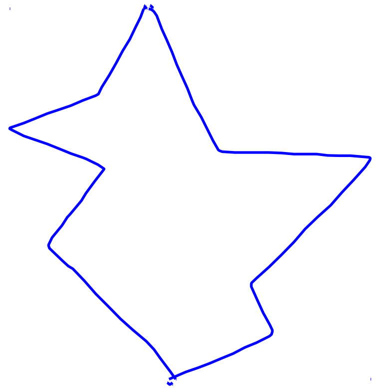
TRIANGULACE MNOHOÚHELNÍKA

- rozdělení na monotónní mnohoúhelníky



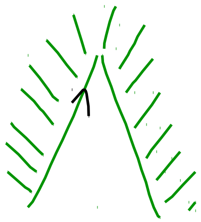
cesta dolů znamená, že pořadí vrcholů je sestupné v lexikografickém uspořádání

$p < q \iff p_y > q_y$ nebo $p_y = q_y$ a $p_x < q_x$

- rozdělení mon. mnohoúhelníků na trojúhelníky

Rozdělení na mon. části znamená doplnit spojnice vrcholů, abychom zrušili split a merge vrcholy.

②

split 

merge 

Metoda – zametací přímka.

Fronta událostí .... uspořádání vrcholů

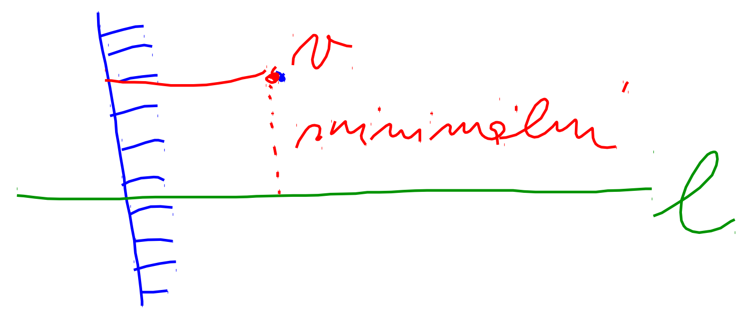
Binární vyhl. strom má v listech uspořádání

stran mnohoúhelníka protínaných zametací přímkou.

Přitom uvažujeme pouze hrany, které mají mnohoúhelník zprava.

