Geometrijski algoritmi i strukture podataka

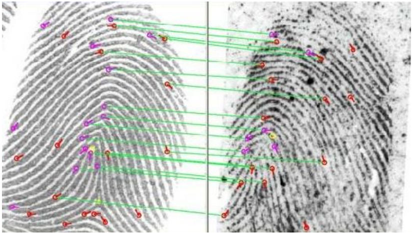
Andrej Ivašković

Matematička gimnazija, Nedelja informatike

1. april 2015.

Zašto bi nas zanimalo?

- Algoritmi kojima ćemo se baviti su dobri kao **ocene** u različitim situacijama:
  - računarska vizija;
  - linearno programiranje (pri uslovima $\mathbf{A}\mathbf{x} \leqslant \mathbf{b}$ i $\mathbf{x} \geqslant 0$ odrediti $\max\{\mathbf{c} \cdot \mathbf{x}\}$);
  - mašinsko učenje.
- Koliko mi treba vremena da stignem do najbližeg restorana?
- Koliko su slična dva otiska prsta?



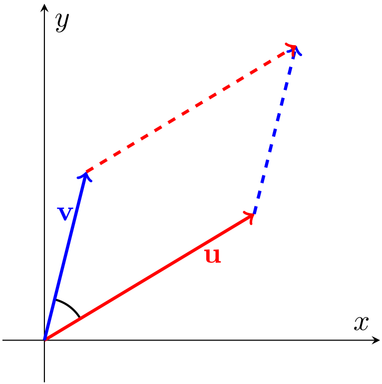
Šta ćemo danas odraditi?

- Posmatrati kako su svojstva **vektorskog proizvoda** korisna u geometriji.
- Definisati **konveksni omotač** i obraditi dva algoritma koja se tiču **konveksnog omotača**.
- Videti na koji način koristimo **kd stablo**.

Opšte napomene

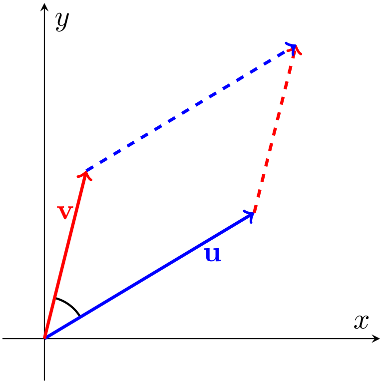
- U narednim algoritmima ću zanemariti probleme koji nastaju zbog *floating point* aritmetike.
- Insistiraćemo na rešenjima koja ne koriste **trigonometrijske funkcije** i **deljenje**.
- Neću nigde dati bilo šta nalik formalnom dokazu algoritma, ali ću se potruditi da "opravdam" zašto radi.
- Pretpostaviću da su mi već definisane strukture koje predstavljaju tačke i vektore u dve dimenzije i da postoje relevantni atributi (x i y). Prelaz na k-dimenzione prostore je nekad lak, nekad problematičan!

Vektorski proizvod vektora



Za dva vektora $\mathbf{u}$ i $\mathbf{v}$ u ravni definišemo njihov **vektorski proizvod** kao vektor $\mathbf{u} \times \mathbf{v}$ ortogonalan na tu ravan i intenziteta koji odgovara površini paralelograma koji grade.

Svojstvo antisimetričnosti



Usmerenost ovog vektora zavisi od orijentacije $\varphi = \sphericalangle(\mathbf{u}, \mathbf{v})$: ako $\varphi > 0$, usmerenost je "iz ravni"; u suprotnom, vektor je usmeren "ka ravni". Važi $\mathbf{u} \times \mathbf{v} = -\mathbf{v} \times \mathbf{u}$.

Računanje vektorskog proizvoda

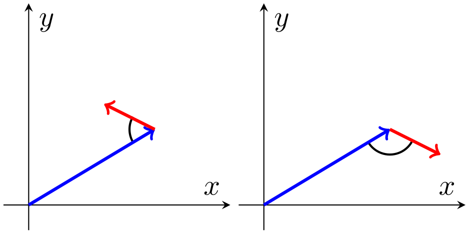
- Namera nam je da napišemo $\mathbf{u} \times \mathbf{v}$ u obliku $\lambda \mathbf{e_z}$, gde ćemo uvesti oznaku $\lambda = \mathrm{VP}\left(\mathbf{u}, \mathbf{v}\right)$.
- Ispostavlja se da imamo fin izraz za $\mathrm{VP}\left(\mathbf{u}, \mathbf{v}\right)$. Ako je $\mathbf{u} = (u_x, u_y)$, $\mathbf{v} = (v_x, v_y)$:

$$\mathrm{VP}\left(\mathbf{u}, \mathbf{v}\right) = \begin{vmatrix} u_x & u_y \\ v_x & v_y \end{vmatrix} = u_x v_y - u_y v_x$$

