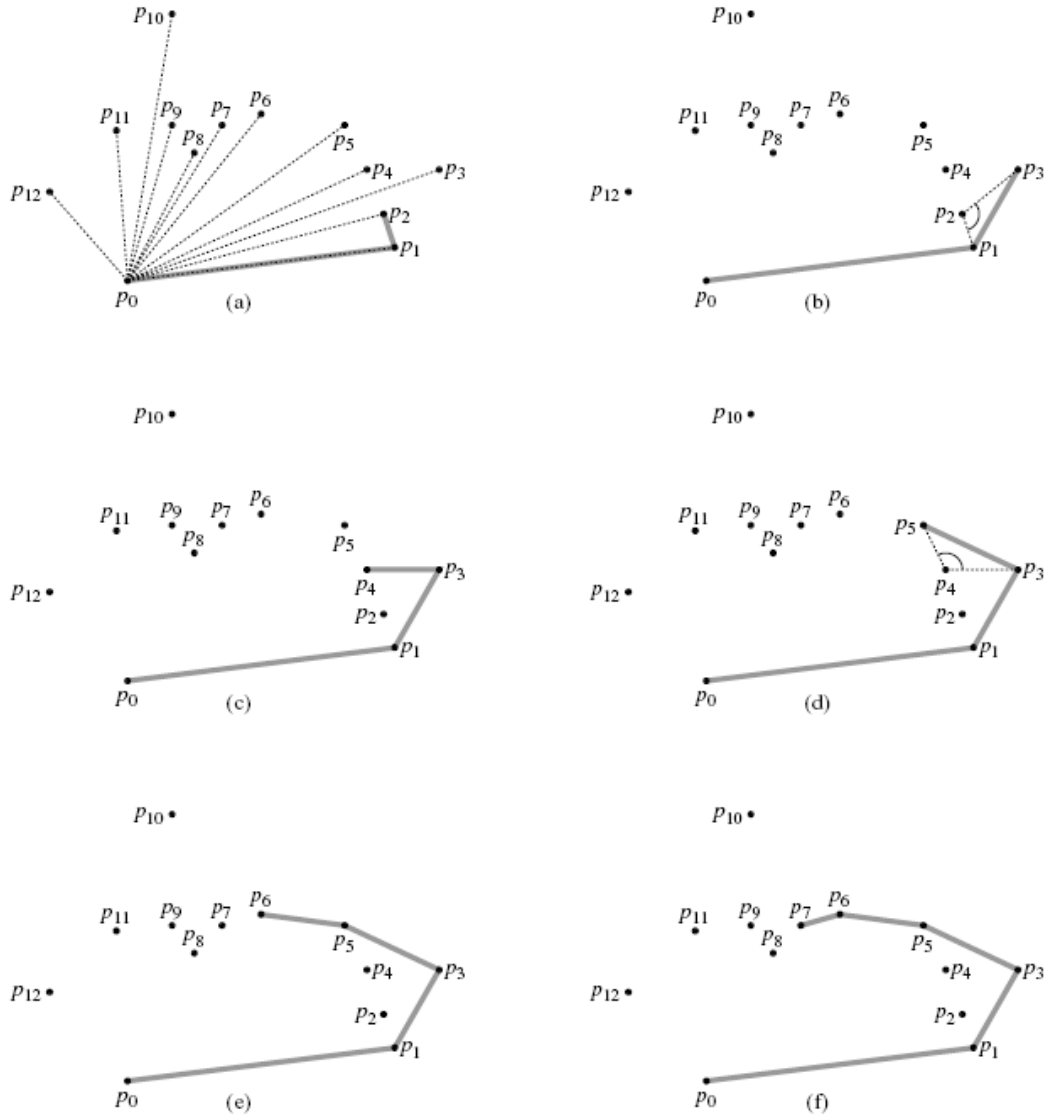
Grahamova pretraga rešava problem konveksnog omotača očuvanjem steka S sa tačkama kandidatima. Svaka tačka ulaznog skupa Q se stavlja jednom na stek, a tačke koje nisu temena poligona $\mathrm{CH}(Q)$ se konačno skidaju sa steka. Kada se algoritam završi, stek S sadrži upravo temena poligona $\mathrm{CH}(Q)$ u redosledu pojavljivanja na poligonu suprotnom od smera kazaljke na satu.