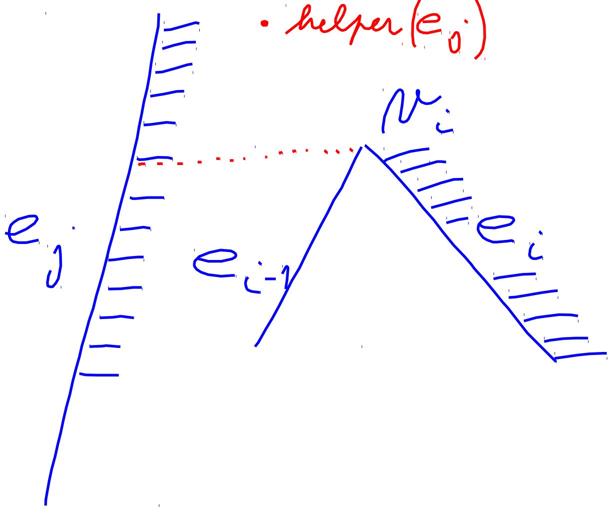
Pro tyto hrany definujeme helper

helper(e) = vrchol v



Minule procedury pre start a end vrcholy

SPLIT VRCHOL



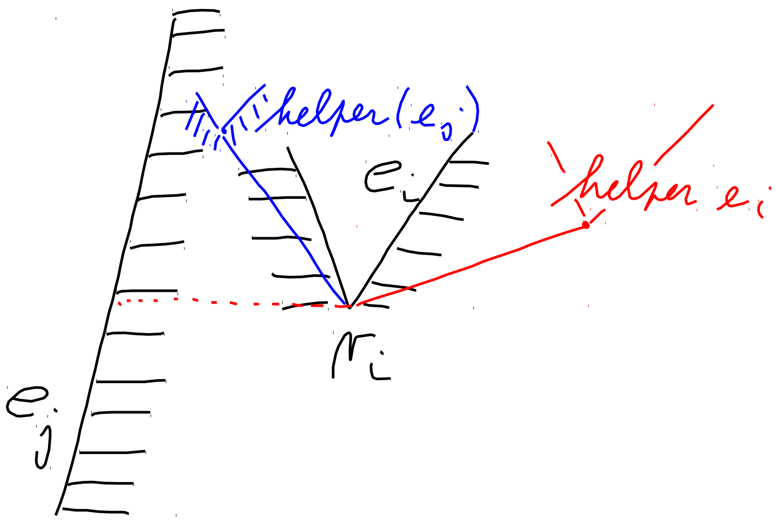
(1) spojíme v_i s helperom(e_j)

(2) $v_i \longrightarrow$ helper(e_j)

(3) vložíme e_i do stromu T

(4) $v_i \longrightarrow$ helper(e_i)

MERGE VRCHOL



(1) Jestliže helper(e_i) je merge, spoj v_i s helperem(e_i).

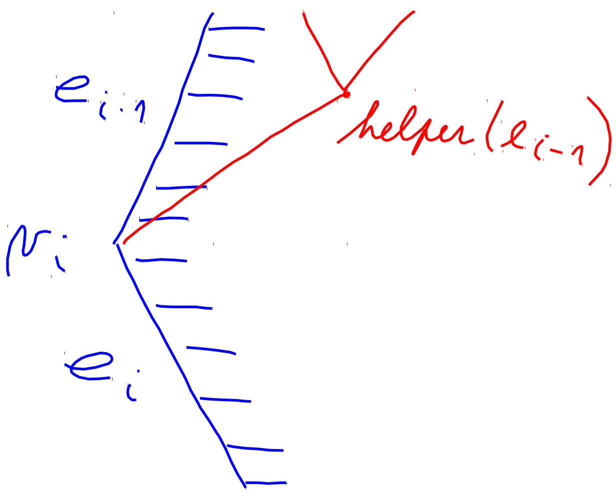
(2) e_i vypustíme ze stromu T

(3) Jestliže helper(e_j) je merge, spoj v_i s helperem(e_j).

(4) $v_i \longrightarrow$ helper(e_j)

5

REGULAR v P vpravo



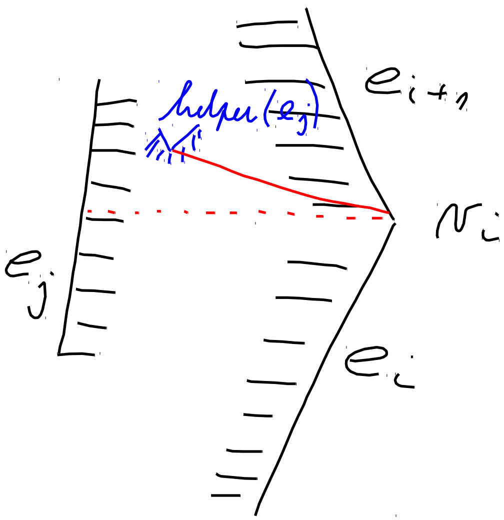
(1) Je-li helper(e_{i-1}) merge,
spoj v_i s helperem(e_{i-1})

(2) e_{i-1} vyndáme ze stromu $\mathcal{T}$

(3) e_i vložíme do stromu $\mathcal{T}$

(4) $v_i \longrightarrow$ helper(e_i)

REGULAR P zleva



(1) je-li helper(e_j) merge, spoj v_i s helperem(e_j).