- Primetimo šta *nemamo*: deljenje, trigonometrijske funkcije, kvadratne korene…

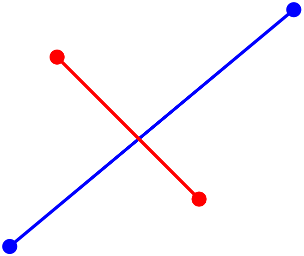
Orijentacija

- Na osnovu prethodne priče o orijentaciji vidimo da je $\mathrm{VP}(\mathbf{u}, \mathbf{v})$ jednako $\pm$ površina paralelograma, gde je znak $+$ u slučaju matematički pozitivnog smera (CCW), a $-$ u suprotnom (CW).
- Ovo možemo da koristimo da bismo utvrdili o kakvom je "zaokretu" reč...

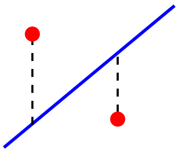


Primer za uvod

- Za date tačke A, B, C, D ispitati da li se duži AB i CD seku.
- Primetimo da se ne traži od nas *gde* se seku!
- Koje je prvo rešenje koje je meni palo na pamet?

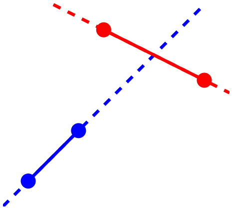


Primetimo...



- Probajmo da ispitamo da li se tačke C i D nalaze sa raznih strana *prave* AB.
- Ovo nije mnogo teško ukoliko je jednačina prave AB data u eksplicitnom obliku $y = kx + n$. Kako određujemo k i n?
- Samo je neophodno da "rastojanja" tačaka od ove prave budu različitog znaka, odnosno:
$$(kc_x - c_y) \cdot (kd_x - d_y) \leqslant 0$$
- Šta je ovde problem?

Ali!



- U stvari je dovoljno da posmatramo $\mathbf{u} = \overrightarrow{AB}$, $\mathbf{v} = \overrightarrow{AC}$ i $\mathbf{w} = \overrightarrow{AD}$!
- Jasno je da je neophodno da budu suprotne orijentacije $\sphericalangle(\mathbf{u}, \mathbf{v})$ i $\sphericalangle(\mathbf{u}, \mathbf{w})$, odnosno:

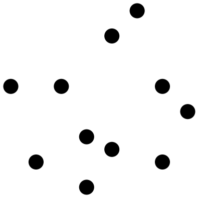
$$\mathrm{VP}(\mathbf{u}, \mathbf{v}) \cdot \mathrm{VP}(\mathbf{u}, \mathbf{w}) \leqslant 0$$

- Ipak, na nešto smo zaboravili!

Specijalni slučajevi!

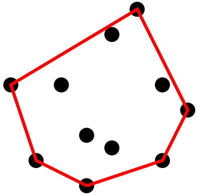
- Često moramo da obratimo pažnju na neke "neprijatne" specijalne slučajeve.
- Nekada detektovanje istih i odgovarajuća obrada zahteva više pažnje i obrade od regularnog toka algoritma!
- Za sada deluje u redu, ali postoje neke situacije kada ovi specijalni slučajeva dovode do dodatnih problema zbog računarske aritmetike.

Definicija konveksnog omotača



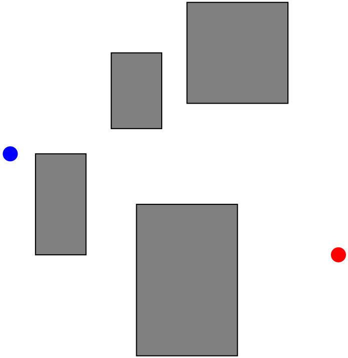
- Za dati skup tačaka S biramo konveksan mnogougao koji sadrži sve tačke iz S na ivicama ili u unutrašnjosti.
- Ima neka interesantna svojstva u vezi sa najudaljenijim parom tačaka.
- Navešćemo dva algoritma i analizirati im složenost. Ipak, postoje i bolji algoritmi od onih koje ćemo obraditi!

