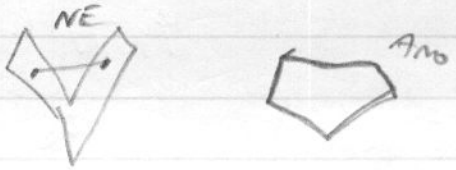
GEOMETRICKÉ ALGORITMY 20. 9. 2007

KONVEXNÍ MNOŽINA
$M \subset \mathbb{R}^2$
$p, q \in M \Rightarrow pq$ úsečka leží v M

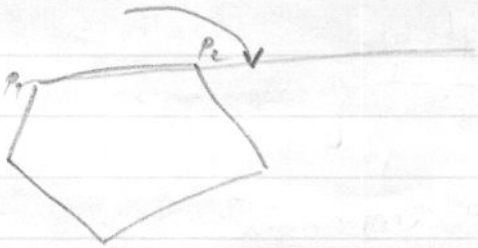


KONVEXNÍ OBAL MNOŽINY M
$CH(M)$ = nejmenší konv. množina

$$CH(M) = \bigcap_{L \supseteq M} K = \bigcap_{P \supseteq M} P$$
P je polorovina

$M = \{p_1 \dots p_n\}$ – konvexní obal je mnohoúhelník, jehož vrcholy leží v množině M

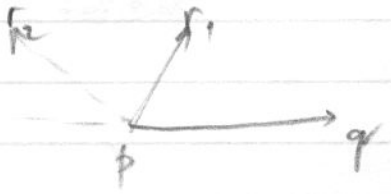
$$CH(M) = \bigcap_{M \subset P_{ij}} P_{ij}$$



P_{ij} je na hraně konv. obalu jestliže žádný bod množiny M neleží vlevo od ní

Algoritmus
I: n bodů v rovině
O: vg vrcholy konv. obalu ve směru hod. ručičky

Původní algoritmus pro KO
množina E – úsečky orientované z hranice $CH(M)$
1. $E \leftarrow \emptyset$
2. for all $(p,q) \in M \times M$, $p \neq q$
   for all $r \in M - \{p,q\}$ zjistíme, zda r leží vlevo od $\vec{pq}$
   if žádný then $(p,q) \rightarrow E$
orientované úsečky v E uspořádáváme



$q-p \quad \& \quad r-p$

$$\begin{vmatrix} q_x - p_x & r_x - p_x \\ q_y - p_y & r_y - p_y \end{vmatrix}$$

pokud r nad $det > 0$

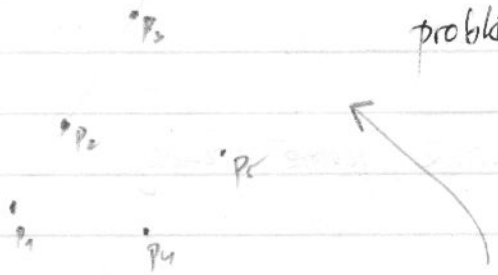
na přímce $det = 0$

$det < 0$

Časová náročnost:

$(n-2) \cdot (n^2 - n) \sim [n^3] \; \mathcal{O}(n^3)$

Degenerovaný vstup

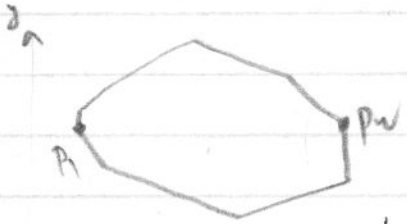


problém při uspořádávacím množiny E

Změna: leží všechny body striktně vpravo nebo uvnitř všečky pq?

chyby při zaokrouhlování: příliš malý determinant $\Rightarrow D = 0$

ALGORITMUS II:



nejlevější bod (nejmižší)
a nejpravější (nejvyšší) bod

L_H Horní konvexní hranice = část hranice konv. obalu nad $p_1 p_n$

L_D Dolní "

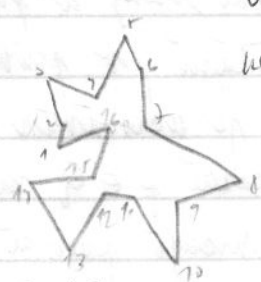
- najdřívě horní k. hrania
- potřebujeme usp. bodů – lexikograficky podle souřadnice x a y
$p < q \Leftrightarrow p_x < q_x$ nebo $p_x = q_x \wedge p_y < q_y$

- přidáváme body a testujeme, zda poslední tři dělají „zatáčku vpravo".

test pomocí znaménka determinantu
náročnost n
uspořádání $n \cdot \log n$

MODIFIKOVANÝ GRAHAM SCAN
1: množina M n bodů
0: seznam konvexních vrcholů obalu clockwise

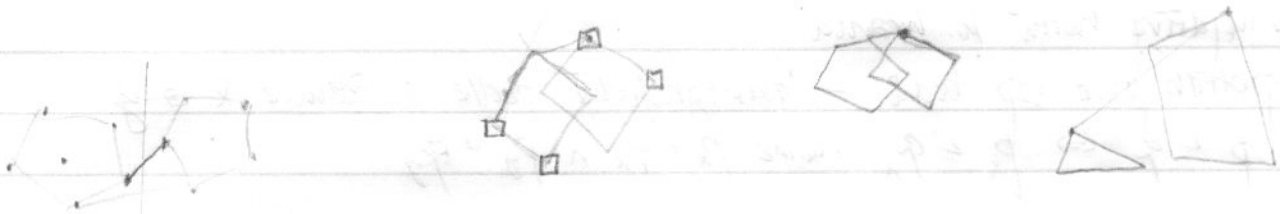
1. uspořádej lexikograficky M
2. $p_1, p_2 \rightarrow L_H$
3. for $i = 3$ to n přidej p_i do L_H
   while L_H obsahuje alespoň 3 body a poslední 3 body nedělají zatáčku vpravo
   → 4. odstraň prostřední předposlední bod
5. $p_n, p_{n-1} \rightarrow L_D$
6. for $i = n-2$ to 1
   p_i přidej do L_D
   while ~~pos~~ L_D obsahuje alespoň 3 body a poslední 3 nedělají zatáčku vpravo
   odstraň předposlední bod
7. zřeď L_D za L_H



uspořádání nejvyšší vrchol

Důkaz korektnosti pomocí indukce

Poznámky
Místo množiny M zadán mnohoúhelník pořadím vrcholů



DÚ: Najít za dvou konvexních mnohoúhelníků jeden společný konvexní obal.

1.

PRŮSEČÍKY ÚSEČEK

n úseček, k průsečíků $\Rightarrow O((n+k)\log n)$

Metoda Sweep line

- strom
- fronta událostí - koncové body úseček

Zametací přímka

- uspořádání bodů v rovině lexikografické - nejdřív podle souřadnice y pak podle x

Fronta událostí Q

- koncové body a průsečíky setříděné podle předchozího ~~to~~ uspořádání, v průběhu algoritmu se mění. Každý bod v Q je uchováván společně s úsečkami, na kterých leží.
- pro každé $p \in Q$ máme vyvážený bin. strom $T(p)$, jehož listy určují pořadí úseček, které pamatá zametací přímka v poloze kousek pod p. Vnitřní listy "hlavní" jsou označeny

kořen stromu je nejpravější list podstromu

podle nejpravějšího listu pravého levého podstromu

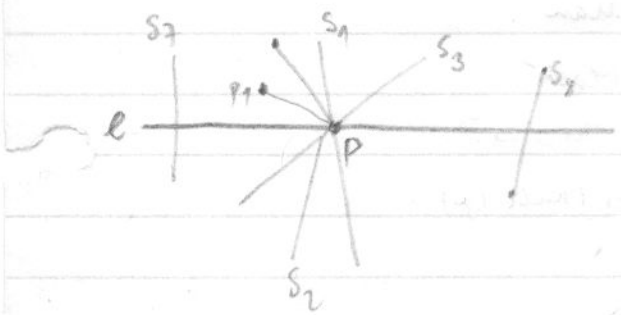
Algoritmus událost (průchod přímky přes bod $p \in Q$)
- $U(p)$ – všechny úsečky s dolním krajním bodem p
- v $T(p)$ najdi množinu $S(p)$ úseček obsahujících bod p
  $L(p) \subset S(p)$ je množina úseček kde p je dolní koncový bod
  $C(p)$ – množina vnitřních bodů

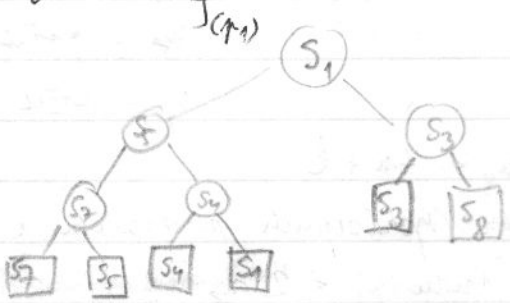
- jestliže $L(p) \cup C(p) \cup U(p)$ obsahuje více než 1 úsečku
- označ jako průsečík a ulož s $L(p) \cup C(p) \cup U(p)$
- vypusť úsečky z $L(p) \cup C(p)$ ze stromu $T(p)$
- ulož úsečky z $U(p) \cup C(p)$ do stromu T
- úsečky z $C(p)$ změní své pořadí. Horizontální úsečka musí mít bude poslední
- jestliže $U(p) \cup C(p) = \emptyset$

» je prostě nerozlišujeme koncový bod a průsečík a

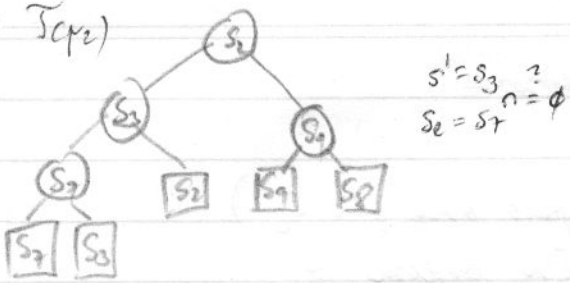
potom nechť S_e a S_p jsou levý a pravý soused p v T
  provedl algoritmus HLEDEJ NOVOU UDÁLOST (S_e, S_p, p)
- jinak najdi s' – nejlevější úsečku z $U(p) \cup C(p)$, S_e její levý soused. Provedl ALG. HLEDEJ N. UD (S_e, s', p)
  najdi s'' nejpravější z $U(p) \cup C(p)$, S_p její pravý soused
  NAJDI N. UD (s'', S_p, p)

NAJDI NOVOU UDÁLOST (s_1, s_2, p)
1. jestliže s_1 protíná s_2 v bodě, který je v uspořádání za p
2. potom vlož průsečík do Q





$L(p) = \{S_5, S_4\}$
$C(p) = \{S_1, S_3\}$
$U(p) = \{S_2\}$



ALG. PRŮSEČÍKY ÚSEČEK

I: množina S úseček v rovině

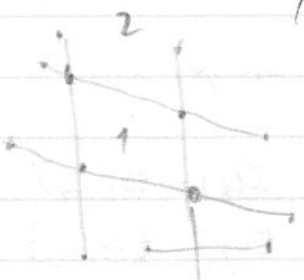
O: — " — průsečíků společné s úsečkami, na kterých leží —