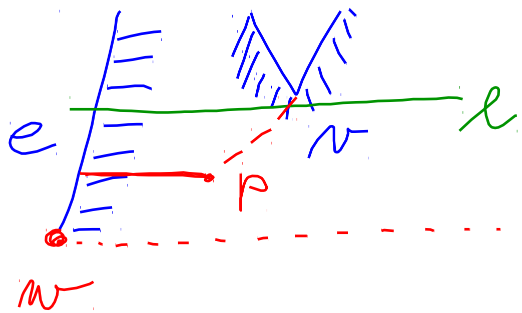
(2) $v_i \longrightarrow$ helper(e_j)

Lemma: Algoritmus rozdělí mnohoúhelník na monotónní mnohoúhelníky.

Dk: (1) Odstraníme split a merge vrcholy.

Split vrcholy odstraňujeme, když jimi prochází zametací přímka.

Odstranění merge vrcholů



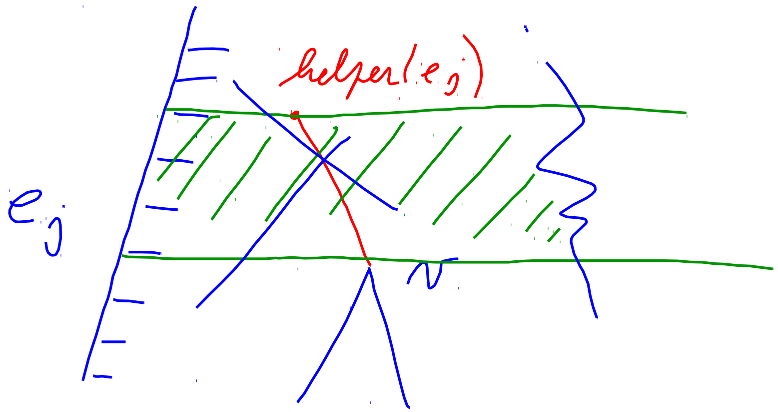
v je merge vrchol. Pak je helper pro nějaké e

Najdeme vrchol p, který algoritmus spojuje s v.

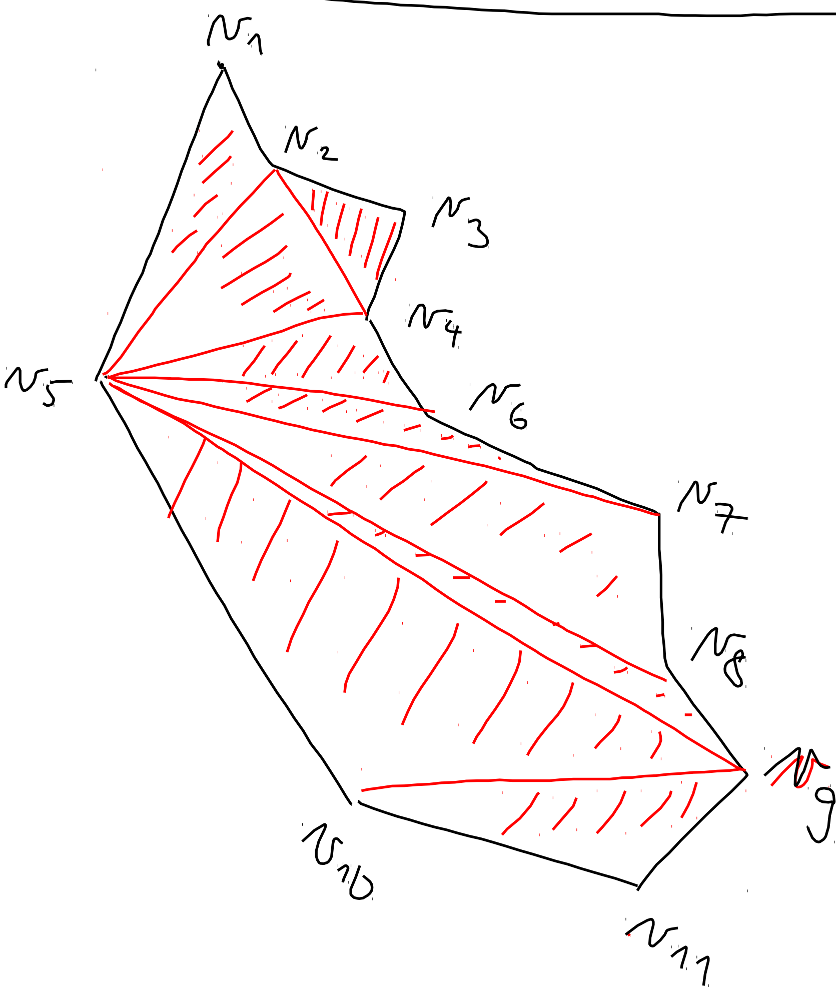
Ještě bychom měli dokázat, že nové přidávané hrany se neprotínají. To je potřeba dělat indukcí. Předpokládáme, že co je nad zametací přímkou, to se neprotíná, a když dojdeme do vrcholu v, tak v závislosti od typu musíme ukázat, že ani po provedení procedury na tento krok se nic neprotíná.