Definicija konveksnog omotača

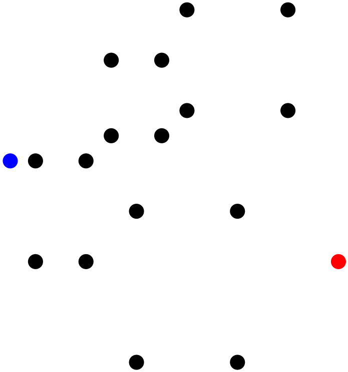


- Za dati skup tačaka S biramo konveksan mnogougao koji sadrži sve tačke iz S na ivicama ili u unutrašnjosti.
- Ima neka interesantna svojstva u vezi sa najudaljenijim parom tačaka.
- Navešćemo dva algoritma i analizirati im složenost. Ipak, postoje i bolji algoritmi od onih koje ćemo obraditi!

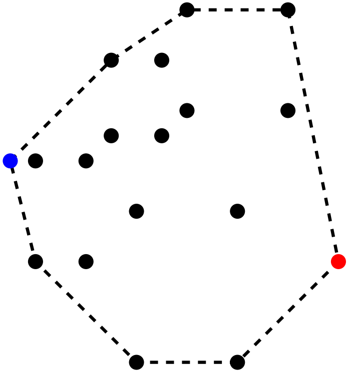
Navigacija robotića



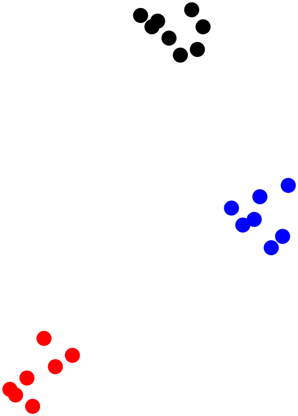
Navigacija robotića



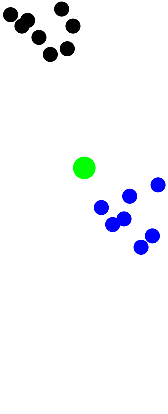
Navigacija robotića



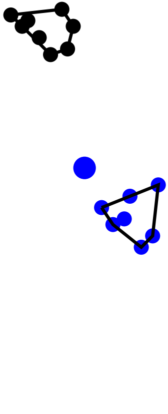
Pokušaj "klasifikacije"



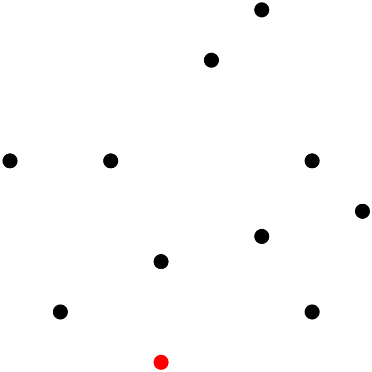
Pokušaj "klasifikacije"



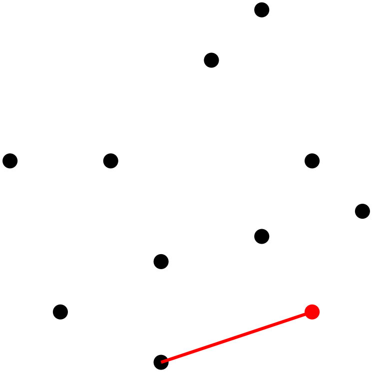
Pokušaj "klasifikacije"



