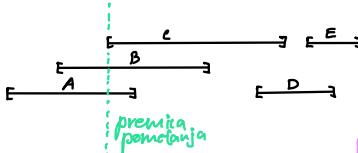


ALGORITMI POMETANJA

Presečišča intervalov



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

Stanje: intervali, ki jih selita premica pometanja

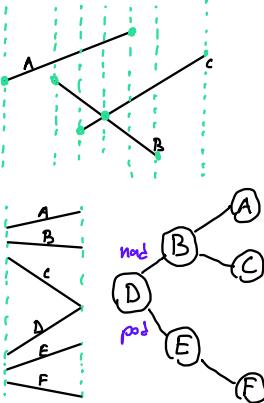
- Dogodki: krajšča intervalov
- levo krajšče intervala I
Dobji presečišče $(I, J) \forall J \in M$
Dobji I v M
 - desno krajšče
Izbriši I iz M

naštevanje:
 $O(n \log n + k)$
L st. presečišč
stetje, detekcija
 $O(n \log n)$

Presečišča daljic

Dogodki: (prioritetna vrsta Q)

- levo krajšče daljice D
 - Dodaj D v stanje M
 - Dodaj morebitna presečišča med D in succ(D) ter D in pred(D) v Q
- desno krajšče daljice D
 - Dodaj morebitna presečišča med succ(D) in pred(D) v Q
 - Odstrani D iz stanja M
- presečišče med D₁ in D₂
 - Shrani presečišče
 - V stanje M zavzemajo D₁ in D₂
 - Dodaj morebitna presečišča med D₁ in pred(D₁) ter D₂ in succ(D₂) v Q



Stanje: (urejanotoženo binarno drevo)

- hraniti daljice ki jih selita premica pometanja urejene po y-koordinanti
- omogoča poizvedbo succ in pred

časovna zahtevnost
 $O((nk) \log n)$
L st. presečišč
prostovska zahtevnost
 $O(n+k)$

KONVEKSNOST

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

Konveksna obujmica $S \subseteq \mathbb{R}^d$

$$CH(S) = \bigcap_{\substack{T \subseteq S \\ T \text{ konv.}}} T \quad \cong \quad CH(S) = \left\{ x \in \mathbb{R}^d \mid x = \sum_{i=1}^n \lambda_i s_i, \sum_{i=1}^n \lambda_i = 1, s_i \in S, \lambda_i \geq 0 \right\}$$

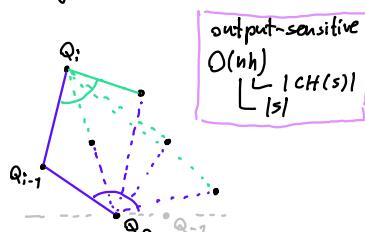
convex hull
dovolj je $n=d+1$
caratheory

Izrek: Presek konveksnih množic je konveksen.

Joinis' march

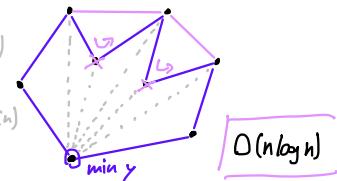
Algoritam zavijanje darila

- $Q_0 \leftarrow$ najnižja točka
 $Q_1 = Q_0 + (1, 0)$
 ponavljajoč dokler ne pride do okrog
 $Q_{i+1} \leftarrow$ točka p, ki minimizira
 $\nexists (Q_{i-1} - Q_i, p - Q_i)$



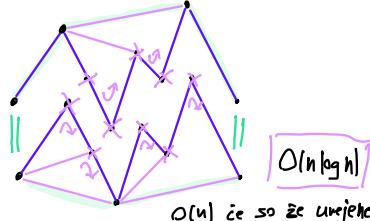
Grahamov algoritem

- poisciemo najnižjo točko
- ustvarimo včokotnik tako da točke uredimo po kotu
- sprehodimo se po včokotniku in brisemo ne ekstremsne točke lejer naredimo zasuk v levo



Inkrementalni algoritem

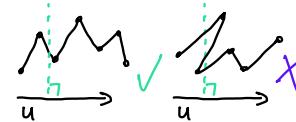
- privrstni
- točke uredimo po x-koordinati
 - posebej izračunamo zgornjo in spodnjo polovico
 - dodajamo točke od leve proti desni. Če naredimo zavoj \nearrow (zgoraj) / \nwarrow (spodaj) odstranimo točke do prejšnje maksimalne/minimalne



VEČKOTNIKI

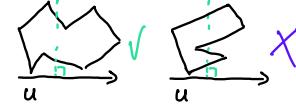
ū-monoton pot P

Presečišča med P in vsodo premico pravokotno na u je lahko največ ena točka



ū-monoton večkotnik P

Presečišča med P in vsodo premico L u je lahko največ ena doljica



Mojo monotonega večkotnika lahko razdelimo na dve monotoni poti

Razpoznavanje večkotnika
Ali se poligona pot sekajo?