Procedura GRAHAM_SCAN uzima kao ulaz skup tačaka Q, pri čemu je $|Q|\geq 3$. Ova procedura poziva funkciju TOP(S), koja vraća tačku sa vrha steka S bez promene steka, i funkciju NEXT_TO_TOP(S), koja vraća tačku koja se nalazi na jednom mestu ispod vrha steka S bez promene steka. Može se dokazati da stek S koji vraća funkcija GRAHAM_SCAN sadrži od dna ka vrhu upravo temena poligona $\mathrm{CH}(Q)$ u smeru suprotnom od kazaljke na satu.

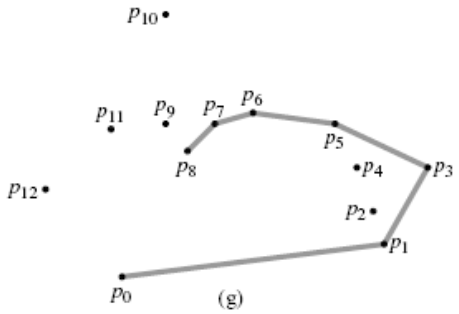
```
GRAHAM_SCAN( Q )
1    neka je p0 tacka u skupu Q sa minimalnom y-koordinatom, ili prva tacka sa
     leve strane u slucaju da ih ima vise sa minimalnom y-koordinatom
2    neka su (p1, p2,..., pm) ostale tacke u skupu, sortirane prema polarnom uglu u
     odnosu na tacku p0 u smeru suprotnom od kazaljke na sat (ukoliko vise tacaka
     ima isti ugao, uklanjaju se sve osim najudaljenije od p0 )
3    PUSH( p0 , S )
4    PUSH( p1 , S )
5    PUSH( p2 , S )
6    for i=3,m
```

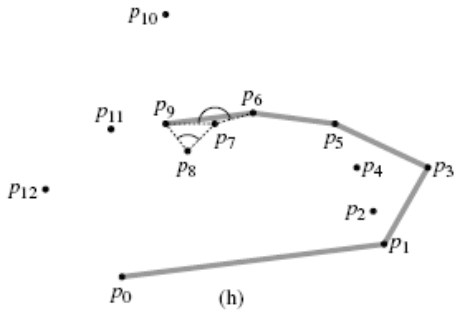
```
7       while ugao koji formiraju tacke NEXT_TO_TOP(S), TOP(S) i  p_i  prave
        "nelevo" skretanje
8          POP(S)
9       PUSH( p_i , S)
10   return  S
```

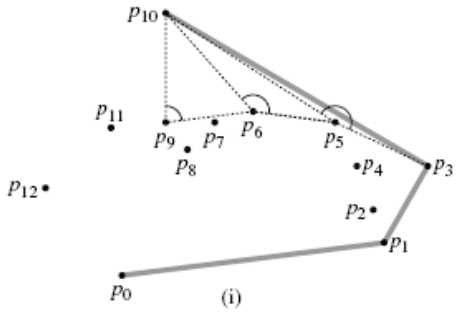
Slika ### prikazuje izvršavanje funkcije GRAHAM_SCAN. Linija 1 bira tačku p_0 kao tačku sa najmanjom y koordinatom, uzimajući prvu sa leve strane ukoliko ih ima više sa minimalnom y koordinatom. S obzirom da nema tačaka u skupu Q koje su ispod p_0, a sve druge tačke sa istom y koordinatom su sa njene desne strane, p_0 je teme poligona $\mathrm{CH}(Q)$. Linija 2 sortira preostale tačke skupa Q po polarnom uglu u odnosu na p_0, poređenjem vektorskih proizvoda. Ukoliko dve ili više tačaka imaju isti polarni ugao u odnosu na p_0, sve tačke među njima osim najudaljenije su konveksna kombinacija tačke p_0 i najudaljenije tačke, tako da ih možemo isključiti iz razmatranja.