1. do Q vložíme koncové body všech úseček
   koncům koncovým bodům přidej i úsečky
2. inicializuj T jako prázdný
3. dokud je Q neprázdná
   1. událost v Q a vymaž ji
      proveď alg. UDÁLOST

Lemma: Algoritmus skutečně najde všechny průsečíky
Důkaz: Všechny x se dostanou do Q

Lemma: Časová náročnost ALG je $(n+k)\cdot\log(n)$
náročnost ALG: vytvoření fronty $\sim O(n\log(n))$
počet událostí $\sim 2n+k$
- všechny operace se stromem i frontou $\sim O(n\log n)$
$m = \sum m_p \quad m_p = U(p)+L(p)+C(p)$

ukážeme, že $m \le 12n+6k-6$
$m = O(n+k)$



rovinný graf n_v - počet uzlů
n_e - počet hran
n_f - počet oblastí

$n_v = 2n+k$

$m_p \le$ počet hran vycházejících z vrcholu $p = \text{index}(p)$
$m = \sum p \le \sum_p \text{index}(p) = 2n_e$

každá hrana leží v 2 obl. / face kolem

$n_f \leq \frac{2n_e}{3} + 1$

↑ bude potřeba pro ohraničení oblastí

Eulerova věta pro rovinné grafy

$n_v - n_e + n_f = 2$

$2 \leq n_v - n_e + n_f \leq (2n+k) - n_e + \frac{2n_e}{3} + 1 = (2n+k) - \frac{n_e}{3} + 1$

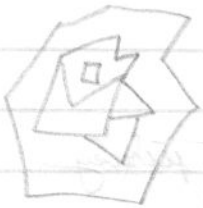
$\frac{n_e}{3} \leq 2n + k - 1$

$n_e \leq \frac{6 \cdot n_e}{3} \leq 6 \cdot (2n+k-1) = 12n + 6k - 6 \Rightarrow m = O(n+k)$

nejkonvexní vrstvy 

PŘEKRYVY MAP

datová struktura pro rozdělení roviny



- vrcholy
- hrany – půlhrany, e_1 a e_2 jsou dvojčata
- oblasti

doubly-connected edge list

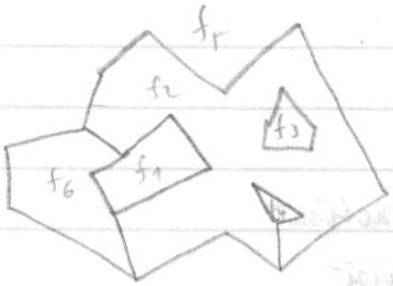
tabulka VRCHOLY:

vrchol | souřadnice | jedna hrana jdoucí k vrcholu

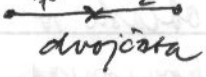
4.10.2007

Rovinné podrozdělení

- vrcholy a hrany tvoří rovinný graf
- oblasti určené hranami



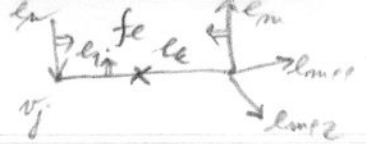
datová struktura: dvojitě spojitý seznam

každá hrana: 

dvojčata

vrchol	souřadnice	hrana vycházející z v	[dvojčata]
v	(x,y)	e_i	e_j

} seznam pro vrcholy



seznam pro hrany:

hrana	poč. vrchol	dvojčí	sousední oblast	následující hrana
e_i	v_j	e_ℓ	f_e	e_m

předchozí hrana

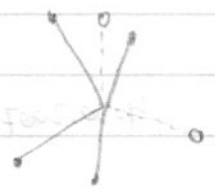
e_m

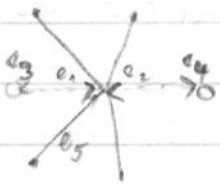
seznam pro oblasti:

oblast	hrana z vnější hranice	hrany z vnitřních hranic
f_i	e_j (nejvýše jedna)	$e_k, e_\ell \dots$

S_1 je rovinné podrozdělení popsané DCEL
S_2 je jiné " "
$O(S_1, S_2)$

D je sjednocený seznam S_1 a S_2 – používáme alg. zametací přímky
v bodech událostí počítáme nejen nové události a průsečíky,
ale rovněž aktualizujeme seznam D pro vrcholy a hrany.
případy:

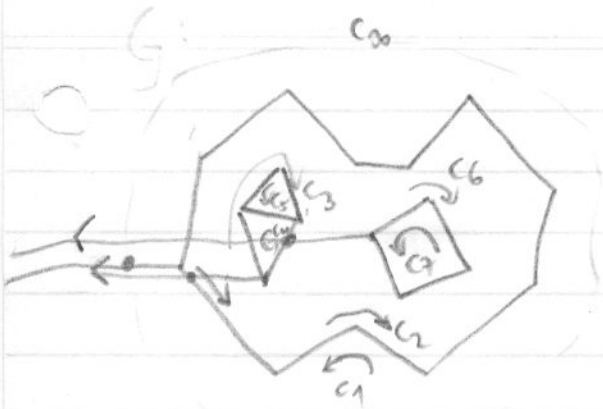


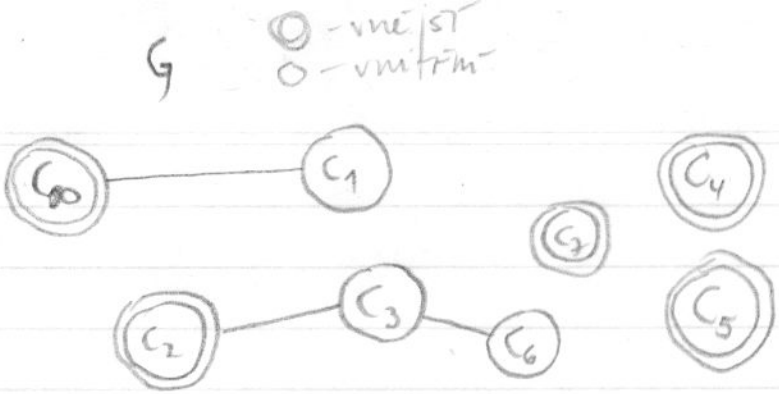


přibude e_3, e_4
$next(e_3) = next(e_2)$
$prev(e_3) = e_5$
zjistíme nejbližší hranu podle směru každého
směrů pomocí zametací přímky

aktualizujeme
8 a víc vrcholů

Nalezení oblasti v $O(S_1, S_2)$
- známe komponenty hranic - cykly - vnitřní
- vnější
- jak u daného cyklu poznáme, zda je vnější či vnitřní
- vezmeme nejlevější a nejpodlevější a koukneme na velikost úhlu – musí být pod 180° ⇒ vnější cyklus





počet oblastí = počet vnějších cyklů

Vypouštění hran v grafu G
- vezmeme ležící dolní bod vnitřního cyklu
  - z něj vedeme polopřímku vlevo
    - vezmeme první hranu, kterou protne
    - ta určuje hraniční cyklus stejné oblasti
    - bylo dva cykly spojíme v grafu G hranou

Lemma:
- každé komponentě souvislosti grafu G odpovídá právě jedna oblast rovinného podrozdělení

Dk: (1) spojené cykly určují jednu komponentu
(2) vnitřní cyklus nějaké oblasti je v komponentě s jiným vnějším cyklem
Sporem: C nějakým vnitřním cyklus dané oblasti vezmeme ten nejlevější

25.10.2007

pokračovat po 18.10.2007

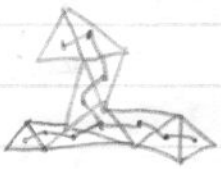
Odlitky
Nechť f je stěna a $\eta(f)$ je vnější normála
Lemma: odlitek je svazu mnohostěnu je neprázdný
ve směru d právě když $\sphericalangle(d, \eta(f)) \geq 90°$ pro všechny stěny mnohostěnu
$d = (d_1, d_2, 1)$ → táhneme nahoru
hledáme $\eta(f) = (a, b, c)$
$\sphericalangle(d, \eta(f)) \geq 90°$ $\cos(\sphericalangle d, \eta) \leq 90° = \frac{d \cdot \eta}{\|d\| \cdot \|\eta\|} \leq 0$ $d \cdot \eta \leq 0$

$ad_1 + bd_2 + c \leq 0$ polorovina v rovině $R^2(d_1, d_2)$
$(d_1, d_2) \in R$ je bod v průniku polorovin

18.10.2007

Jednoduchý n-úhelník rozdělit pomocí uhlopříček na $\Delta(n-2)$

K monitorování n-úhelníku stačí $[\frac{n}{3}]$ kamer
Vrcholy v triangulaci obarvíme třemi barvami tak, aby sousední vrcholy měly různou barvu



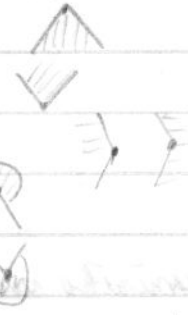
duální graf je strom
procházíme strom duálního grafu a obarvujeme vrcholy – $O(n)$

Algoritmus pro triangulaci $O(n \log n)$ – existuje i $A(n)$
(1) n-úhelník rozdělíme na y-monotonní mnohoúhelníky
(2) ty pak rozdělíme na Δ

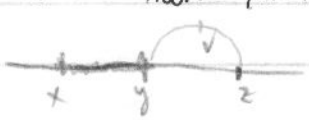
Mnohoúh. je monotonní k ℓ, jestliže pro každou přímku $\ell' \perp \ell$ je $\ell' \cap$ mnohoúhelník souvislá množina (ϕ, bod, úsečka)
ℓ osa $y \Rightarrow$ y-monotonní

co ruší monotonii 

Typy vrcholů mnohoúhelníka
(1) startovní vrchol
(2) koncový vrchol
(3) regulární vrchol
(4) split vrchol úhel > 180°
(5) merge vrchol



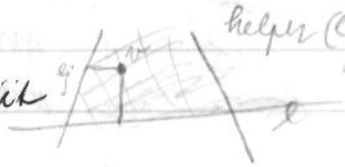
Lemma: jestliže mnohoúhelník nemá split a merge vrcholy, je monotónní:
mnohoúh není monotonní ⇒ má vrchol split / merge



⇒ nejvyšší vrchol ⇒ v je split

Řeší se zametací přímkou: split vrcholy odstraňujeme v bodě příchodu, merge vrcholy ale později

- Q – fronta událostí; událost = vrchol n-úhelníka
  v, w – dva vrcholy; v je před w jestliže $v_y > w_y \vee (v_y = w_y \wedge v_x < w_x)$
- pro každou polohu přímky známe strom T, který uchovává pořadí hran n-úhelníka protínající danou přímku
- zajímají nás pouze hrany, které mají mnohoúhelník zprava
  – ke každé hraně ve stromu je uložen pomocník = vrchol n-úhelníka nejníže nad e takový, že lze spojit vodorovnou úsečkou s hranou e uvnitř mnohoúhelníka



Algoritmus pro rozklad na mon. části
I: jednoduchý n-úhelník P zadaný ve formě dvojitě spojového seznamu.
O: rozložení P na mon. množiny úhelníky zapsané ve formě dvojitě spojového seznamu D

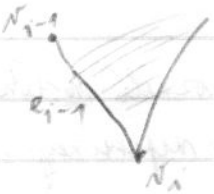
1. sestav frontu Q $\quad O(n \cdot \log n)$
2. zinicializuj prázdný strom T
3. pokud je Q neprázdná
4. odstraň první vrchol v z Q
5. zavolej proceduru podle typu odstraněného vrcholu

Procedury přidávají diagonály (1) je-li aktuální vrchol split nebo (2) je-li vyměněný helper merge. Mění helpery a aktualizují strom T.