Plosčina 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?
3ječemo presečišča
liho \rightarrow noben sedež \rightarrow zunaj



triangulacija
dualni graf
(drevo stopnje največ 3)

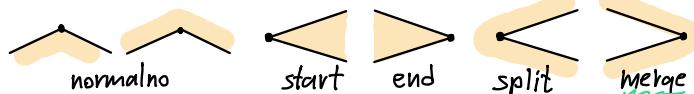
Triangulacija večkotnika

Razdelitev večkotnika na trikotnike
Oglizca trikotnikov morajo biti oglizca originalnega večkotnika.

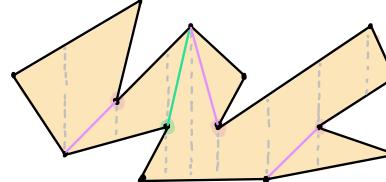
Triangulacija večkotnika z n oglizci

- n oglizci
- $n-3$ diagonal
- $n-2$ trikotnikov

Razdelitev večkotnika na monotone večkotnike



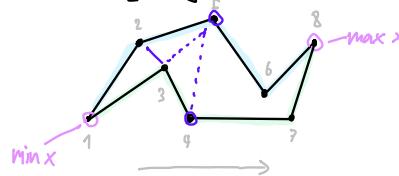
Večkotnik je (x)-monoton \iff nina voglena oglišča split ali merge



- večkotnik razdelimo na trapeze
- poiščemo split in merge oglizca in dodamo diagonalne do drugega oglizca v trapezu

$$O(n \log n)$$

Triangulacija monotonega večkotnika



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

$$O(n)$$

OSTALO

krovni izrek

$$T(n) = aT\left(\frac{n}{b}\right) + O(n^c) = \begin{cases} O(n^c) & a < b^c \\ O(n^{\log_b a}) & a = b^c \\ O(n^{\log_b a}) & a > b^c \end{cases}$$

O-notacija

Notation	Description
$f(n) = o(g(n))$	f is dominated by g asymptotically $\forall k > 0 \exists n_0 \forall n > n_0: f(n) \leq k g(n)$ $\lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} = 0$
$f(n) = O(g(n))$	$ f $ is asymptotically bounded above by g $\exists k > 0 \exists n_0 \forall n > n_0: f(n) \leq k g(n)$ $\limsup_{n \rightarrow \infty} \frac{f(n)}{g(n)} < \infty$
$f(n) = \Theta(g(n))$	f is asymptotically bounded by g both above and below $\exists k_1 > 0 \exists k_2 > 0 \exists n_0 \forall n > n_0: k_1 g(n) \leq f(n) \leq k_2 g(n)$ $f(n) = O(g(n))$ and $g(n) = O(f(n))$
$f(n) \sim g(n)$	f is asymptotically equal to g $\forall \epsilon > 0 \exists n_0 \forall n > n_0: \left \frac{f(n)}{g(n)} - 1 \right < \epsilon$ $\lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} = 1$
$f(n) = \Omega(g(n))$	f is bounded below by g asymptotically $\exists k > 0 \exists n_0 \forall n > n_0: f(n) \geq k g(n)$ $\liminf_{n \rightarrow \infty} \frac{f(n)}{g(n)} > 0$
$f(n) = \omega(g(n))$	f dominates g asymptotically $\forall k > 0 \exists n_0 \forall n > n_0: f(n) > k g(n)$ $\lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} = \infty$

DCEL - doubly connected edge list

Hčemo opisati delitev ravnih na vozlišča, povezane in lice

Half Edge

- origin : Vertex
- twin : Half Edge
- twin je vedno obrnjeno v drugo smere
- face : Face
- next : Half Edge
- prev : Half Edge

Vertex

edge

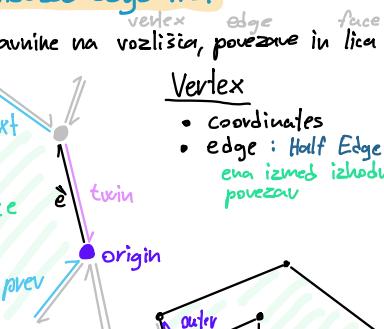
- coordinates
- edge : Half Edge ena izmed izhodnih povezav

face

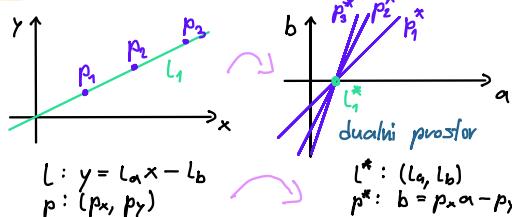
- outer-component : Half Edge ena izmed povezav na zunanjem ciklu

