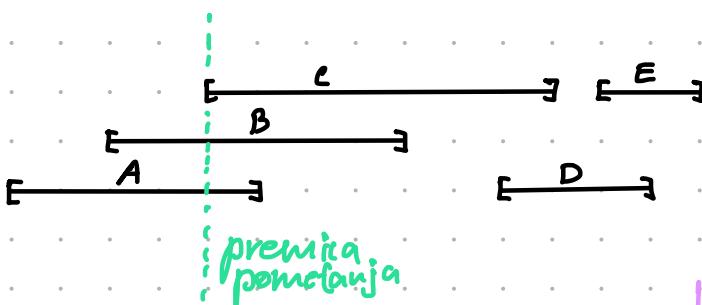


ALGORITMI POMETANJA

Presečišča intervalov



$$M = \{A, B, C\}$$

Stanje: intervali, ki jih seka premica pometanja

Dogodki: krajišča intervalov

- levo krajišče intervala I
Doda presečišče (I, J) v M
Doda I v M
- desno krajišče
Izbriši I iz M

naštevanje:

$$\mathcal{O}(n \log n + k)$$

L št. presečišč
k št. intervalov

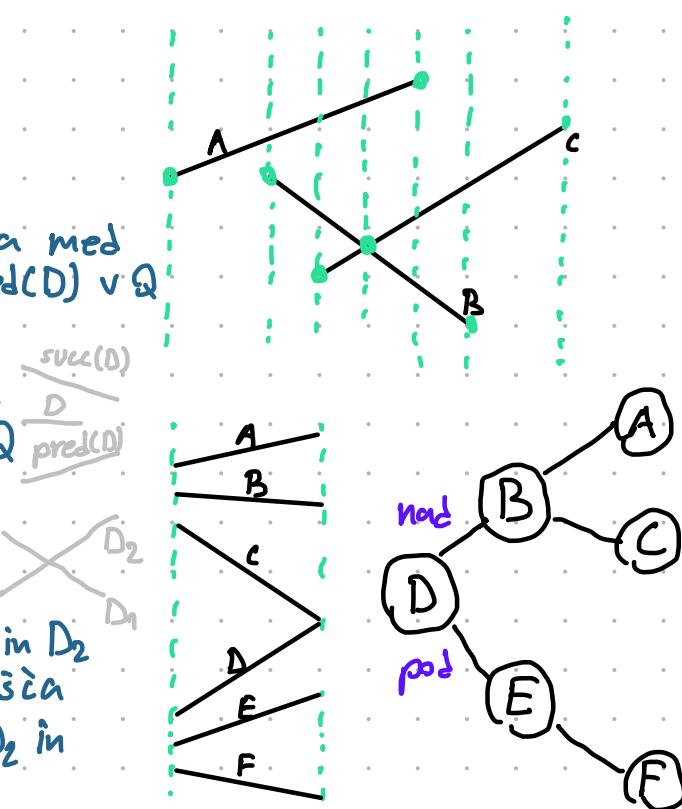
stetijska detekcija

$$\mathcal{O}(n \log n)$$

Presečišča doljic

Dogodki: (prioritetna vrsta Q)

- levo krajišče doljice D
 - Dodaj D v stanje M
 - Dodaj morebitna presečišča med D in $\text{succ}(D)$ ter D in $\text{pred}(D)$ v Q
- desno krajišče doljice D
 - Dodaj morebitna presečišča med $\text{succ}(D)$ in $\text{pred}(D)$ v Q
 - Odstrani D iz stanja M
- presečišče med D_1 in D_2
 - Shrani presečišče
 - V stanju M zamenjaj D_1 in D_2
 - Dodaj morebitna presečišča med D_1 in $\text{pred}(D_1)$ ter D_2 in $\text{succ}(D_2)$ v Q



Stanje: (uvravnoteženo binarno drevo)

- hrani doljice ki jih seka premica pometanja urejene po y-koordinati
- omogoča poizvedbi succ in pred