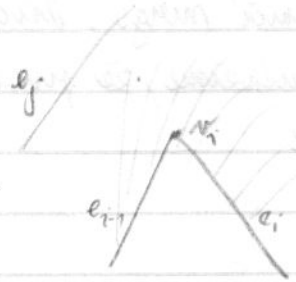
vrcholy očíslovány proti směru ⑤
$e_i = (v_i, v_{i+1})$

Procedura pro startovací vrchol v_i
1. vlož e_i do stromu T
2. helper $(e_i) \leftarrow v_i$

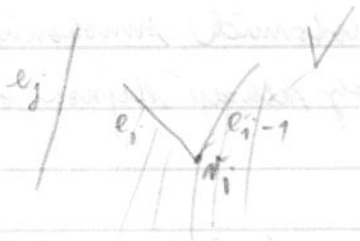
Procedura pro end vrchol v_i
1. jestliže helper (e_{i-1}) je typu merge
2. pak potom vytvoř diagonálu
   v_i – helper (e_{i-1})
3. vymaž e_{i-1} z T.



Procedura pro split vrchol v_i
1. najdi v T nejbližší levou stranu e_j k v_i
2. vytvoř diagonálu v_i – helper (e_j)
3. helper $(e_j) \leftarrow v_i$
4. vlož e_i do T, helper $(e_i) = v_i$



Procedura pro merge vrchol v_i
1. if helper(e_{i-1}) je typu merge
2. then spoj v_i – helper (e_{i-1})
3. vymaž e_{i-1} ze T
4. najdi v T stranu e_j nejblíže vlevo od v_i
5. if helper (e_j) typu merge
6. then spoj v_i – helper (e_j)
7. helper $(e_j) \leftarrow v_i$



Procedura pro regulární vrchol v_i
1. if P vpravo od v_i
2. then if helper (e_{i-1}) typu merge
3. then vytvoř diagonálu v_i – helper(e_{i-1})
4. vymaž e_{i-1} z T
5. vlož e_i do T, helper $(e_i) \leftarrow v_i$
6. if P vlevo od v_i

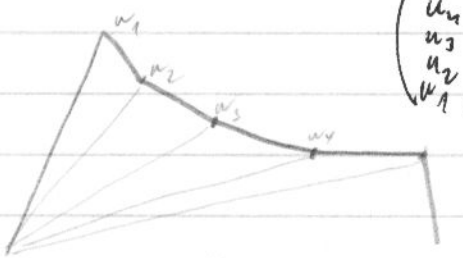
7. najdi e_j nejblíže k v_i zleva
8. if helper(e_j) typu merge
9. then spoj v_i – helper(e_j)
10. helper(e_j) ← v_i

Lemma: Algoritmus rozdělí n-úhelník na monotonní části.
Přidané diagonály se nepotínají.
Důkaz: Při průchodu spolu vrcholem přidáváme diagonálu
Přidaná diagonála zmizí merge. Přidané diagonály se nepotínají –
v těch procedur ukážeme, že přidáváme to, co nepotíná nic,
co už jsme přidali:
SPLIT: ...

Věta: alg. rozdělení ↑ má časovou složitost $O(n \cdot \log n)$. Vyrovnání $Q \approx O(n \cdot \log n)$
uložení změny v čase $O(\log n)$, pro n vrcholů

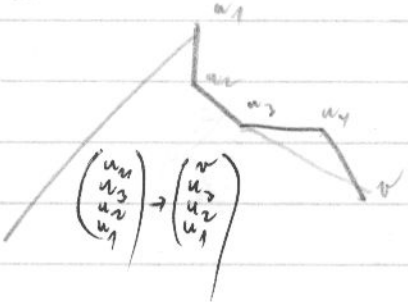
Triangulace monotonních mnohoúhelníků
(žádné dva vrcholy nemají stejnou souřadnici y)

pomocná situace



$\begin{pmatrix} u_4 \\ u_3 \\ u_2 \\ u_1 \end{pmatrix}$ ≡ stack – vlevo jedna strana, vpravo
pořadí stran svírajících úhel > 180°

- další vrchol vlevo → vložím pak na obr. vlevo



- další vrchol vpravo

ALG

I: striktně monotonní n-úhelník ve formě dvojic souvislého seznamu

O: Triangulace n-úhelníka P ve formě ——"——

1. vytvoř hrom a prázdnou cestu
2. jejich vrcholy seřaď lex. do seznamu Q. Nechť $w_1 \dots w_2$ je posloupnost v Q
3. vlož w_1 a w_2 do S
4. pro $j = 3$ do $n-1$
   je-li w_j ² vrchol nahoře v S na různých cestách
5. → then vymaž vše z S a spoj w_j se všemi vymazanými vrcholy
   else vymaž horní vrchol z S
   maž vrcholy z S tak dlouho, dokud spojnice s w_j leží v P
   poslední vymazaný vrchol přidej do S
   vlož w_j do S

přidej diagonály z w_n do vrcholů S s výjimkou prvního a posledního

Věta: Triangulace proběhne v čase $O(n)$

25.10.2007 II

1. Algoritmus na průnik polorovin
2. Algoritmus pro nalezení jednoho bodu
3. Algoritmus pro nalezení maxima lin. funkce na průniku polorovin

Algoritmus pro průnik polorovin PRŮNIK(H)

I: H obsahující n polorovin $H = \{h_1, h_2, \dots\}$

O: konvexní oblast $C = \bigcap_{i=1}^{n} h_i$

1. if card(H) = 1 then $C = \{H\}$
2. else rozděl H na H_1 a H_2 $\lceil \frac{n}{2} \rceil$ a $\lfloor \frac{n}{2} \rfloor$
3. $C_1 \leftarrow$ výsledek PRŮNIK(H_1)
4. $C_2 \leftarrow$ výsledek PRŮNIK(H_2)
5. $C \leftarrow$ PRŮNIK KONV. OBLASTÍ (C_1, C_2)

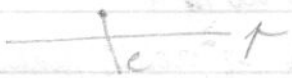
$T(n) = 2T(\frac{n}{2}) + O(n \log)$

$T(n) = O(n \cdot \log^2 n)$

umět odvodit

Algoritmus pro průnik dvou konvexních oblastí
pravá a levá cesta.

při rozdružování hranic pravíme o málo. –
Algoritmus zametací přímky
Události jsou vrcholy na levé, pravé cestě C_1, levé a pravé cestě C_2



dromě vrchol p hrany e v levé cestě C_1

(1) Testujeme, zda p leží mezi levou cestou C_2 a
pravou cestou C_2

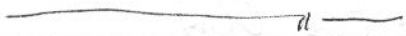
$\Rightarrow$ e tvoří levou cestu $C = C_1 \cap C_2$

(2) e protíná levou cestu C_2

(a)

$\Rightarrow$ levá cesta v $C = C_1 \cap C_2$ je f, e

(b)

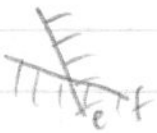
$=$  e, f

(3) e protíná pravou cestu C_2

(a)

$\Rightarrow$ e je konec levé cesty v $C_1 \cap C_2$
f je konec pravé cesty v $C_1 \cap C_2$

(b)



$\Rightarrow$ e je začátkem levé cesty $C_1 \cap C_2$
f začátkem pravé cesty

množinová algoritmus, $O(n)$
po zapojení do kostry $O(n \cdot \log n)$

III

maximalizuj $c_1x_1 + c_2x_2$

za podmínek $a_{11}x_1 + a_{12}x_2 \leq b_1$ } dává poloroviny
$a_{21}x_1 + a_{22}x_2 \leq b_2$
$\vdots$
$a_{n1}x_1 + a_{n2}x_2 \leq b_n$

hledáme maximum - nejvzdálenější bod od přímky ve směru normály
...

poloroviny $H = \{h_1, \dots h_n\}$
hraniční přímky $l_1, \dots l_n$
$\vec{c} \neq \vec{0}$ $\vec{c} = (c_1, c_2)$
$C_i = h_1 \cap h_2 \cap \dots h_i$

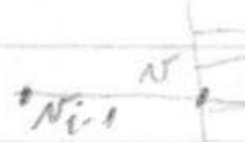
Řešení postupně pro $C_1, C_2 \dots C_n$ (postupně zmenšujeme oblast přidáváním polorovin)

v_i - bod, kde $c_1x_1 + c_2x_2$ nabývá maxima na C_i

Lemma: (a) je-li $v_{i-1} \in h_i$, pak $v_i = v_{i-1}$
(b) je-li $v_{i-1} \notin h_i \Rightarrow (C_i = \emptyset \vee v_i \in l_i)$ (l_i hraniční přímka h_i)

(a)

(b)



$f(v_{i-1}) \geq f(v) \geq f(v_i)$
$\Rightarrow v_i$ lze nabrat na l_i

jednorozměrná úloha lin. prog. = najít maximální číslo

v množině	$\cap \{x \geq x_i\}$	$P = \{x_i, x \geq x_i\}$	$\min P$	$\max L \leq \min P$
	$\{x \leq x_j\}$	$L = \{x_j, x \leq x_j\}$	$\max L$	$\Rightarrow$ libovolné číslo $= \min P$
				jinak $\emptyset$

algoritmus na hledání max, min ... $\sim O(n)$

NEOMEZENÝ LP(H, C), jehož výstupem je ↦ v případě, že ~~úloha LP~~ funkce $c_1x_1 + c_2x_2$ je neomezená na $\cap H$ (na níž ↦ fce roste do nekonečna). V opačném případě je výstupem $\cap H = \phi$ nebo dvě $E\ h_1, h_2$ tak, že $c_1x_1 + c_2x_2$ je omezená v $h_1 \cap h_2$ (v tomto případě má úloha řešení). Jeho alg. $\sim O(n)$.