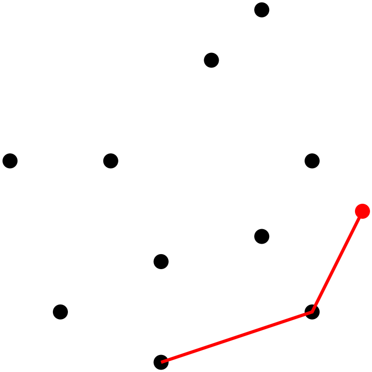
Algoritam Džarvisovog marša



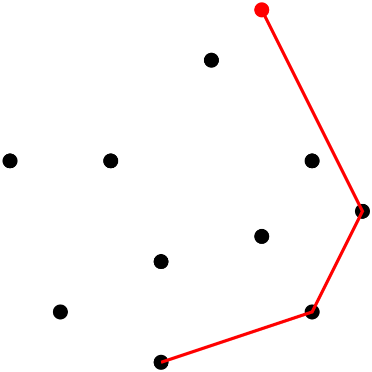
Algoritam Džarvisovog marša



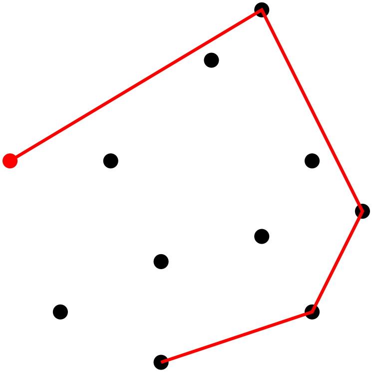
Algoritam Džarvisovog marša



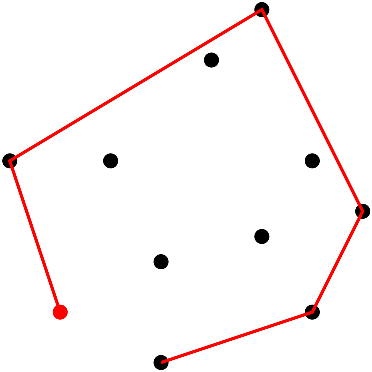
Algoritam Džarvisovog marša



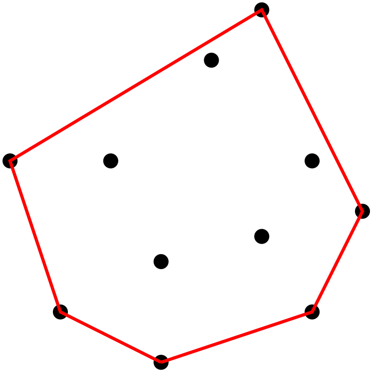
Algoritam Džarvisovog marša



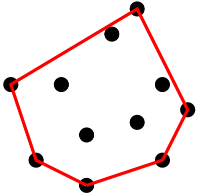
Algoritam Džarvisovog marša



Algoritam Džarvisovog marša



Algoritam Džarvisovog marša



- Početi od tačke iz S sa najmanjom ordinatom (u slučaju više takvih tačaka biramo onu sa najmanjom apscisom).
- Onda "jurimo" sledeću tačku, nakon čega se bira ona nakon nje itd. dok ne dođemo do početne tačke opet.

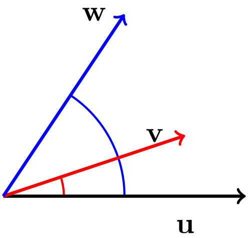
Odabir naredne tačke na omotaču

- Dve faze algoritma: put od minimalnog y do maksimalnog y; put od maksimalnog y do minimalnog y.
- U prvoj fazi tražimo najmanji mogući ugao zaklopljen sa vektorom $\mathbf{e_x}$.
- U drugoj fazi tražimo najmanji mogući ugao zaklopljen sa vektorom $-\mathbf{e_x}$.
- Ne želimo da poredimo uglove preko trigonometrije, odnosno inverznih trigonometrijskih funkcija…

Poređenje uglova

- Želimo da uporedimo $\alpha = \sphericalangle(\mathbf{u}, \mathbf{v})$ i $\beta = \sphericalangle(\mathbf{u}, \mathbf{w})$, pri čemu $0 \leqslant \alpha, \beta < 180°$.
- Tada je $\beta > \alpha$ ako i samo ako su $\sphericalangle(\mathbf{w}, \mathbf{u})$ i $\sphericalangle(\mathbf{w}, \mathbf{v})$ isto orijentisani $\Longrightarrow$ vektorski proizvod!