⑧

<u>v is split</u>



TRIANGULACE MONOTÓNNÍCH MNOHOÚHELNÍKŮ



Seřadíme vrcholy z levé a pravé cesty v lexikografickém uspořádání.

Místo pointy zde budeme mít tzv. zásobník (stack)

Ten na začátku je

$$\begin{pmatrix} v_2 \\ v_1 \end{pmatrix}.$$

Zásobník po příchodu vrcholem v_{j-1} obsahuje nějaké vrcholy

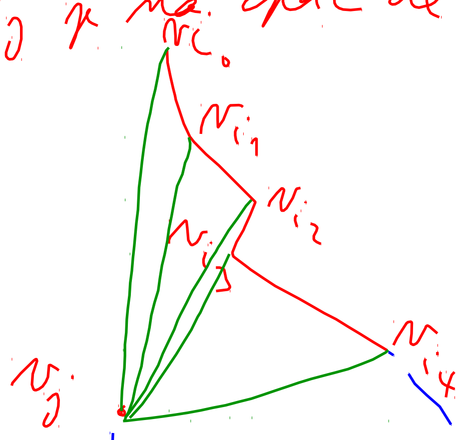
$$\begin{pmatrix} v_{i_0} \\ v_{i_1} \\ v_{i_2} \\ \vdots \end{pmatrix}$$

Tyto vrcholy leží v levé nebo pravé cestě mnohoúhelníka, ze kterého jsme odřízli již vytvořené trojúhelníky.

Další vrchol v našem pořadí nechť je v_j. Mohou nastat 2 možnosti

$i_0 > i_1 > \dots$

(1) v_j je na opačné straně než vrcholy v zásobníku

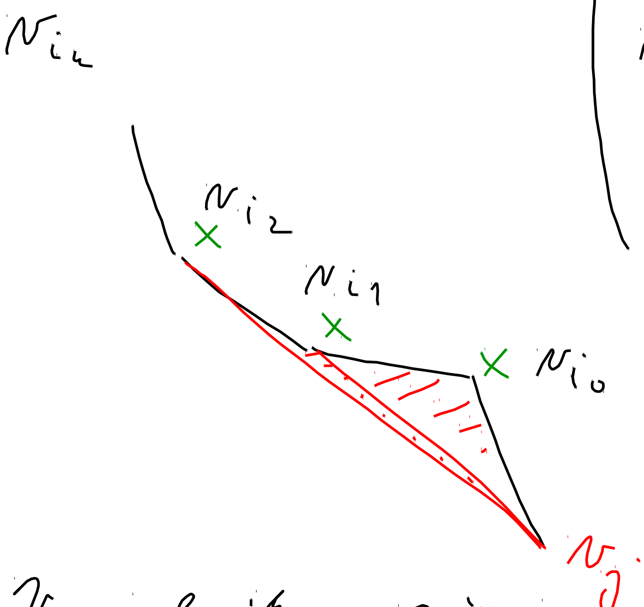


Všechny vrcholy v zásobníku spojíme s v_j. Nový zásobník bude

$$\begin{pmatrix} v_j \\ v_{i_4} \end{pmatrix}$$

(2) v_j leží na stejné straně jako vrcholy v zásobníku.