Algoritmus 2D LP (H, C)
I: H, C
O: $\cap H = \phi$ jinak ↦ p v $\cap H$ na níž $c_1x_1 + c_2x_2$ roste do ∞ pokud ne, tak $v_n \in \cap H$, v kterém fce maximu
1. if NEOMEZENÝ LP(H, C) hlásí $\cap H = \phi$ nebo p ~~takové~~ dělej totiž
2. else h_1, h_2 jsou polorovinu z alg. NLP(H, C)
   for $i = 3$ to n
      if $v_{i-1} \in h_i$, polož $v_i = v_{i-1}$
      else řeš úlohu LP v dim 1 na přímce l_i, ~~výsledek je v_i~~
      pokud řešení neexistuje oblast $\cap H = \phi$, exit
      jinak výsledek $= v_i$
   oblast v_n jako řešení

Časový odhad: NLP $\sim O(n)$
- v nejhorším případě je potřebný čas $O(n^2)$

Náhodné 2D LP
RANDOM(k)
vybere náhodně číslo $1 \dots k$ v $O(1)$

NÁHODNÁ PERMUTACE $(A) \sim O(n)$
I: pole $A[1, \dots n]$
O: pole $A[1, \dots n]$ náhodně permutované
1. for $k = n$ to 2 {
2. RINDEX ← RANDOM(k)
3. zaměň $A[k]$ za $A[$RINDEX$]$ }

Náhodný algoritmus jako předchozí, pouze provedeme
NÁHODNÁ PERMUTACE $(h_3, \dots h_n)$

X_i je náhodná veličina
$X_i = 0$ je-li $v_{i-1} \in h_i$
$= 1$ $v_{i-1} \notin h_i$, pak potřebujeme $O(i)$
celkový čas k provedení je
$\sum_{i=2}^{n} O(i) X_i$ ← také náhodná veličina, hledáme střední hodnotu = průměrný čas
$E(\sum O(i) X_i) = \sum O(i) E(X_i) = \sum O(i) \cdot p(v_{i-1} \notin h_i) = \sum k \cdot i \frac{2}{i-2} \longrightarrow$

$p(v_{i-1} \notin h_i) = p($~~$v_{i-1} \in P_i$~~ optimální bod se změní při odstranění přímky $\in \{h_1, \dots h_i\}) = 2/i-2$

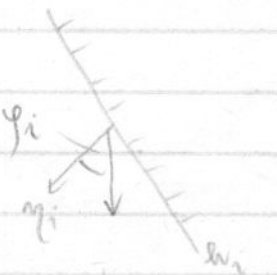
$\longrightarrow \leq \sum_{i=3}^{n} k \cdot 2 \cdot \frac{2}{i-2} = O(n)$

1.11.2007

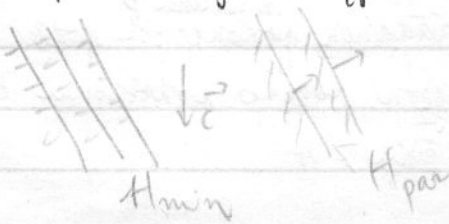
$\sum_{i=3}^{n} E(X_i) O(i) \leq \sum \frac{2}{i-2} O(i) = \sum_{i=2}^{n} O(1) = O(n)$

$0 \cdot p(v_i = v_{i-1}) + 1 \, p(v_i \neq v_{i-1})$
$p(v_i \neq v_{i-1}) \leq \frac{2}{n-2}$

NEOMEZENÝ LP $(H, \vec{c})$
- $\bigcap_{i=1}^{n} h_i = \emptyset$
- pro $c_1 x + c_2 y$ je omezená v $h_i \cap h_j$
- poloúsečka $\rho \in \bigcap_{i=1}^{n} h_i$ na níž je fce $c_1 x + c_2 y$ neomezená



φ_i je $\measuredangle$ mezi η_i (vnější normálou k h_i) a $\vec{c}$
$H_{min} = \{h_j \in H, \varphi_j = \min_{1 \leq i \leq n} \{\varphi_i\}\}$
$H_{par} = \{h_j \in H, \eta_j = -\eta_i, h_i \in H_{min}\}$

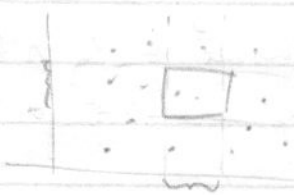


Spočítáme průnik
$h_i \in H_{min} \cup H_{par}$

(1) průnik $= \emptyset$, tak celý výsledek je $\emptyset$
(2) není $\emptyset$, je omezen $l_i^*, h_i^* \in H_{min}$ (jde o pás)

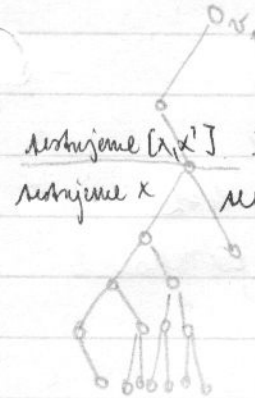
NEOMEZENÝ LP $(H, \vec{c})$
1: $H, \vec{c}$
0: tři možnosti (1) $(H, \vec{c})$ neomezený na polopřímce s
(2) $(H, \vec{c})$ omezený na $h_1 \cap h_2$
(3) $(H, \vec{c})$ $\cap H = \emptyset$
1. for all $h_i \in H$ spočti φ_i
2. nechť h_i je $\in$ s $\varphi_i = \min_{1 \le j \le n} \varphi_j$
3. $H_{min} \leftarrow \{ h_j \in H, \eta_j = \eta_i \}$
4. $H_{par} \leftarrow \{ h_j \in H, \eta_j = -\eta_i \}$
5. $\hat{H} \leftarrow H - \{ H_{min} \cup H_{par} \}$
6. spočítej $\cap [H_{min} \cup H_{par}]$
7. if $\cap = \emptyset$ then end("$\emptyset$")
8. else $h_i^* \in H_{min}$ je $\in$ spodní hranicí průniku, hran. přímka l_i^*
if existuje $h_j^* \in \hat{H}: h_i \cap h_j^*$ je omezená ve směru $\vec{c}$ then
pič $(\{ h_i^*, h_j^* \}, \vec{c})$ je omezený LP
else pič $(H, \vec{c})$ neomezený $\not\in$ na $\mapsto l_i^* \cap \bigcap_{h_j \in H} h_j$

Stavy řešení vyhledávání



vyhledávání v DB

1-dimenzionální verze
P = množina čísel
$[x, x']$ interval
Pro P vytvoříme bin. vyhledávací strom s touto vlastností
(1) listy stromu tvoří uspořádanou podmnožinu P podle velikosti
(2) pro každý vrchol v (číslo), listy jsou levého podstromu čísla $\le v$, pravého podstromu jsou čísla $> v$



$l(x)$ – levý následník

rozsjemne $[x,x']$ štěpící vrchol: x vlevo, x' vpravo

rozsjemne x | rozsjemne x'

NAJDI ŠTĚPÍCÍ VRCHOL (T, x, x')

1: strom T, čísla $x \leq x'$

0: uzel v takový, že $x \leq v < x'$
nebo něco trochu jiného

1. $v \leftarrow$ root T
2. do v není list $\wedge$ $(x' \leq v \vee x > v)$ then
3. if $x' \leq v$ then $v \leftarrow l(v)$
4. else $v \leftarrow r(v)$
5. od
6. vrať (v);

1-dimenzionální VYHLEDÁVÁNÍ (T, x, x')

1: T strom, čísla x, x'

0: všechny listy $T \in [x, x']$

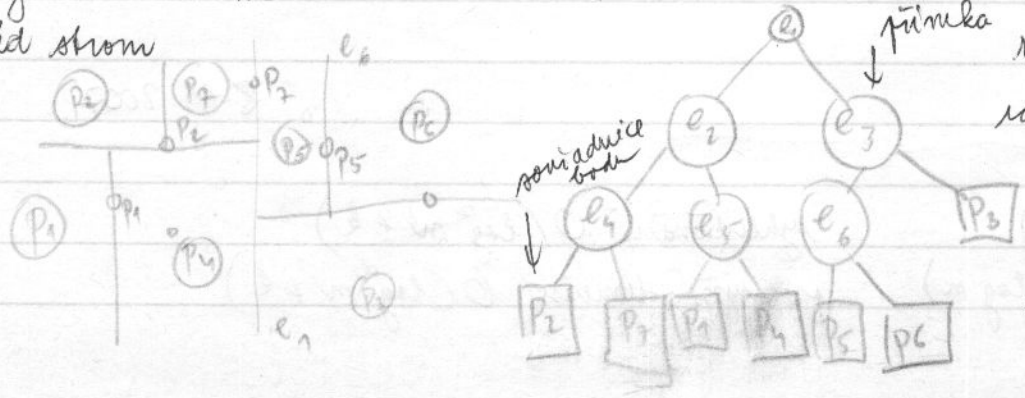
1. $v \leftarrow$ NAJDI ŠB (T, x, x');
2. if v je list then zjisti, zda leží v $[x, x']$
3. else $v \leftarrow l(v)$

do v není list then
if $x \leq v$ then vrať (podstrom $r(v)$), $v \leftarrow l(v)$
else $v \leftarrow r(v)$

zjisti zda list leží v $[x, x']$
opakuj pro x'

vytvoření stromu $O(n \cdot \log n)$
paměť $O(n)$
vyhledávání $O(\log n + k)$
↓ délka intervalu

Vyhledávání ve 2 dimenzích
kd strom



region (p) je oblast roviny omezená E, které jsou určeny přímkami uloženými ve vrcholech uzel nad v.

4.9 1A

VYTVOŘ kd-strom (P, d) ← množina bodů, hloubka

← začátek / nebo — dělením v závislosti na sudosti a lichosti

1: P, $d \in \mathbb{N}$

0: kořen kd stromu, v němž je uložena P

1. if P obsahuje pouze 1 bod then označ tento bod jako kořen a list

2. else if d je sudé then rozděl P na 2 množiny vertikální přímkou ℓ

po souřadnici x = medián P_x

$P_1 = \{p \in P \mid p_x \leq \text{medián } P_x\}$

$P_2 = \{p \in P \mid p_x > \text{medián } P_x\}$

» žádné 2 body nemají stejnou x a y souřadnici !!! «

else rozděl P na 2 množiny horizontální přímkou ...

... obdobně ...