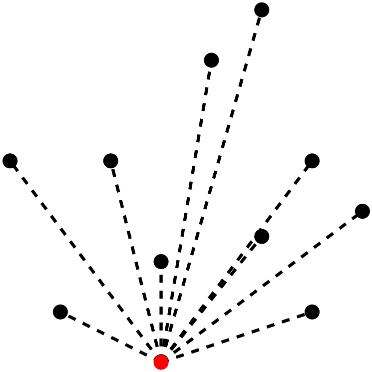
$$\mathrm{VP}(\mathbf{w}, \mathbf{u})\, \mathrm{VP}(\mathbf{w}, \mathbf{v}) < 0$$



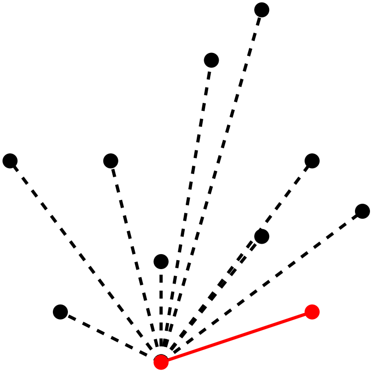
Analiza Džarvisovog marša

- Asimptotska složenost je $O\left(nh\right)$, gde je n broj tačaka u S, a h broj tačaka na konveksnom omotaču (za svaku tačku sa konveksnog omotača moramo da "prođemo" kroz ceo S).
- Odličan algoritam ukoliko znamo da će h biti malo.
- Međutim, ako je $h = n$, dobijemo $O\left(n^2\right)$!
- Koji su nam specijalni slučajevi?