- inner-component : HalfEdge[] seznam povezav na notranjih ciklih

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



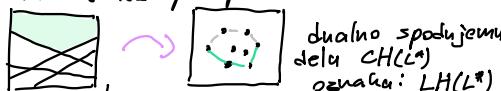
DUALNOST TOČKA-PREMICA



Lastnosti

- incidence preserving: $p \in L \iff L^* \text{ leži pod } p^*$
- p_1, \dots, p_n kolinearne $\iff p_1^*, \dots, p_n^*$ selajo v isti točki
- p leži pod $L \iff L^*$ leži pod p^*

Zgorajna ovojnica
upper envelope



RAZPOREDITVE

ARRANGEMENTS

L ... množica n premic

↳ splošen položaj: ni vzporednic, ni presečišč 3 premic

$A(L)$

↳ povezava

lice

vzlišje

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

$|E| = n^2$

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

zone(L) množica lici/celic, ki sklajo L

inkrementalna konstrukcija $A(L)$ kot DCEL

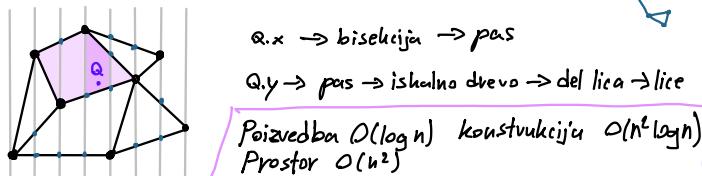
$O(n^2)$

DOLOČANJE POLOŽAJA

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

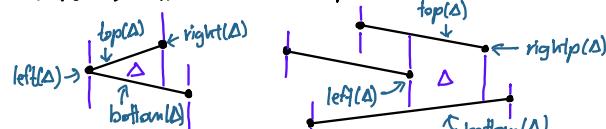
množica vozlišč povezav in lici, ki se ne skljajo

► Naivna rešitev



► Vertical decomposition / trapezoidal decomposition

↳ lici so trikotniki ali trapezoidi



• Vsaka lica hrani kazalce do svojih sosedov

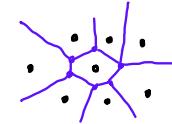
• Vsaka povezava hrani svoji vozlišči in kazalce na originalno lice, ki je nadajo.

Vse točke zajamčeno v aci pravokotnik R
 Inkrementalno dodajamo povezave v naključnem vrstnem redu
 in posodobljamo dekompozicijo T.
 Sproti gradimo iskalno drevo D.

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

VORONOIJEV DIAGRAM

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



$V(p_i)$... Voronoijski diagram (množica vozlišč, povezav in lici)

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

↑ Voronoijski celični sholi: p_i

\cup

q

Izhodi: za $n \geq 3$ je v Voronoijskem diagramu: $|V| \leq 2n - 5$ $|E| \leq 3n - 6$

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

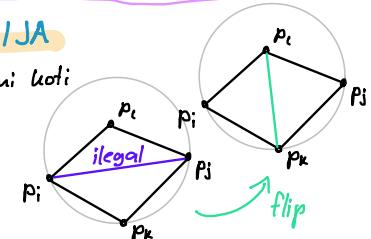
DELAUNAYEVA TRIANGULACIJA

Želimo triangulacijo s čim večjimi koti

Povezava $p_i p_j$ je ilegalna

↔

p_i leži v očrtani krožnici trilateralnega $\Delta p_i p_j p_k$



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

Algoritmi:

- iterativno obračanje povezav
- plane sweep
- randomised incremental construction

$O(n \log n)$

Euclidean minimum spanning tree - EMST

↳ Graf ki povezuje vse točke iz P in ima minimalno skupino dolžino povezav.

Izhodi: vsaka povezava v EMST je tudi v DT

Algoritmi:

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

$O(n \log n)$

OBMOČNA DREVESA

RANGE TREES

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

Želimo odgovoriti na vprašanje:

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

	one tree	ove tree for first dimension, many next
konstrukcija	KD-tree $O(n \log n)$	range tree $O(n \log n)$
poizvedba	$O(k + \sqrt{n})$	$O(k + \log n)$
prostor	$O(n)$	$O(n \log n)$

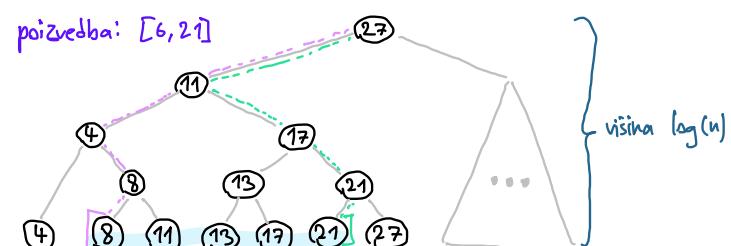
k output size

1D primer

• uvajanje binarno iskalno drevo

z spremembami: ključe hranimo samo v listah vozlišča hranijo maksimum levega poddrveza

poizvedba: $[6, 21]$



višina $\log(n)$