$v_{\text{vlevo}} \leftarrow$ VYTVOŘ kd-strom (P_1, $d+1$)

$v_{\text{vpravo}} \leftarrow$ VYTVOŘ kd-strom (P_2, $d+1$)

8. vytvoř uzel v uchovávající přímku ℓ

9. ohlas $v(\ell)$

$R =$

SEARCH kd-STROM (v, R)

1: kořen podstromu v kd-stromu a obor $R = [x, x'] \times [y, y']$

0: všechny body v R

if v je list then write (leží-li v R)

else if region ($\ell(v)$) je celý obsažen v R then write (podstrom s kořenem $\ell(v)$)

else if region $\ell(v)$ protíná R then SEARCH kd-STROM ($\ell(v)$, R)

if region $r(v)$ je celý obsažen v R then write (podstrom s kořenem $r(v)$)

else if region $r(v)$ protíná R then SEARCH kd-STROM ($r(v)$, R)

» region ($\ell(v)$) = region (v) $\cap$ $\ell(v)$ [levá půlrovina] «

... vytvoření v čase $O(n \log n)$

$T(n) = 2T(\frac{n}{2}) + O(n)$

$T(k) = k \cdot 2^k$

vytvoření $O(n \cdot \log n)$

vyhledávání $O(\sqrt{n} + k)$ kde k je počet bodů v oblasti R

v dimenzi d je čas v $O(n^{1-\frac{1}{d}} + k)$

8.11.2007

Range tree

paměť $O(n \log n)$ vyhledávání $O(\log^2 n + k)$

konstrukce $O(n \log n)$ pro vyšší dimenze $O(\log^d n + k)$

2D stromy: body se stejnou x resp. y souřadnicí?

obecná množina P: $p = (p_x, p_y)$

$p' = ((p_x, p_y), (p_y, p_x))$ a uspořádáme lexikograficky resp. 1. i 2. složka

$p_1 = q_1 \Leftrightarrow p_x \leq q_x \vee (p_x = q_x \wedge p_y \leq q_y)$

P: množina bodů s různými x a y souřadnicemi

$\mathcal{T}$ - vyvážený binární strom, který vyhledává podle souřadnice x v množině P

$\mathcal{T}(v)$ - podstrom v $\mathcal{T}$ s vrcholem v – listy označíme $P(v)$

- na množině jeho listů uděláme usp. podle y $\Rightarrow$ strom $\mathcal{T}_{as}(v)$

BUILD 2D RANGE TREE (P)

1: množina P_n bodů v rovině

0: kořen 2D range tree stromu

vytvoř $\mathcal{T}_{as}$ na P_y (y souřadnice z P), jako listy ulož body z P (obě souřadnice)

if P obsahuje 1 bod then vytvoř list o tomto bodu

else rozděl P na dvě části P_{left} P_{right} podle

x souřadnice (p_x ... medián P)

$v_{left} \leftarrow$ BUILD 2D RANGE TREE (P_{left})

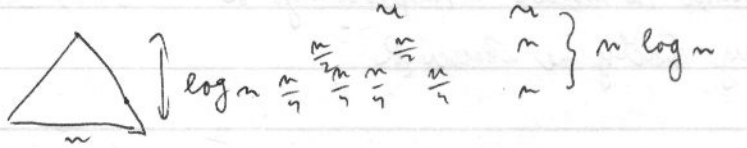
$v_{right} \leftarrow$ ——— // ——— (P_{right})

$v \leftarrow p \in P$, p_x = medián P_x; vytvoř $\mathcal{T}_{as}(v)$

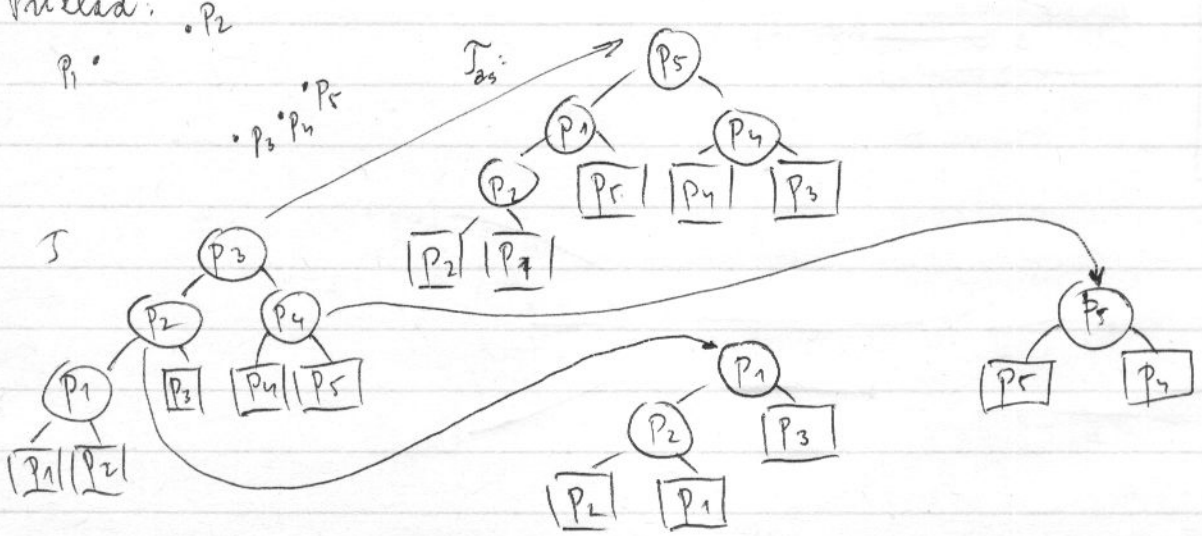
return (v)

Jemná: paměť: $O(n \log n)$

čas: $O(n \log n)$



Příklad:



1D_SEARCH (T, $[x, x'] \times [y, y']$)

I: range tree T, interval ↑ (obor) R

O: všechny listy v T ležící v R

$v_{split} \leftarrow$ NAJDI ŠTĚPÍCÍ VRCHOL (T, $[x, x']$)

if v_{split} je list Jakým způsobem zjistit, zda leží v R

else $v \leftarrow P(v_{split})$

while v není list do

if $x \leq v_x$ // pravý podstrom je ok $\in [x, x']$

then 1D SEARCH ($T_{as}(r(v))$, $[y, y']$); $v \leftarrow l(v)$

else $v \leftarrow r(v)$

zjistit, zda v leží v R ($v \in R$)

Obdobně pro x'

Věta: čas $O(\log^2 n + k)$ k je počet nalezených bodů

Lokalizace bodu

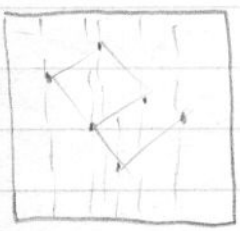
- dáno rozdělení roviny (pomocí n úseček)
- žádné 2 konc. body nemají stejnou x souřadnici
- rozdělíme rovinu na svislé pásy – velké nároky na paměť ($O(n^2)$)

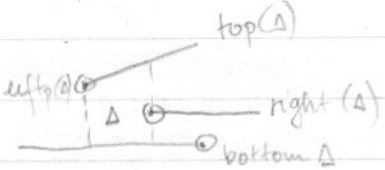
⇒ lichoběžníková mapa

S vpíšeme n úseček – neprotínají se $T(S)$ lichoběžníková mapa

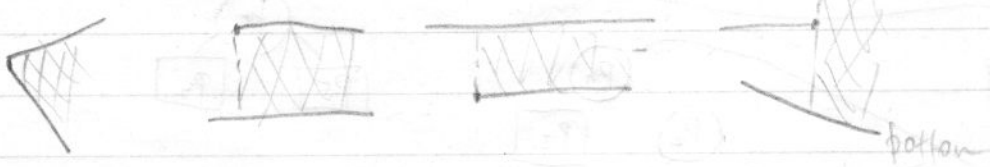
- všechny úsečky ne čtverci R

-2





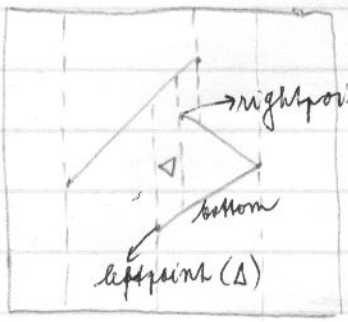
levá strana:



15.11.2007

S - systém neprotínajících se úseček $\angle$ ve čtverci R

$\mathcal{T}(S)$

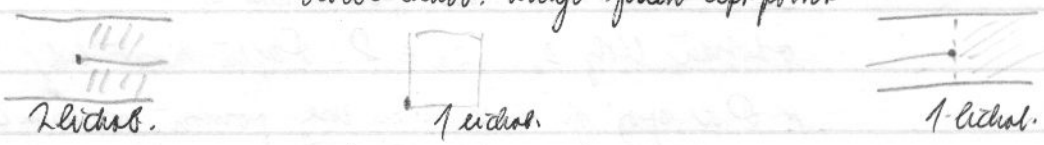


Lemma: $\mathcal{T}(S)$ obsahuje nejvýše $6n+4$ bodů (vrcholů $\square$ a Δ) a nejvýše $3n+1$ $\square$

Dk: počet bodů $\leq 4 + 2n + 4n$

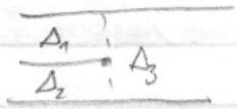
čtverce, strany vytvářející konce úseček

Kolik lichob. určuje jeden leftpoint?



$\Rightarrow \leq 1 + 1n + 2n$

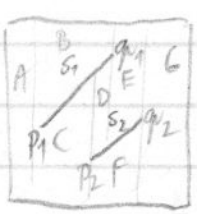
Sousední lichoběžníky v $\mathcal{T}(S)$

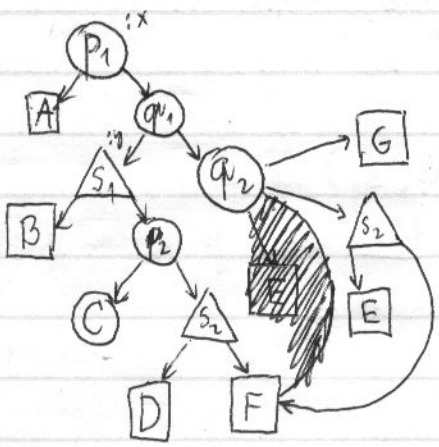


- mají společnou svislou hranu
- sousední lichoběžník je horní, mají společnou horní hranu
  - další - - dolní - -
  - pravý - - - -

PŘÍRŮSTKOVÝ NÁHODNÝ ALGORITMUS

D je orientovaný graf. Z každého uzlu vycházejí 2 hrany. Koncové uzly - listy jsou popsány lichob. z $\mathcal{T}(S)$. Vnitřní uzly jsou typu x... - označeny krajními body úseček ↔ vlevo, vpravo od bodu
a typu y... - označeny úsečkami $s \in S$ ↔ nad, pod úsečkou





nevytváříme úsečku, dokud nejsou zpracovány oba body

ALGORITMUS LICHOBĚŽNÍKOVÁ MAPA (S)

I: množina S o n úsečkách

O: lichoběžníková mapa $\mathcal{T}(S)$ a vyhledávací struktura $\mathcal{D}$

Určí obdélník R, v němž leží všechny úsečky z S

inicializuj $\mathcal{T}(S)$ jako graf s jedním vrcholem = listem R.