Nechť zásobník je $\begin{pmatrix} v_{i_0} \\ v_{i_1} \\ \vdots \\ v_{i_k} \end{pmatrix}$



Zkoumáme spojit v_j s v_{i_1}. Leží-li tato spojnice v mnohoúhelníku vytvoříme $\triangle(v_j, v_{i_0}, v_{i_1})$. Toto provádíme s vrchol v_{i_2}, atd., dokud je to možné. Vždy vyndáme v_i ze zásobníku.

V zásobníku nic zbude, k těmto vrcholům doplníme poslední vyndaný vrchol a vrchol v_j $\begin{pmatrix} v_j \\ v_{i_s} \\ v_{i_{s+1}} \\ \vdots \end{pmatrix}$

Časová náročnost je $O(n)$, kde n je počet uzlů.

Algoritmus zapsán pseudokódem číslo 13 v pseudo.pdf.

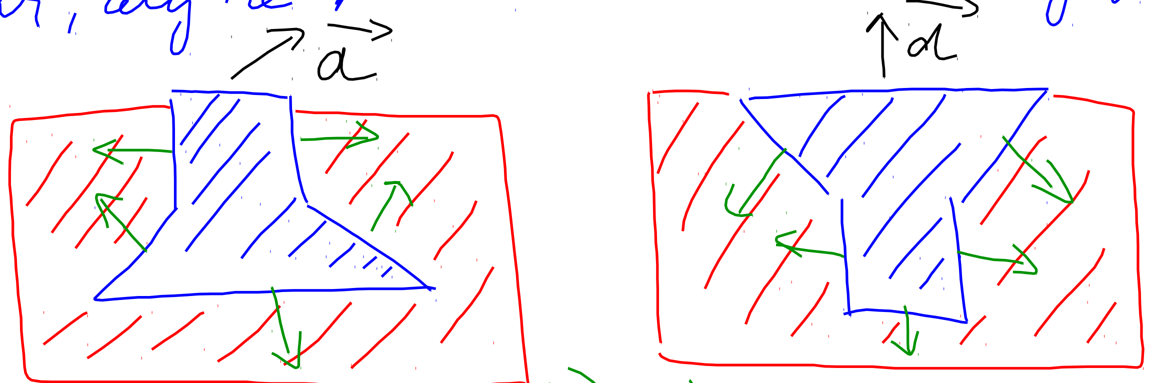
Průnik polorovin

- s lineárním časem řešíme úlohu lin. programování v rovině.

Motivace

Slévárna, formy pro odlitky.

Odlitky jsou mnohostěny. Chceme je z formy vytáhnout tak, aby se forma nerozbila. Kdy je to možné?



$\sphericalangle(\vec{d}, \vec{n}) \geq 90°$

$\vec{n}$ normálový vektor stěny. Jestliže pro všechna $\vec{n}$ platí $\sphericalangle(\vec{d}, \vec{n}) \geq 90°$, lze odlitek vytáhnout ve směru $\vec{d}$.

$$\vec{d} = (d_x, d_y, 1)$$

$$\sphericalangle(\vec{d}, \vec{n}) \geq 90^\circ \iff \text{skalární součin } \langle \vec{d}, \vec{n} \rangle \leq 0.$$

$$d_x n_x + d_y n_y + n_z \leq 0 \quad \forall \text{ normálové vektory.}$$

Hledáme řešení soustavy nerovnic

$$a_{i1} x + b_{i1} y \leq c_i \qquad i = 1, 2, 3, \dots n$$

Geometricky jde o to najít bod z průniku polorovin v rovině.