časovna zahtevnost

$$\mathcal{O}((n+k) \log n)$$

L št. presečišč
k št. premic

prostorska zahtevnost

$$\mathcal{O}(n+k)$$

VEČKOTNIKI

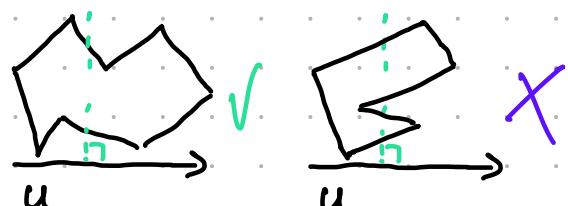
\hat{u} -monotona pot P

Presečisce med P in vsako premico pravokotno na \hat{u} je lahko največ ena točka



\hat{u} -monoton večkotnik P

Presečisce med P in vsako premico $\perp \hat{u}$ je lahko največ ena doljica



Mejo monotonega večkotnika lahko razdelimo na dve monotoni poti

Razpoznavanje večkotnika
Ali se poligonska pot sekla?

$O(n \log n)$

lahko tudi: $O(n)$

Ploščina večkotnika

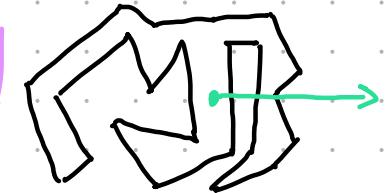
$$\sum_{i=1}^n \frac{1}{2} (x_{i+1} - x_i)(y_{i+1} + y_i)$$

$O(n)$

Ali je točka v večkotniku?

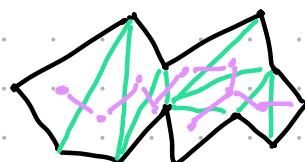
$O(n \log n)$

štejemo presečisa
liho \rightarrow noben sato \rightarrow znam



Triangulacija večkotnika

Razdelitev večkotnika na trikotnike.
Oglišča trikotnikov morajo biti oglisca originalnega večkotnika.

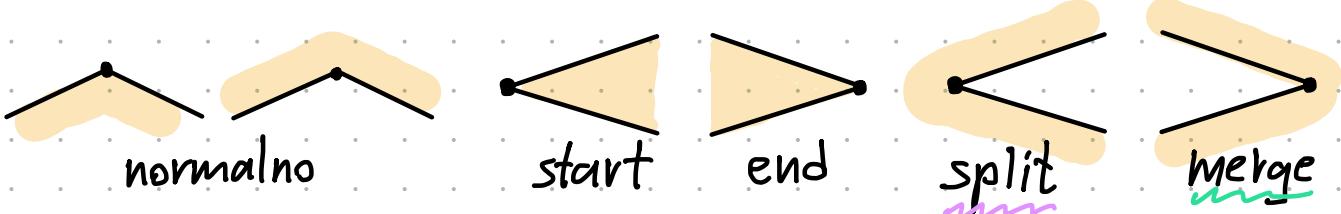


triangulacija
dualni graf
(drevo stopnje
največ 3)