$\mathcal{T}(S) = \{R\}$

Náhodně urči pořadí úseček z S

for $i = 1$ to n do najdi $\Delta_0, \Delta_1, \Delta_2 \dots \Delta_k$ z $\mathcal{T}(S)$, které jsou protínány úsečkou s_i (s_i obsahují jejich odstraň $\Delta_0, \dots \Delta_k$ z $\mathcal{T}$, které a nahraď je Δ, které vznikne přidáním s_i odstraň listy $\Delta_0 \dots \Delta_k \in \mathcal{D}$. Doplň nové listy v $\mathcal{D}$ a spoj je stávajícími uzly pomocí vnitřních uzlů označených Δ, $\textcircled{p_i}$, $\textcircled{q_i}$, což jsou koncové body s_i

PODROB MĚ

$S_i = \{s_1, \dots s_i\} \in \mathcal{T}(S_{i-1})$ a

Přidáváme s_i do $\mathcal{T}(S_{i-1})$

Δ_{j+1} je pravý soused Δ_j

najdi Δ_0 p_i je levý koncový bod úsečky s_i

SLEDUJ ÚSEČKU ($\mathcal{T}$, s_i)

I: lichob. mapa $\mathcal{T}$ a úsečka s_i

O: posloupnost $\Delta_0 \dots \Delta_k$ lichoběžníků z $\mathcal{T}$ protínaných úsečkou s_i

nechť p a q jsou levý a pravý koncový bod s_i

v $\mathcal{D}$ najdi Δ_0, v němž leží p

$j = 0$

while q leží vpravo od rightpoint(Δ_j) do Δ_j

if rightpoint(Δ_j) leží na s_i then Δ_{j+1} je dolní pravý soused Δ_j

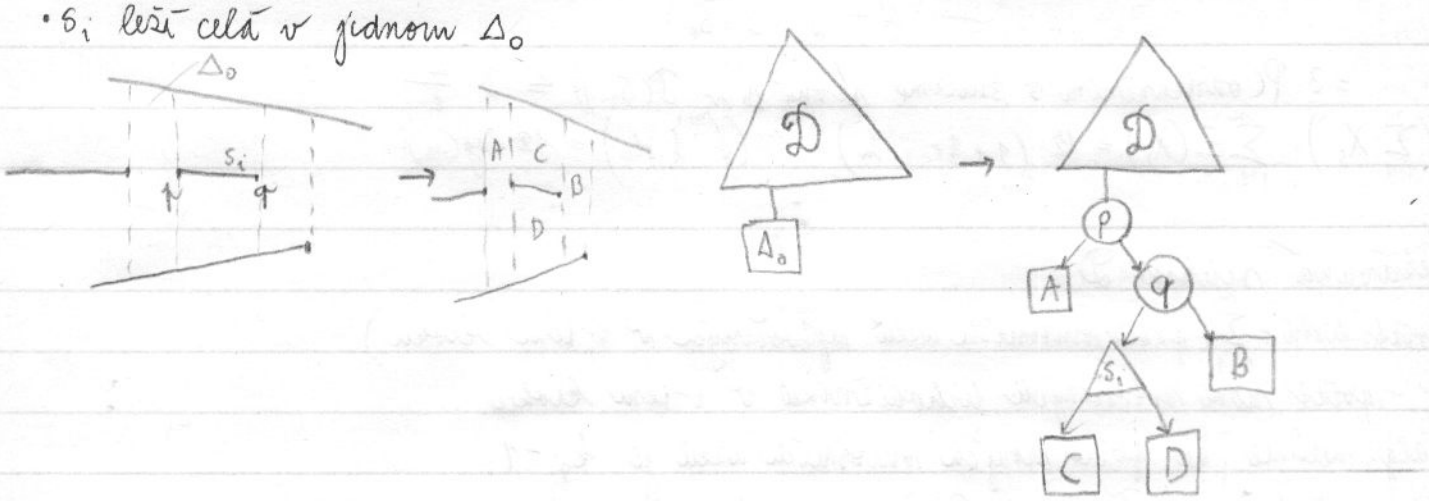
else Δ_{j+1} je horní pravý soused Δ_j

$j := j+1$

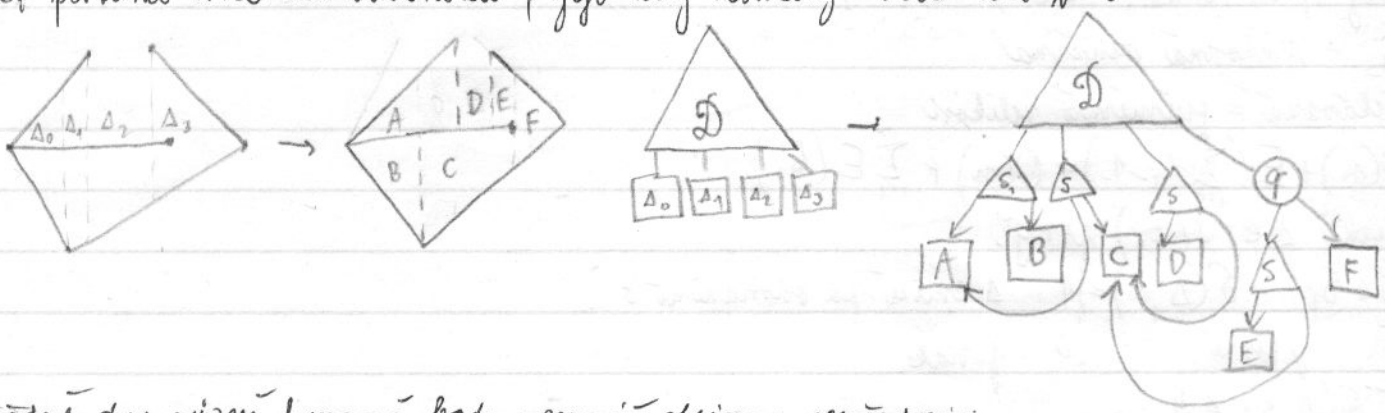
vypiš $\Delta_0 \dots \Delta_j$

Aktualizace $\mathcal{T}$ a $\mathcal{D}$

- s_i leží celá v jednom Δ_0



- s_i protíná více lichoběžníků, jejich levý koncový bod leží v $\mathcal{T}$



žádné dva různé koncové body nemají stejnou souřadnici

obecnou transformací

$$\begin{pmatrix} x \\ y \end{pmatrix} \xrightarrow{\varphi} \begin{pmatrix} x + \varepsilon y \\ y \end{pmatrix}$$

$p \neq q$ ~~existuje~~ $\varepsilon \Rightarrow \varphi(p) \neq \varphi(q)$

existuje ε tak, že $\varphi_\varepsilon(p_1), \varphi_\varepsilon(p_2) \ldots$ nemají stejnou souřadnici x

Věta: ALGORITMUS LICHOB. MAPA dává mapu $\mathcal{T}(\mathcal{Y})$ a vyhledávací strukturu $\mathcal{D}$ v čase $O(n \cdot \log n)$. Očekávaná velikost vyhledávací struktury je $O(n)$ a hledání bodu q [v obecné poloze] lze provést v čase $O(\log n)$.

q je pevné

X_i – počet uzlů vytvořených v i-tém kroku, kterými vede cesta při hledání q

X_i – náhodná veličina; nabývá hodnot $0 \ldots 3$ (viz obrázky nahoře)

$X_i \neq 0$ právě když $q \in \Delta \in \mathcal{T}(S_{i-1})$ a s_i protíná Δ a Δ není v $\mathcal{T}(S_i)$

(c neuvažujeme)

$E(X_i) \le 3 \cdot P(\Delta^T(S_{i-1}) \neq \Delta^q(S_i))$

(uchovržníků v $T(S_{i-1})$ v nemž leží q — " — v $T(S_i)$)

$= 3 \cdot P(\text{odstraněnému } s \text{ zničíme pevný } \Delta \text{ v } T(S_i)) \le 3 \cdot \frac{4}{i}$

$E(\sum_{i=1}^{n} X_i) = \sum_{i=1}^{n} E(X_i) = 12 \cdot (1 + \frac{1}{2} + \dots \frac{1}{n}) \doteq 12 \left(\int_1^{n+1} \frac{1}{x}\, dx\right) \le 12 \log(n)$

Očekávaná velikost $\mathcal{D}$

počet listů + $\sum_{i=1}^{n}$ (počet vnitřních uzlů vytvořených v i-tém kroku)

k_i - počet nově vytvořených uchovržníků v i-tém kroku

z alg. víme, že počet nových vnitřních uzlů je $k_i - 1$

$k_i \le$ počet $\square$ v $T(S_i) = O(i)$

pak počet listů $= O(n) + O(\frac{n(n-1)}{2}) = O(n^2)$ ← horní odhad

k_i - náhodná veličina

očekávaná = průměrná velikost

$O(n) + E(\sum_{i=1}^{n} k_i - 1) \doteq O(n) + \sum_{i=1}^{n} E(k_i)$

vrcholů $\Delta \in T(S_i)$ pevný.

$s \in S_i \quad \delta(\Delta, s) = \begin{cases} 1 & \Delta \text{ zmizí po odstranění } s \\ 0 & \text{jinak} \end{cases}$

(první)

$\sum_{s \in S_i} \delta(\Delta, s) \le 4$

$\sum_{\Delta \in T(S_i)} \sum_{s \in S_i} \delta(\Delta, s) \le 4 \,|T(S_i)| = O(i)$

$E(k_i) = \frac{1}{i} \sum_{\Delta} \sum_{s} \delta(\Delta, s) \le \frac{O(i)}{i} = O(1)$

Očekávaná velikost

$O(n) + \Sigma E(k_i) = O(n) + O(n) = O(n)$

22.11.2007

Čas konstrukce $T(S)$ a $\mathcal{D}(S)$

$O(1) + \sum_{i=1}^{n} \{ O(\log i) + O[E(k_i)] \}$ $\quad k_i$ - přidané uchovržníky

↑ očekávaný čas vyhledání listu konvexního bodu úsečky

$= O(\sum_{i=1}^{n} \log i) + O(n) \le O(n \log n) + O(n) = O(n \log n)$

Lemma: Nechť S je systém úseček a $q \in \mathbb{R}^2$ pevný bod a $\lambda > 0$ parametr. Pak že vyhledávací cesta pro q má délku větší než $3\lambda \ln(n+1)$ uzlů je nejvýše $\frac{1}{(n+1)^{\frac{1}{4}\lambda \ln \frac{5}{4} - 1}}$

Lemma: Pravděpodobnost, že výška vyhledávací struktury je $> 3\lambda \ln(n+1)$ je nejvýše $\frac{2}{(n+1)^{\lambda \ln \frac{5}{4} - 3}}$

$2(n+1)^2$ oblastí

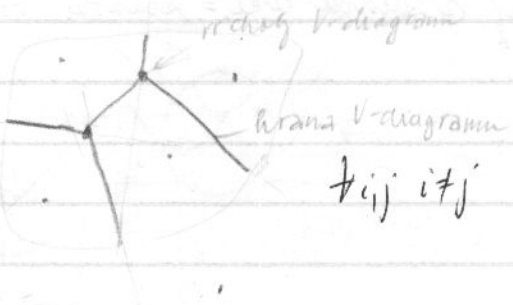
Pro bod známe $2(n+1)^2$ vyhl. cest
$\Rightarrow$ vzorec z $L1 \cdot 2(n+1)^2 =$ vzorec z $L2$