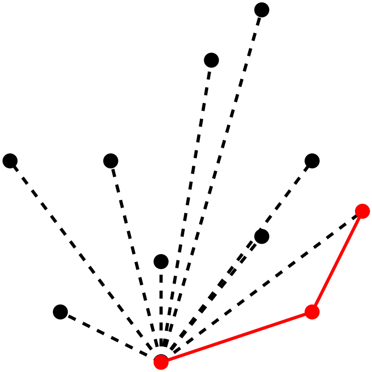
Algoritam Grahamovog skena



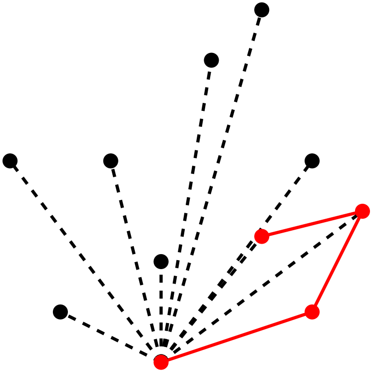
Algoritam Grahamovog skena



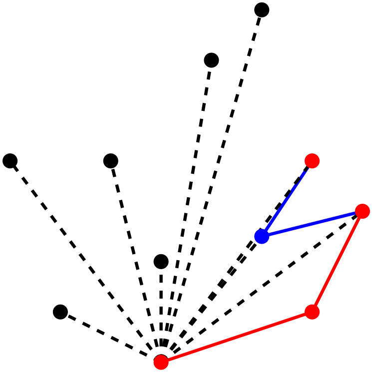
Algoritam Grahamovog skena



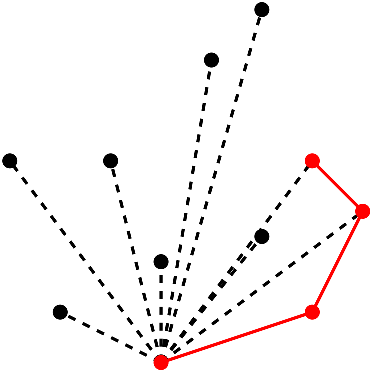
Algoritam Grahamovog skena



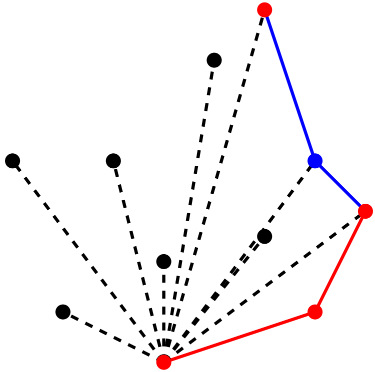
Algoritam Grahamovog skena



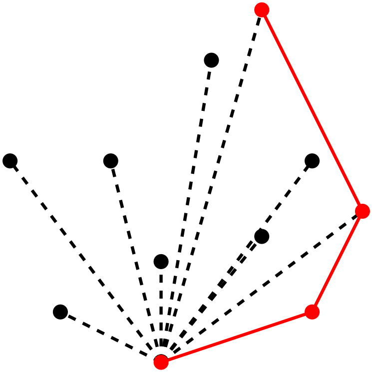
Algoritam Grahamovog skena



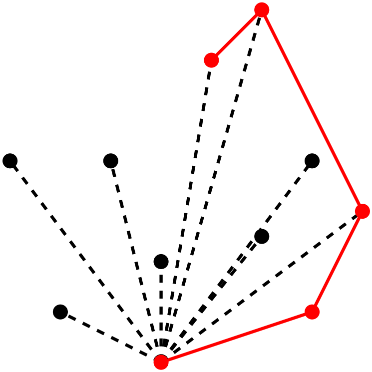
Algoritam Grahamovog skena



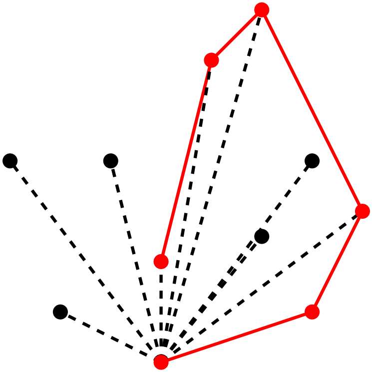
Algoritam Grahamovog skena



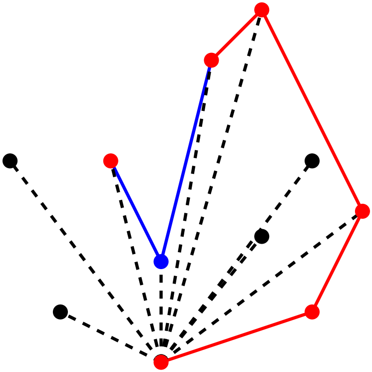
Algoritam Grahamovog skena



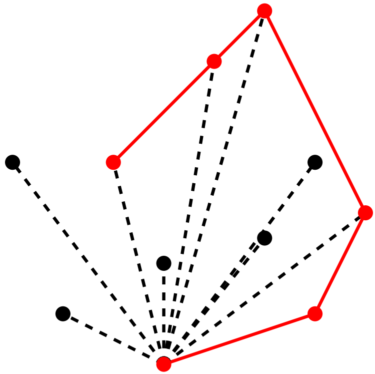
Algoritam Grahamovog skena



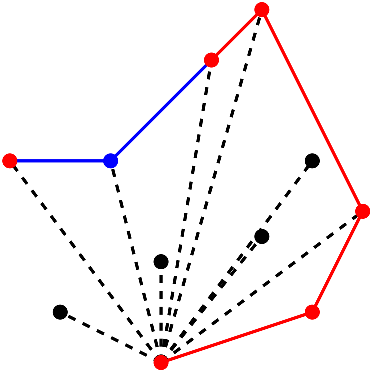
Algoritam Grahamovog skena



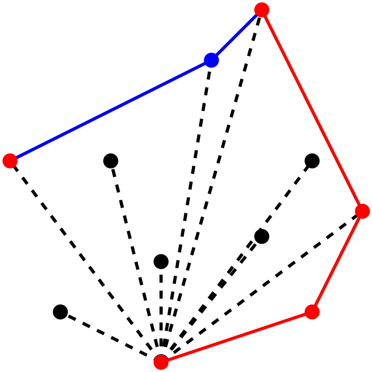
Algoritam Grahamovog skena



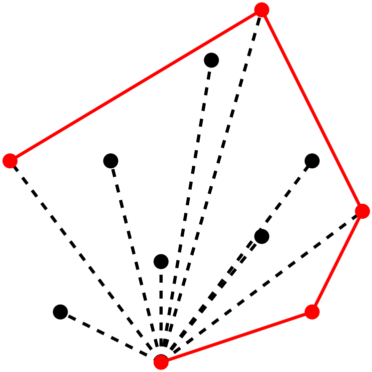
Algoritam Grahamovog skena



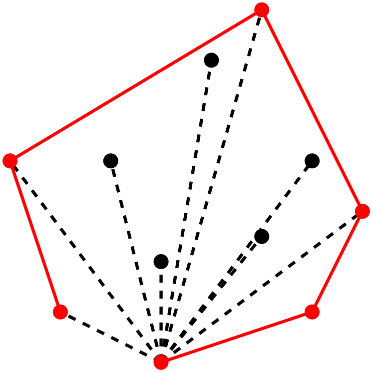
Algoritam Grahamovog skena



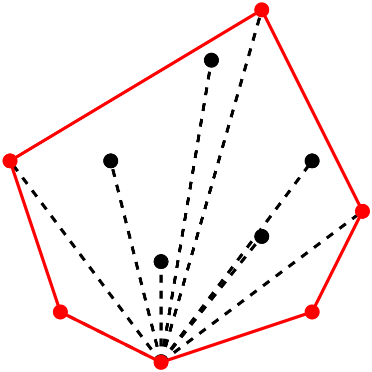
Algoritam Grahamovog skena



Algoritam Grahamovog skena



Algoritam Grahamovog skena

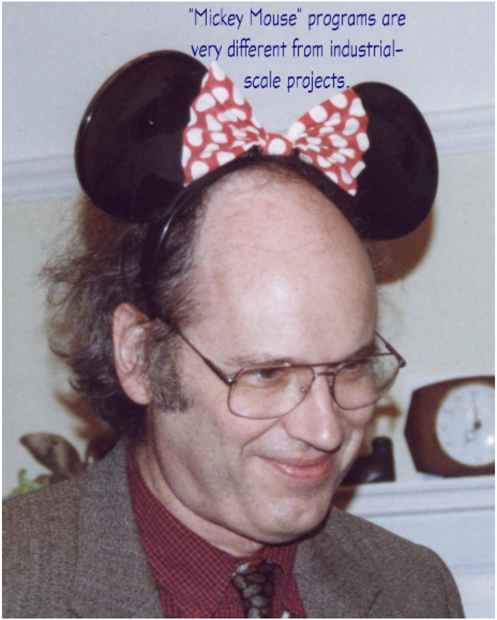


Opis Grahamovog skena