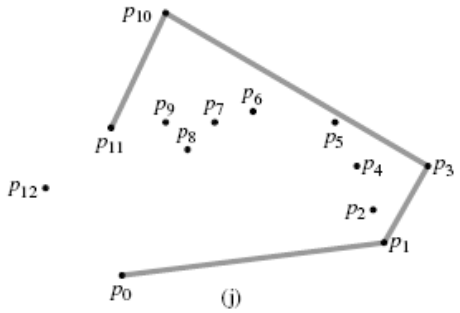


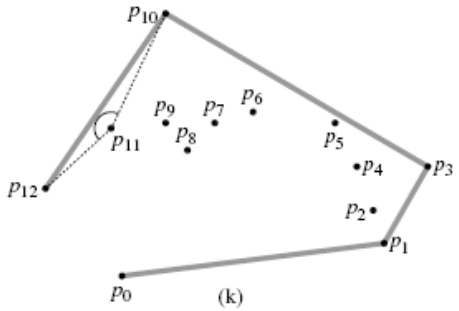
Slika ###. Izvršavanje Grahamove pretrage na skupu tačaka sa Slike ###.

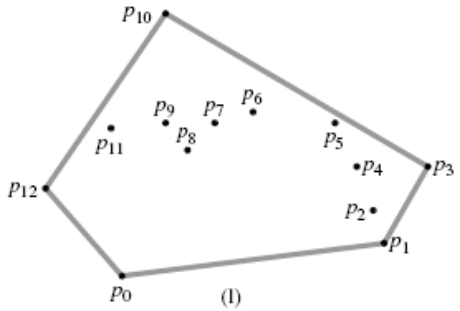












Označimo sa m broj preostalih tačaka osim tačke p_0. Polarni ugao svake tačke iz skupa Q u odnosu na tačku p_0, mereno u radijanima, se nalazi u opsegu $[0, \pi)$. Pošto su tačke sortirane u skladu sa polarnim uglom, to zapravo znači da su sortirane u smeru suprotnom od kazaljke na satu u odnosu na p_0. Ovaj sortirani niz tačaka smo označili sa $(p_1, p_2, ..., p_m)$. Primetimo da su tačke p_1 i p_m temena poligona $\mathrm{CH}(Q)$. Slika ###a prikazuje tačke sa Slike ### redom numerisane u smeru povećanja polarnog ugla u odnosu na tačku p_0.

Ostatak procedure koristi stek S. Linije od 3-5 inicijalizuju stek tako da sadrži, od dna ka vrhu, prve tri tačke p_0, p_1 i p_3. Slika ###a prikazuje početni stek S. **For** petlja u linijama 6-9 prolazi kroz sve tačke podniza $(p_3, p_4, ..., p_m)$. Namera je da nakon obrade tačke p_i stek sadrži, od dna ka vrhu, temena poligona $\mathrm{CH}(\{p_0, p_1, ..., p_i\})$ u smeru suprotnom od kazaljke na satu. **While** petlja u linijama 7-8 uklanja tačke sa steka ukoliko se utvrdi da one nisu temena konveksnog omotača. Kada obilazimo konveksni omotač u smeru suprotnom od kazaljke na satu, trebalo bi da pravimo skretanje u levo u svakom temenu. Zbog toga, svaki put kada **while** petlja pronađe tačku u kojoj se pravi "nelevo" skretanje, tačka se skida sa steka. Provera "nelevog" skretanja se koristi umesto provere desnog skretanja da bi se sprečila mogućnost pravog ugla u tački rezultujućeg konveksnog omotača. Kada uklonimo sa steka sve tačke koje nemaju levo

skretanje prema tački p_i, onda p_i stavljamo na stek. Slike ###(b)-(k) pokazuju stanje steka S posle svake iteracije **for** petlje. Konačno, GRAHAM_SCAN vraća stek S u liniji 10. Slika ###(l) prikazuje odgovarajući konveksni omotač.