$\lambda = 20 \ldots 60 \ln(n+1)$ – omezení výšky

$\frac{2}{(n+1)^{1.4}} \leq \frac{1}{4}$ pro $n > 4$ $\quad P(\text{výška} \leq 60 \ln(n+1)) \geq 3/4$

Obdobně lze ukázat, že pravděpodobnost, že velikost D je $\leq k n$ (pevné) je $\geq 3/4$. Pravd..
Konstrukční čas je $\leq Ln \log n$ je alespoň $3/4$
Tedy celkově $3 \cdot 3 \cdot 3/4 \cdot 4/4 = 27/64 > \frac{1}{3}$ – při závislosti

Diagramy Voroného



$i \neq j$

- n bodů v rovině $p_1 \ldots p_n$
- $V(p_i) = \{q \in \mathbb{R}^2, \text{dist}(q, p_i) \leq \text{dist}(q, p_j)\}$
- $V(p_i) = \bigcap_{i \neq j} h(p_i, p_j)$ mnohoúhelník

Lemma: Počet vrcholů V-diagramu $\leq 2n - 5$
počet hran $\leq 3n - 6$

$(n_v + 1) - n_e + n = 2$ $\quad$ půlhran je $2n_e$
$2n_e \geq 3(n_v + 1)$

$n_v + 1 - \frac{3}{2}(n_v + 1) + n \geq 2 \quad / \cdot 2$

$2n - 5 \geq n_v \Rightarrow n_e = (n_v + 1) + n - 2$

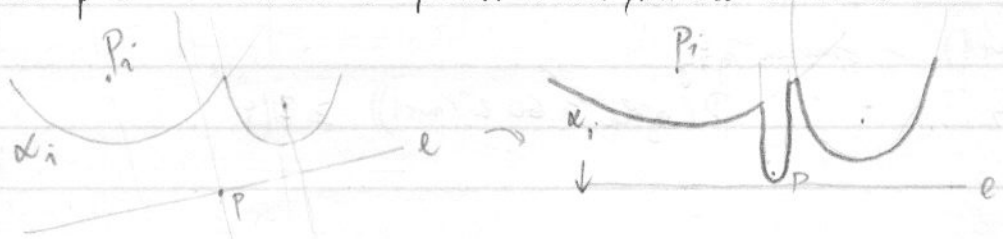
$n_e \leq 3n - 6$

Podrozdělení s oblastmi $V(p_i)$ označíme $Vor(P)$.
q je vrchol $\in Vor(P) \Leftrightarrow \exists$ O se středem q na níž leží alespoň 3 body $\in P$ a uvnitř neleží žádný.
Osa o úsečky $p_i p_j$ určují hranu V-diagramu, jestliže $\Leftrightarrow \exists$ O se středem na o procházející p_i a p_j a neobsahující žádný další bod množiny P.

zametací přímka se pořadím
beach line – pobřežní linie

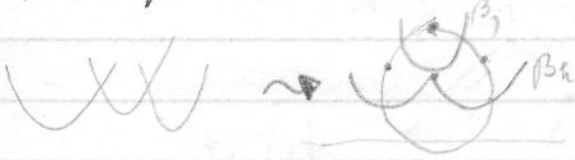
paraboly (stejná vzdálenost od bodu a přímky)
při animaci to lze pozorovat

Bod $p \in P$ dělící na l způsobí vznik nového oblouku na beach line.



Lemma: jediný způsob, jak se na pobřežní linii může objevit nový oblouk paraboly je průchod l přes $p \in P$

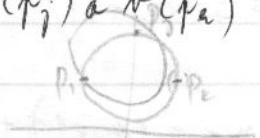
Hledáme body takové, že když l přes ně přechází → vymizí oblouk paraboly z beach line = kruhové body.



kruhový bod je bod kružnice obsahující body p_i, p_j, p_a, která uvnitř nemá žádný bod $\in P$.
→ ten spočítáme průsečíkem na l

s myšlenkou v souvislosti

Lemma: Všechny vrcholy V-diagramu jsou definovány přechodem přímky l přes kruhový bod.
q vrchol mezi $V(p_i)$, $V(p_j)$ a $V(p_a)$
střed kružnice $\beta_i \cap \beta_j - \beta_a$



$\beta_{i,j,a}$ byly v pobřežní linii

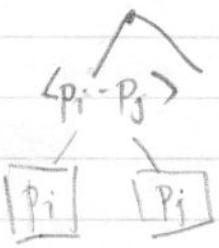
Datová struktura

Fronta Q určena body $\in P$ a kruhovými body

Pobřežní linie má nejvýše $2n-1$ oblouků $\Rightarrow$ paměť $O(n)$

Strom T – vyvážený binární strom; v listech jsou oblouky pobřežní linie zadané body $\in P$; T slouží k vyhledání oblouku, který je nad daným bodem na l

- vnitřní uzly zadávají hrany V-diagramu
  - v každém listu je odkaz na kruhový bod ztotožněný tímto bodem jako prostředním



kruhový bod (p_i, p_j, p_k) – vymizení paraboly je vrcholem ve V-diagramu

V-DIAGRAM (P)

I: množina P n bodů

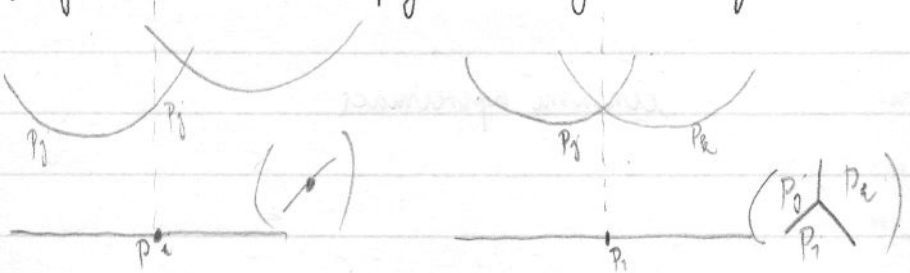
O: V-DIAGRAM Vor (P) uvnitř čtverce obs. všechny $p \in P$ ve formě dvojitě spojitého seznamu

inicializuj Q (body z P uspořádané podle y-souřadnice
T je prázdný

while Q je neprázdná do vezmi událost z Q s největší y-sou.:
if událost $p_i \in P$ then HANDLE SITE EVENT (p_i)
else HANDLE CIRCLE EVENT (p_i)
vymaž událost z Q

vnitřní uzly T reprezentují hrany Vor (P), které jsou polopřímky
najdi čtverec R obklopující P a vytvoř dvojitě spojitý seznam pro Vor (P) a R

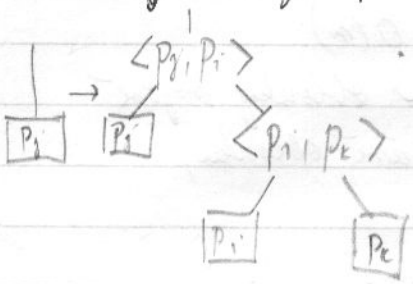
29. 11. 2007



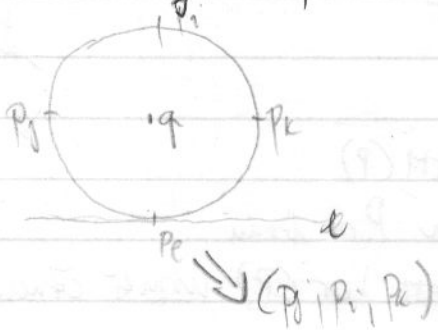
HANDLE SITE EVENT (p_i)

najdi v T oblouk α_j vlevo od p_i a α_k vpravo od p_i (často $j=k$)
všechny kruhové události vytvořené pomocí p_j a p_k vymaž

v T nahradí list reprezentující $\mathcal{L}_{p_j}$ a $\mathcal{L}_{p_k}$ podstromem o 3 listy p_j | p_i | p_k. Jako uzly reprezentující zlomové body dej $\langle p_j, p_i \rangle$ $\langle p_i, p_k \rangle$



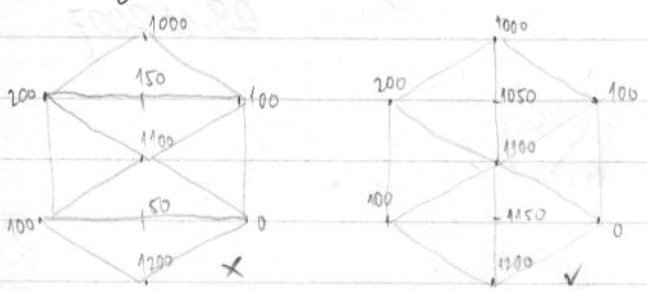
- Proveď vyvážení stromu T
- Vytvoř ve $Vor(P)$ nový záznam pro hranu oddělující $v(p_i)$ a $v(p_j)$ případně hranu v (p_i) a $v(p_k)$.
- Zkontroluj, zda nově trojice oblouků vytváří kruhový bod, pokud ano, dopeň do Q.



HANDLE CIRCLE EVENT (p_c)
- vymaž kruhovou událost p_c z Q
- vymaž list p_i z T
  aktualizuj T a proveď vyvážení
- přidej střed kruhu pro událost p_c do seznamu vrcholů $Vor(P)$ a vlož do $Vor(P)$ novou hranu mezi p_j a p_k
- Zkontroluj, zda nově trojice oblouků vytváří kruhový bod, pokud ano, zařaď jej do Q

Lemma: Algoritmus vyžaduje $O(n)$ paměti a $O(n \log n)$ času

Delaunayova triangulace



lineární aproximace

- „snažíme se o co největší úhly"

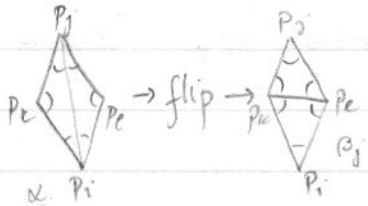
Věta: Nechť P je množina n bodů v rovině. Nechť její konvexní obal má k hran. Potom každá triangulace má $2n-2-k$ Δ a $3n-3-k$ hran.

Důkaz: Eulerova věta. Počet Δ nechť je m. $3m$ hran $+k$ $\sim$
$\frac{3m+k}{2}$ = počet hran $= h$. EV: $n-h+(m+1)=2$.

$$n-\frac{3m+k}{2}+m+1=2 \Rightarrow 2n-3m+k+2m=2 \Rightarrow 2n-k-2=m \quad \square$$

Je nějaká triangulace množiny P má m Δ a ty mají $3m$ úhlů. Seřadíme podle velikostí úhlů $\alpha_1 \leq \alpha_2 \dots \leq \alpha_{3m}$. Dvě triangulace můžeme porovnávat. $T < T' \Leftrightarrow$ posloupnost úhlů pro T je v lexikografickém uspořádání menší než posloupnost pro T'. T je optimální, pokud $\forall T' : T' \leq T$.