1. Najpre odaberemo neku tačku koja će sigurno biti na omotaču. Takva je, na primer, tačka sa najmanjim y, nazovimo je O.
2. Sve preostale tačke sortiramo po uglu koji radijus vektor (u odnosu na O) zaklapa sa $\mathbf{e_x}$.
3. Tačku O stavljamo da bude jedini element *steka* $\mathcal{H}$.
4. Za svaku tačku A iz S, pri čemu ih razmatramo u sortiranom poretku:
    - ako je dodavanjem ove tačke napravljen "levi zaokret", dodajemo ovu tačku u $\mathcal{H}$;
    - u suprotnom, brišemo tačke sa vrha $\mathcal{H}$ dokle god ne dođemo do "desnog zaokreta" – tada se dodaje A u $\mathcal{H}$.
5. Na kraju dodati tačku O.

Analiza Grahamovog skena

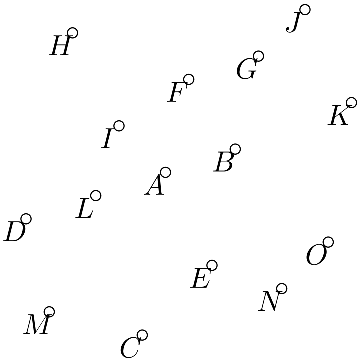
- Ne moramo da računamo ugao, dovoljno je da samo poredimo u toku sortiranja – vektorski proizvod!
- Najpre imamo sortiranje, što znamo da uradimo u $O\left(n \log n\right)$ vremenu.
- Svaku tačku iz S dodajemo u $\mathcal{H}$ jednom i izbacujemo najviše jednom. Stoga je vremenska složenost drugog dela algoritma $O\left(n\right)$.
- Dominira sortiranje: $O\left(n \log n\right)$.
- Postoje algoritmi koji su **asimptotski optimalni**: $O\left(n \log h\right)$.


"Mickey Mouse" programs are
very different from industrial-
scale projects.

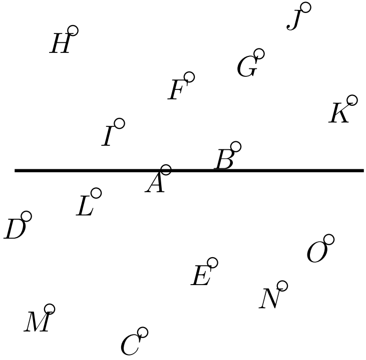
!

Sada pratite sve ovo na tabli!