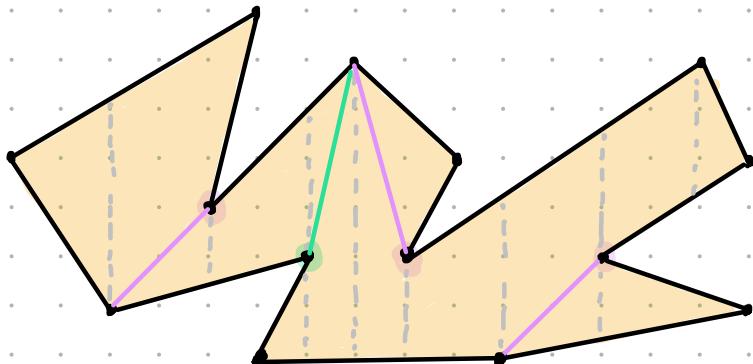
Triangulacija večkotnika z n oglisči

- n oglisči
- $n-3$ diagonali
- $n-2$ trikotnikov

Razdelitev večkotnika na monotone večkotnike



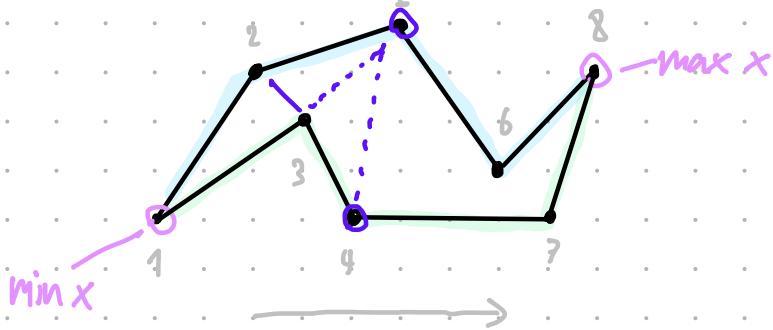
Večkotnik je (x)-monoton \iff nima nobenega oglisca split ali merge



- večkotnik razdelimo na trapeze
- poisčemo split in merge oglisca in dodamo diagonale do drugega oglisca v trapezu

$O(n \log n)$

Triangulacija monotonega večkotnika



Oglisca razvrstimo po x -koord.
En konec elastike je na zgornji
drugi pa na spodnji polovici.
Na vsakem koraku dodamo
diagonale (elastika).

$O(n)$

KONVEKSNOST

$C \subseteq \mathbb{R}^d$ konveksna $\iff \forall p, q \in C \forall \lambda \in [0, 1]: (1-\lambda)p + \lambda q \in C$

konveksna ovojnica $S \subseteq \mathbb{R}^d$

convex hull

$$CH(S) = \bigcap_{\substack{T \subseteq \mathbb{R}^d \\ T \text{ konv.} \\ S \subseteq T}} T \quad \cong$$

$$CH(S) = \left\{ x \in \mathbb{R}^d \mid x = \sum_{i=1}^n \lambda_i s_i, \right. \\ \left. \sum_{i=1}^n \lambda_i = 1, s_i \in S, \lambda_i > 0 \right\}$$

dovolj je $n=d+1$
caratheodory

Izrek: Presek konveksnih množic je konveksen.

Janis' march

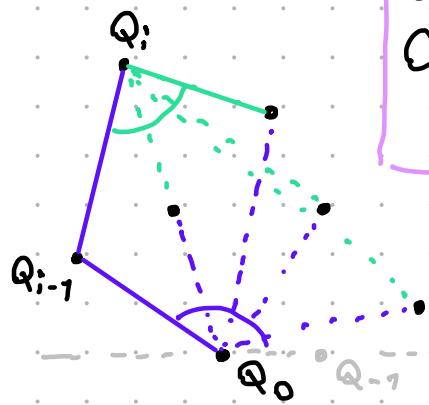
Algoritam zavijanje darila

$Q_0 \leftarrow$ najnižja točka

$$Q_{-1} = Q_0 + (1, 0)$$

ponavljam dokler ne prideš okrog

$$Q_{i+1} \leftarrow \text{točka } p, \text{ ki minimizira} \\ \angle(Q_{i-1} - Q_i, p - Q_i)$$



output-sensitive
 $O(nh)$

$$\lceil \text{CH}(s) \rceil \\ |s|$$

Grahamov algoritem

- poiščemo najnižjo točko
- sestavimo večkotnik tako da točke uredimo po kotu
- sprehodimo se po večkotniku in brišemo ne ekstremne točke lejer naredimo zasuk v levo