- 1. úloha najít celý průnik → průnik polorovin
- 2. úloha najít nějaký bod z průniku → úloha lin. programování v rovině

Průnik poloroviny:

$H = \{h_1, h_2, \dots h_n\}$ množina polorovin v rovině

$H = H_1 \cup H_2 \qquad H_1 = \{h_1 \dots h_{\lceil \frac{n}{2} \rceil}\}$

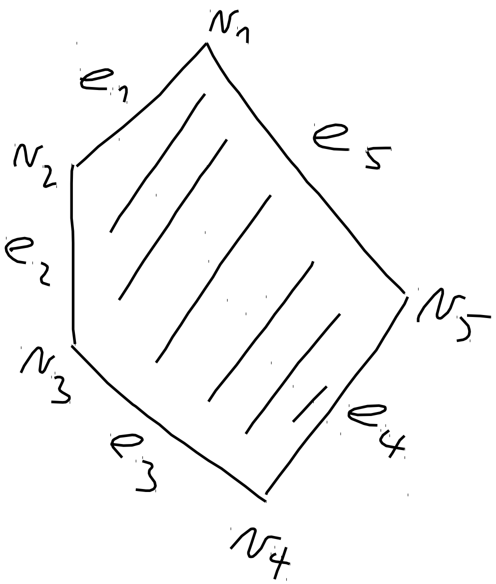
$H_2 = H \setminus H_1.$

Najdem

$$C_1 = \bigcap_{h_i \in H_1} h_i \qquad C_2 = \bigcap_{h_i \in H_2} h_i$$

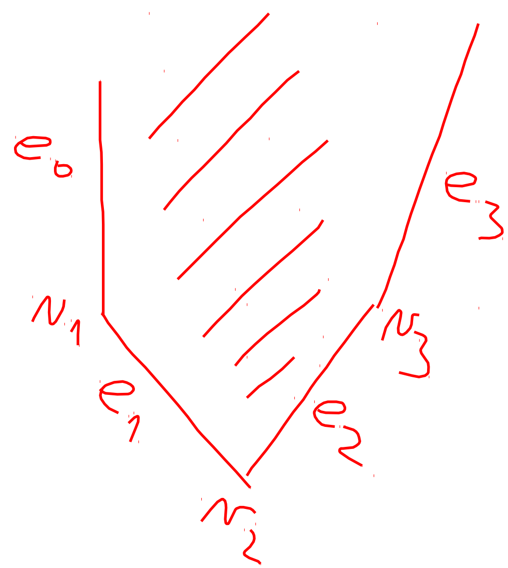
$$C = \bigcap_{h \in H} h = C_1 \cap C_2.$$

Průnik polorovin je konvexní množina popsaná levou a pravou hranicí :



levá hranice $v_1, e_1, v_2, e_2, v_3, e_3, v_4$

pravá hranice v_1, e_5, v_5, e_4

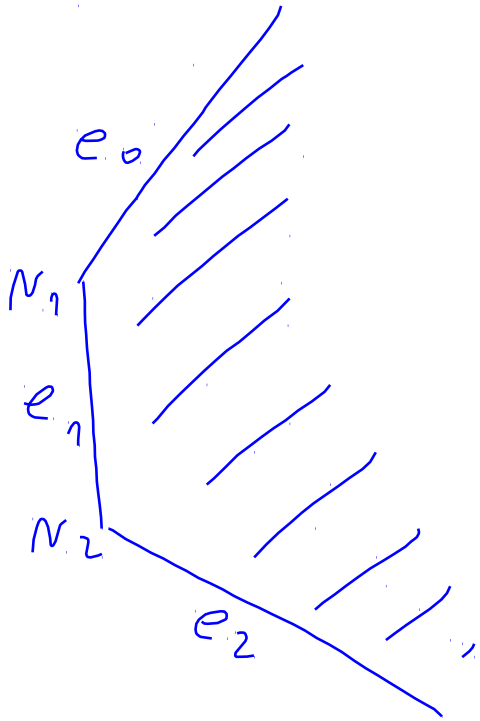


levá hranice

e_0, v_1, e_1, v_2

pravá hranice

e_3, v_3, e_2, v_2



levá hranice e_0, v_1, e_1, v_2, e_2

pravá hranice $\emptyset$

Jde o to najít algoritmus pro nalezení levé a pravé hranice $C_1 \cap C_2$, pokud známe levé a pravé hranice C_1 a C_2.