Legální triangulace
$\Leftrightarrow$ všechny hrany jsou legální

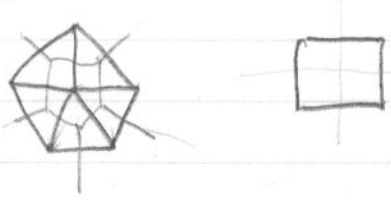


P_iP_j je ilegální v (P_i, P_e, P_j, P_e) jestliže po flipu je minimální úhel větší než $\min(\alpha_i) < \min(\beta_j)$

P_iP_j je ilegální pro jakýkoliv, jestliže kružnice procházející $P_iP_jP_e$ obsahuje uvnitř bod P_e

Věta: Optimální triangulace je legální.
Důkaz: Sporem

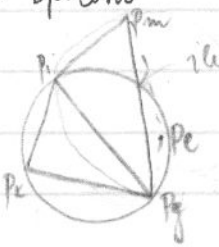
Delaunayova triangulace
Delaunayův graf = duální graf k diagramu Voroného. Body množiny P jsou vrcholy a vrcholy P_i a P_j jsou spojeny hranou, pokud spolu sousedí v V-diagramu



Věta: Tento graf je rovinný (žádné 2 hrany se nepotkají).
$p_i p_j$ tvoří hranu $\Leftrightarrow \exists$ kružnice mající $p_i p_j$ na kružnici a uvnitř není žádný z P. Pokud nějaká oblast v Del-grafu není Δ, pak její vrcholy leží na 1 kružnici.
Del-triangulace vznikne z D-grafu triangulací víc než 3 úhelníků.

Věta: Triangulace T množiny bodů P v rovině je Delaunayova
$\Leftrightarrow$ (vždy) kružnice opsaná Δ z T neobsahují uvnitř žádný další bod.
Důkaz: $\Rightarrow$ nafukujeme kružnici až se dotkneme ...
$\Leftarrow$ všechny Δ a opsané O neobsahují další bod uvnitř

Věta: Triangulace je legální $\Leftrightarrow$ je Delaunayova.
$\Leftarrow$ z předchozí věty $\Rightarrow$ neexistují ilegální hrana
$\Rightarrow$ sporem



není D-triangulace díky p_e bodu $\}$ $\Rightarrow$ obs. ilegální hrana, spor!
$\wedge$ je legální

Důsledek: Každá optimální triangulace je D-triangulace.
Důkaz: optimální $\Rightarrow$ legální = Delaunayova

Věta: je-li D-triangulace jediná $\Rightarrow$ je optimální!
Jedy máme jediný D-graf (samé Δ, nedělíme např. $\square$, $\pgon{5}$...)

VÁHOVÝ PROGRAM:

↑ chybí 1 přednáška!

13.12.2007

Podrozdělení roviny podle n přímek v čase $O(n^2)$

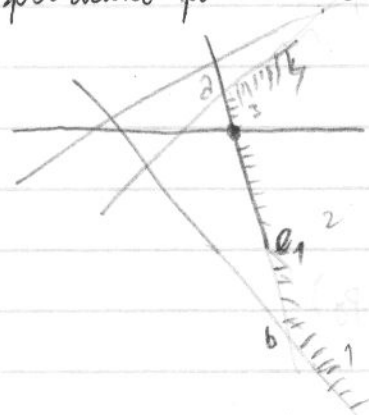
- přidáme k $m-1$ přímkám přidáme další přímku ℓ, počet oblastí přilehlých k této přímce, jejich hran a vrcholů je v $O(m)$

počítáme hrany, které ohraničují přilehlé oblasti k ℓ zleva, těchto hran je $\leq 4n$

indukcí:
$n=1$
IP:

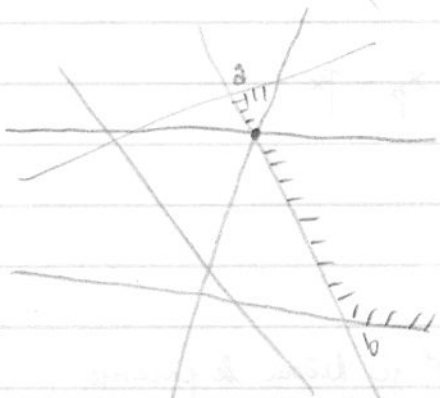
- počet liv. hran $=2$

platí pro $m-1$ přímek. Z m přímek vybereme tu, která prochází průsečíkem nejvíce vpravo



(1) která prů. prochází pouze ℓ_1 přibude nejvýše 4 krajní hran
⇒ celkem $4m$ hran, které jsou levou hranicí

předpokládáme, že žádné 2 přímky nejsou ∥
→ kdyžtak pootočíme a tím vždy přidáme oblast



(2) jedna z nich ℓ_1
přibude nejvýše 4 hrany
celkem krajních hran
$4(m-1)+4 = 4m$

Dualita přímek a bodů (nejsou vertikální)

bod $p = (p_x, p_y) \mapsto$ přímka $p^*: y = p_x x - p_y$

přímka $\ell: y = mx + b \mapsto$ bod $\ell^* : (m, -b)$

Vlastnosti

(1) $p \in \ell \Leftrightarrow \ell^* \in p^*$

$y = mx + b$ $\qquad \ell^* = (m, -b)$

$p_y = p_x x - y = mp_x + b$ $\qquad p^* = p_x x - y$

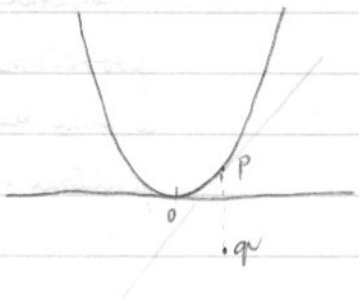
$\rightsquigarrow \ell^* \in p^* : -b = p_x m - p_y$

(2) p leží nad $\ell \Leftrightarrow \ell^*$ leží nad p^*

$p_y > mp_x + b$ $\qquad -b > p_x m - p_y$

Geometrická interpretace

parabola $y = \frac{x^2}{2}$



p je bod na parabole $p = (p_x, p_y)$

tečna k parabole v p je

$$\frac{y - \frac{p_x^2}{2}}{x - p_x} = \left(\frac{x^2}{2}\right)'\Big|_{x=p_x} = x_{|x=p_x} = p_x$$

geometrické odvození

$$y + \frac{p_x^2}{2} = p_x x - p_x^2$$

$$y = p_x x - \underbrace{\frac{p_x^2}{2}}_{p_y}$$

tedy p^* je tečna k parabole v bodě p

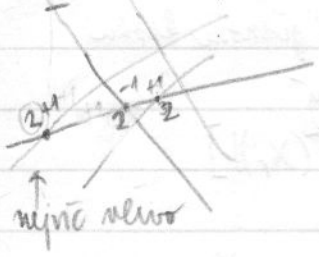
q^* (→ dualní přímka): $y = p_x x - q_y = p_x x + q_y$

Dáno n bodů v rovině. Pro každou přímku $p(i,j)$, i,j z dané množiny - zjistěte kolik bodů leží (nad) pod $p(i,j)$
- naivní přístup ... $O(n^3)$

⇒ duální úloha: dány přímky $l_1 \dots l_n$. Počet bodů ležících nad $l_i \cap l_j$

Předchozí alg. dává podrozdělení roviny jistou přímkami. Ke každému vrcholu tohoto podrozdělení přiřadíme číslo udávající počet přímek nad tímto vrcholem.

Postupně procházíme přímky zleva doprava a přidělujeme čísla.



pro všechny přímky ~ $O(n^2)$

} využití duality

Přírůstkové náhodné algoritmy
- průnik půlrovin (úloha lin. programování) A
- vybarvení jednoduché mapy B
- Delaunayova triangulace C

Konfigurační prostor (X, Π, D, k) $X = \{x_1, x_2 \dots x_n\}$ náhodná pos. $X_r = \{x_1, \dots x_r\}$
X - množina geom. objektů
A: množina (v obecné poloze) půlrovin B: množina úseček C: množina bodů
Π - množina konfigurací $\Delta \in \Pi$
A: průsečíky hraničních přímek půlrovin
B: jednoduché mnohoúhelníky C: trojúhelníky
D - zobrazení $\Pi \to \text{subset}(X)$, $\Delta \in \Pi$
A: $D(\text{průsečíky}) = \{$ poloroviny, jejichž hranice prochází bodem $p\}$
B: $D(\text{jednoduchý mnohoúhelník}) = \{$ úsečky určující jednoduchý mnohoúhelník $\}$
C: $D(\Delta) =$ vrcholy Δ
d - maximum počtu prvků ... $\max\{\text{card } D(\Delta), \Delta \in \Pi\}$ konečné

C: $d=3$ B: $d=4$ A: $d=2$

K – konfliktní zobrazení ; $K: \Pi \to 2^X$

$K(\Delta)$ jsou objekty z X v konfliktu s Δ

C: $K(\Delta^{cela}) = \{$všechny body ležící uvnitř kružnice opsané $\Delta\}$

B: $K(\Delta^{delta}) = \{$ všechny úsečky mající průnik s vnitřkem lichoběžníku $\Delta^{delta}\}$

A: $K(\Delta^{delta\text{-}bod}) = \{$ všechny polorovny neobsahující Δ bod $\}$

$S \subseteq X: \rightsquigarrow \mathcal{T}(S) = \{\Delta \in \Pi \mid D(\Delta) \subseteq S, K(\Delta) \cap S = \phi\}$ (konfig. jsou nekonfliktní se zbylými objekty)

C: $\mathcal{T}(S)$ – všechny trojú. takové, že v žádném obsaném kružnici neleží žádné jiné body z S, tzv $\mathcal{T}(S)$ – Delaunayova triangulace pro S

B: $\mathcal{T}(S)$ – soubor lich. tvořící lich. mapu pro S

A: $\mathcal{T}(S)$ – body ležící na všech polorovinách v S

Věta 1: Nechť (X, Π, D, K) je konfigurační prostor ⇒ očekávaný počet konfigurací v $\mathcal{T}(X_r) - \mathcal{T}(X_{r-1})$ (počet konf. přidaných v jednom kroku) je nejvýše $\frac{d}{r} E[\text{card}(\mathcal{T}(X_r))]$ kde $d = \max\{\text{card}(D(\Delta), \Delta \in \Pi\}$.

Důkaz: Kolik konfigurací zmizí z $\mathcal{T}(X_r)$ při odebrání objektu x_r z X_r.

$\sum_{x \in X_r} \text{card}\{\Delta \in \mathcal{T}(X_r) \mid x \in D(\Delta)\} \leq \text{card}\,\mathcal{T}(X_r) \cdot d$ (Δ zmizí po odebrání 3 bodů)

... střední hodnota: $\frac{d}{r} E(\text{card}\,\mathcal{T}(X_r))$

C – aplikace věty 1

$\leq 3 | r\,(2(r+3)-5) = \frac{6r+3}{r} \approx 6$

B $\leq \frac{4}{r}[3r+1] = \frac{12r+4}{r} \approx 12$

A $\leq \frac{2}{r}[r] \leq 2$