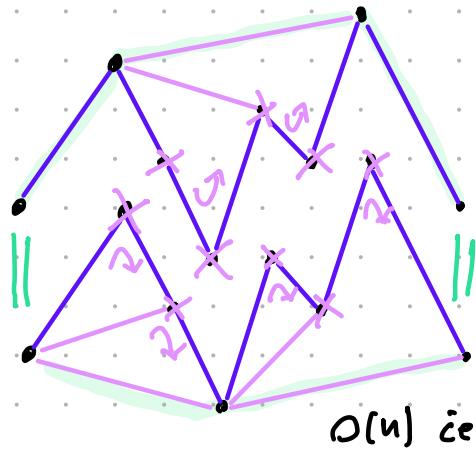
$O(n)$
 $O(n \log n)$

$O(n)$
 $\min y$

$O(n \log n)$

Inkrementalni algoritem privrstni

- točke uredimo po x-koordinati
- posebej izračunamo zgornjo in spodnjo polovico
- dodajamo točke od leve proti desni. Če naredimo zavoj \nearrow (zgoraj) / \searrow (spodaj) odstranimo točke do prejšnje maksimalne/minimale



$O(n \log h)$

$O(n)$ če so že urejene

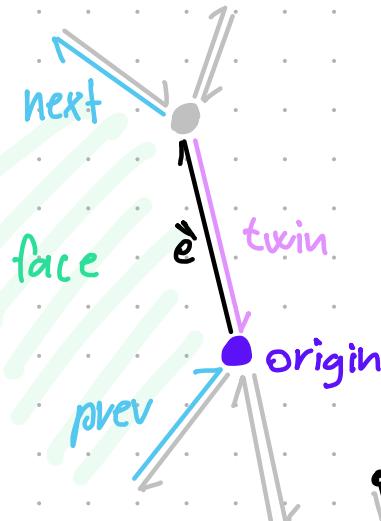
DCEL - doubly connected edge list

Hčemo opisati delitev ravni na vozlišča, povezave in lica

Half Edge

- origin : Vertex
- twin : Half Edge
twin je vedno obrnjen v drugo smr
- face : Face
- next : Half Edge
- prev : Half Edge

twin je vedno obrnjen v drugo smr

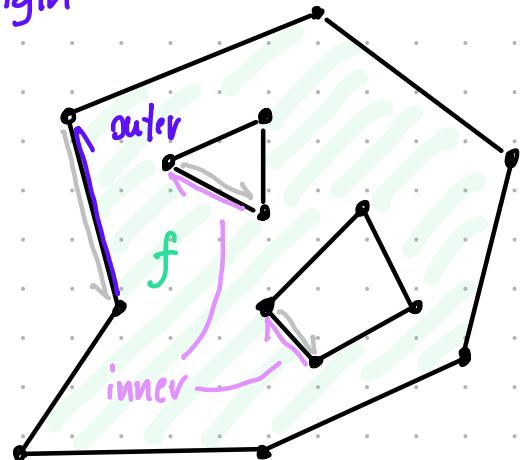


Vertex

- coordinates
- edge : Half Edge
enam izmed izhodnih povezav

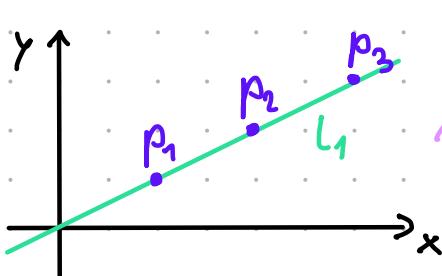
Face

- outer-component : HalfEdge
enam izmed povezav na zunanjem ciklu
- inner-components : HalfEdge[]
seznam povezav na notranjih ciklilih



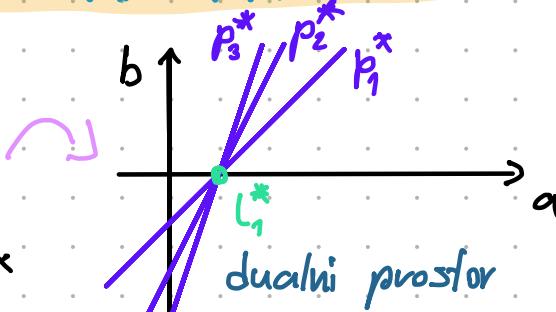
Vsa lica (razen zunajje) imajo outer-component.
inner-components imata lahko 0 ali več komponent.