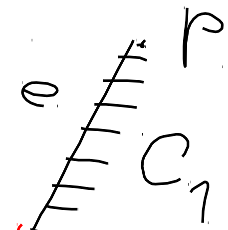
Vrcholy hranic seřadíme lexikograficky naším známým způsobem.
Procházíme je a určujeme, které hrany a vrcholy tvoří levou nebo pravou hranici průniku.

Přechod přes vrchol p konvexní oblasti C_1.

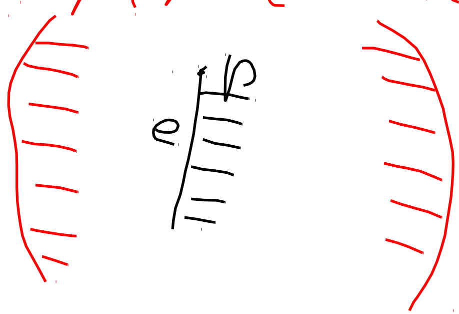
Nechť p leží v levé hranici C_1. Z p vychází hrana e.

I p leží mezi levou a pravou hranicí C_2

e neprotíná hranici C_2.



V tomto případě zařadíme p a e do levé hranice $C = C_1 \cap C_2$

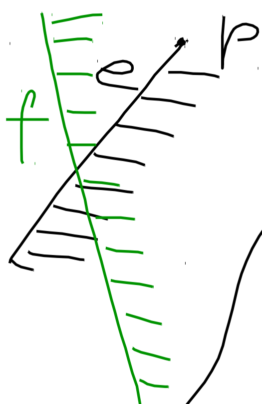


II. p leží mezi levou a pravou hranicí C_2 a e jediná

(a) levou hranicí C_2

(b) pravou hranicí C_2

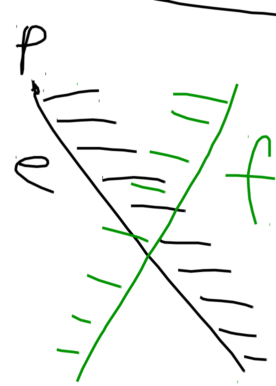
II (a)



$p, e, e \cap f, f$ bude v levé hranici

$C = C_1 \cap C_2$

II (b)



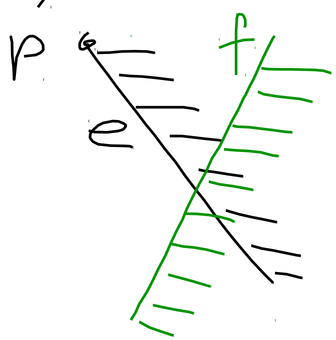
$p, e, e \cap f$ je konec levé hranice C

$f, e \cap f$ je konec pravé hranice C

Analogicky $\underline{\text{III}}$ p neleží mezi hranicemi a e nejediná hranici C_2 – nic se neděje

$\underline{\text{IV}}$ p ~~lez~~ neleží mezi hranicemi a e jediná

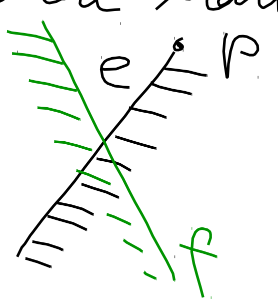
(a) levou hranici C_2



levá hranice $C = C_1 \cap C_2$ bude mít na f pokračování

$e \cap f$, e

(b) pravou hranici



levá hranice $C = C_1 \cap C_2$ začíná

$e \cap f$, e

pravá hranice $C = C_1 \cap C_2$ začíná

$e \cap f$, f

Časová náročnosť algoritmu pre $C_1 \cap C_2$ je $O(n_1 + n_2)$, kde n_i je počet vrcholov v C_i.

Časová náročnosť pre prienik n polrovín nech je $T(n)$. Platí

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

Riešení tohoto rekurentného vzťahu

$$T(n) = O(n \log n).$$