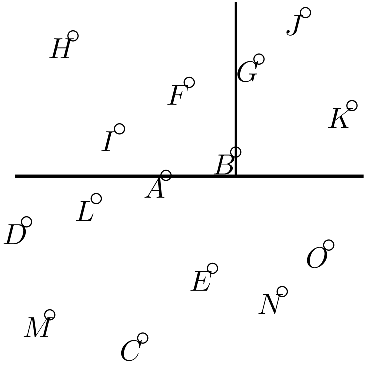
Izgradnja kd stabla



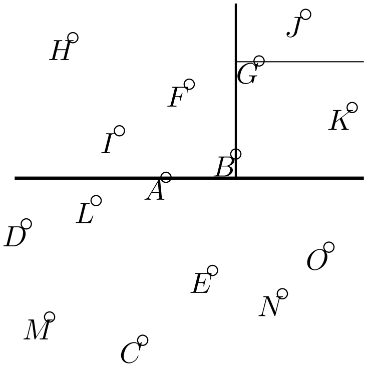
Izgradnja kd stabla



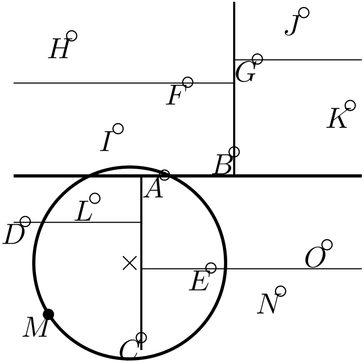
Izgradnja kd stabla



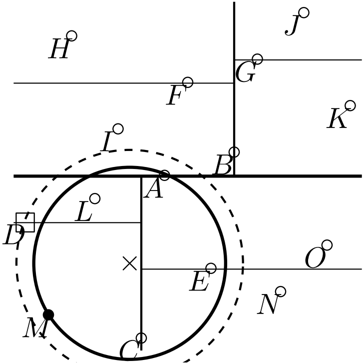
Izgradnja kd stabla



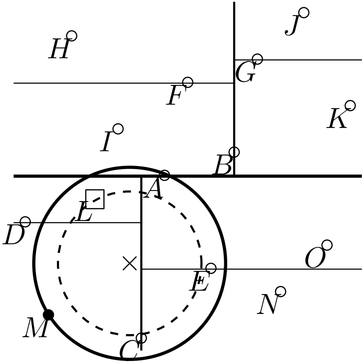
Najbliži sused



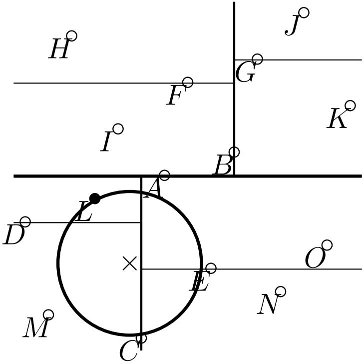
Najbliži sused



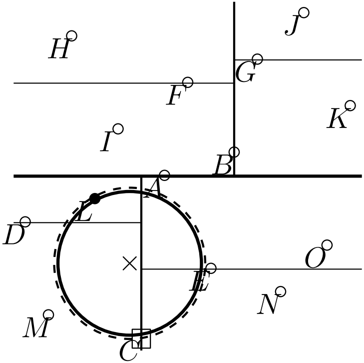
Najbliži sused



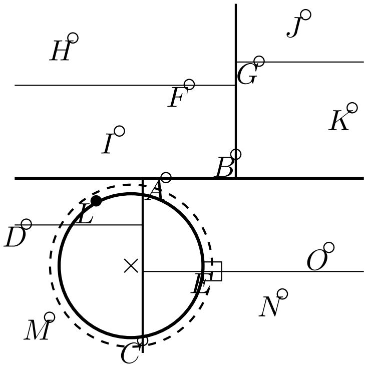
Najbliži sused



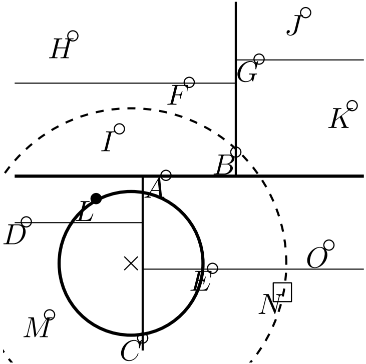
Najbliži sused



Najbliži sused



Najbliži sused



Najbliži sused