DUALNOST TOČKA-PREMICA



$$L: y = L_a x - L_b$$

$$p: (p_x, p_y)$$



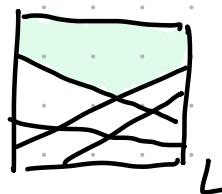
$$L^*: (L_a, L_b)$$

$$p^*: b = p_x a - p_y$$

Lastnosti

- incidence preserving
- p_1, \dots, p_n kolinearne $\iff p_1^*, \dots, p_n^*$ sedajo v isti točki
- p leži pod $L \iff L^*$ leži pod p^*

Zgorja ovojnica
upper envelope



$$p \in L \iff L^* \in p^*$$

p_1^*, \dots, p_n^* sedajo v isti točki



dualno spodnjemu delu $CH(L^*)$
označi: $LH(L^*)$

RAZPOREDITVE

ARRANGEMENTS

L ... množica n premic

\hookrightarrow splošen položaj: ni vzporednic, ni presečišč 3 premic

$A(L)$



$$|V| = \binom{n}{2}$$

$$|E| = n^2$$

$$|F| = \binom{n}{2} + n + 1$$

$\text{zone}(L)$ množica lici/celic, ki sekajo L

inkrementalna konstrukcija $A(L)$ kot DCEL

$O(n^2)$

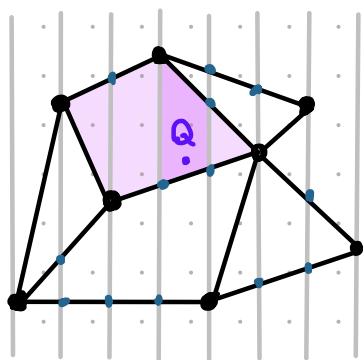
DOLOČANJE POLOŽAJA

Radi bi hitro ugotovili v kateri celici subdivizije ravnine, leži dana točka Q .

množica vozlišč povezav in lici, ki se ne sekajo



► Naivna rešitev



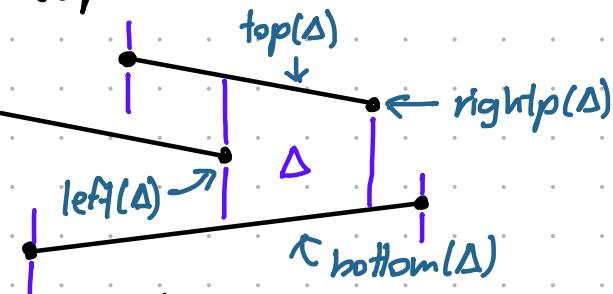
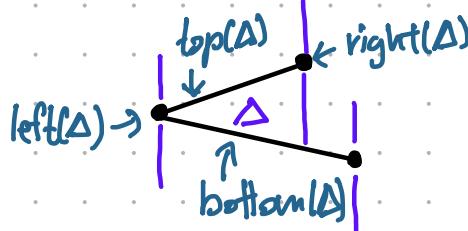
$Q.x \rightarrow$ bisekcija \rightarrow pas

$Q.y \rightarrow$ pas \rightarrow izhodno drevo \rightarrow del lica \rightarrow lice

Poizvedba $O(\log n)$ konstrukcija $O(n^2 \log n)$
Prostор $O(n^2)$

► Vertical decomposition / trapezoidal decomposition

\hookrightarrow lici so trikotniki ali trapezoidi



- Vsaka lica hrani kazalce do svojih sosedov
- Vsaka povezava hrani svoji vozlišči in kazalec na originalno lice ki je nad njo.

Vse točke zajamemo v ac. pravokotnik R

inkrementalno dodajamo povezave v naključnem vrstnem redu in posodabljammo dekompozicijo T .

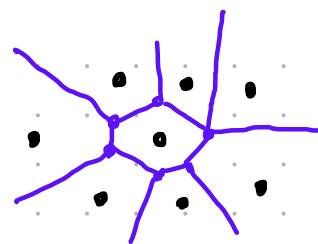
Sproti gradimo izhodno drevo D .



poizvedba $O(\log n)$ pričakovana
konstrukcija $O(n \log n)$
prostор $O(n)$

VORONOIJEV DIAGRAM

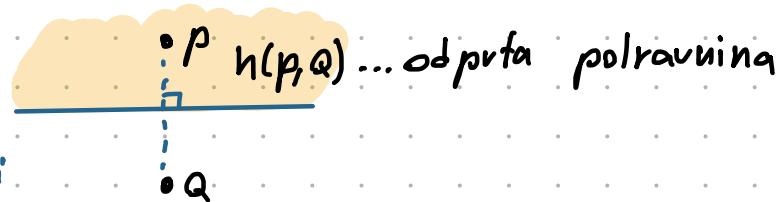
$P = \{p_1, \dots, p_n\}$... množica točk



$\text{Vor}(P)$... Voronojev diagram (množica vozlišč, povezani in lic)

$$V(p_i) = \bigcap_{j \neq i} h(p_i, p_j)$$

↑ Voronojeva celica okoli p_i



Izrek: Za $n \geq 3$ je v Voronojevem diagramu: $|V| \leq 2n - 5$ $|E| \leq 3n - 6$

Fortunes sweep line algorithm: $\mathcal{O}(n \log n)$, $\mathcal{O}(n)$ prostor

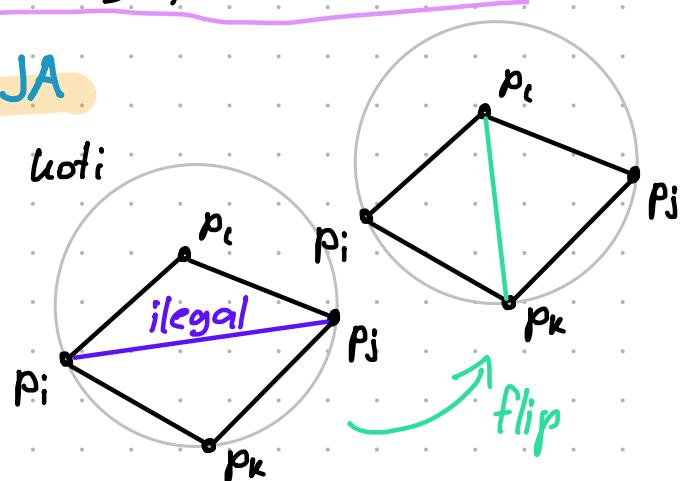
DELAUNAYEVA TRIANGULACIJA

Želimo triangulacijo s čim večjimi koti

Povezava $p_i p_j$ je ilegalna



p_l leži v očrtani krožnici trikotnika $\Delta p_i p_j p_k$



Delaunayev graf je dualni graf (z poravnanimi povezavami) Voronojevega diagrama.

Algoritmi:

- iterativno obračanje povezav
- plane sweep
- randomized incremental construction
- pretvorba iz voronojevega diagrama

$\mathcal{O}(n \log n)$

Euclidian minimum spanning tree - EMST

↪ Graf ki povezuje vse točke iz P in ima minimalno skupno dolžino povezav.

Izrek: Vsaka povezava v EMST je tudi v DT

Algoritem:

- izračunaj Delaunayevu triangulacijo $DT(P)$
- sortiraj povezave po dolžini
- dodajaj povezave dokler ne našteš cikla

$\mathcal{O}(n \log n)$

OBMOČNA DREVEŠA

RANGE TREES

Iznamo točke $P = \{p_1, \dots, p_n\} \subseteq \mathbb{R}^d$

Želimo odgovoriti na vprašanje:

Katerih točk so v $[a_1, b_1] \times [a_2, b_2] \times \dots \times [a_d, b_d]$?

one tree

one tree for first dimension, many next

	KD-tree	range tree	
konstrukcija	$O(n \log n)$	$O(n \log n)$	k output size
poizvedba	$O(k + \sqrt{n})$	$O(k + \log^d n)$	
prostор	$O(n)$	$O(n \log n)$	

1D primer

- Učinkovito binarno iskalno drevo
z spremembami: ključe hranimo samo v listih
vozlišča hranijo maksimum levega poddrevesa

poizvedba: $[6, 21]$

