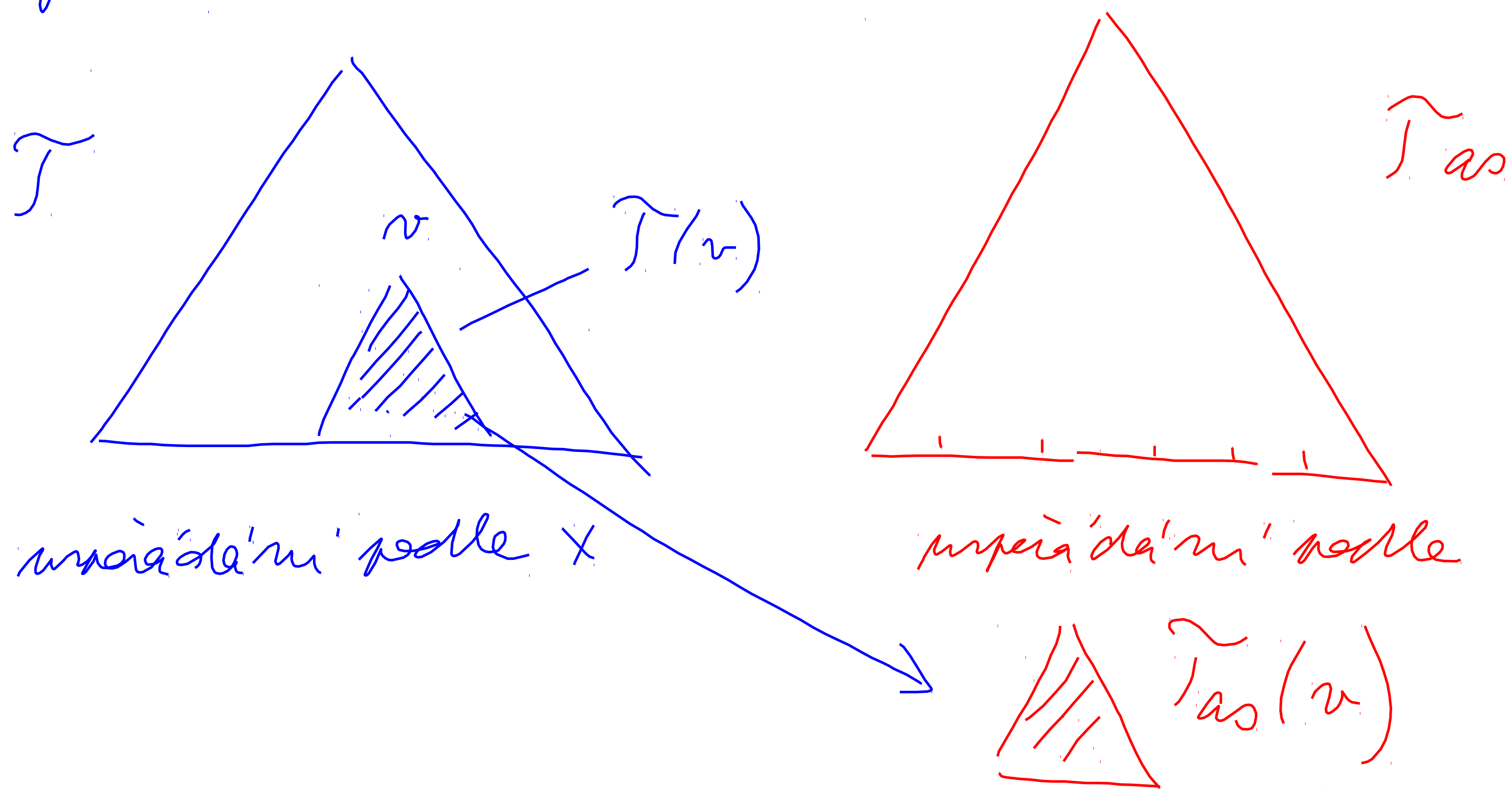
RANGE TREES

Vyhledávací struktura

$P = \{p_1, p_2, \dots, p_n\}$

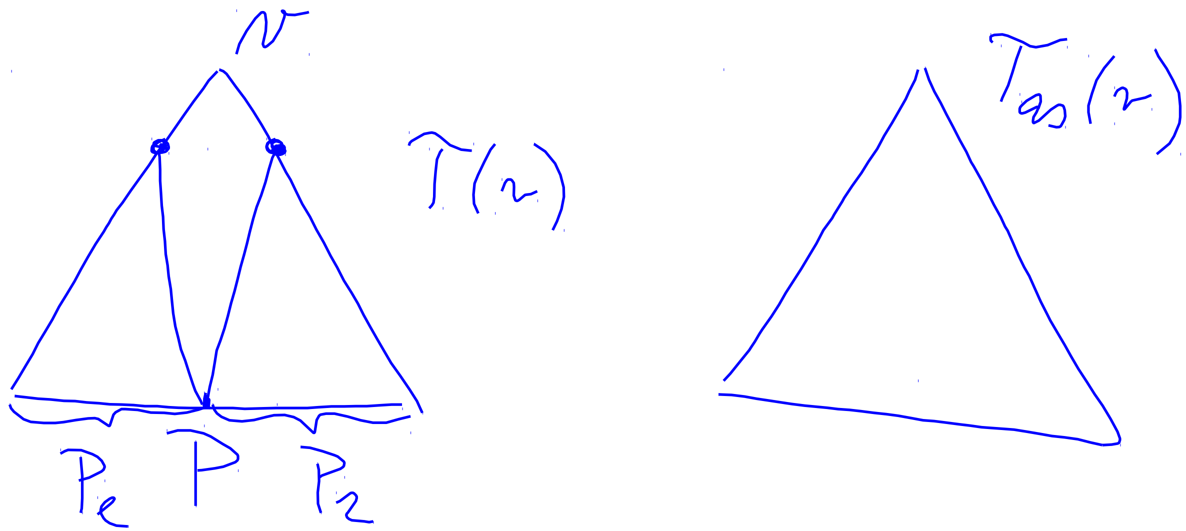


uspořádání podle y

S listy stejnými jako v $T(v)$

Algoritmus pro vytvoření stromu

- vstup množina P
- výstup je kořen range tree



Čas potřebný na konstrukci je dán rekurentně

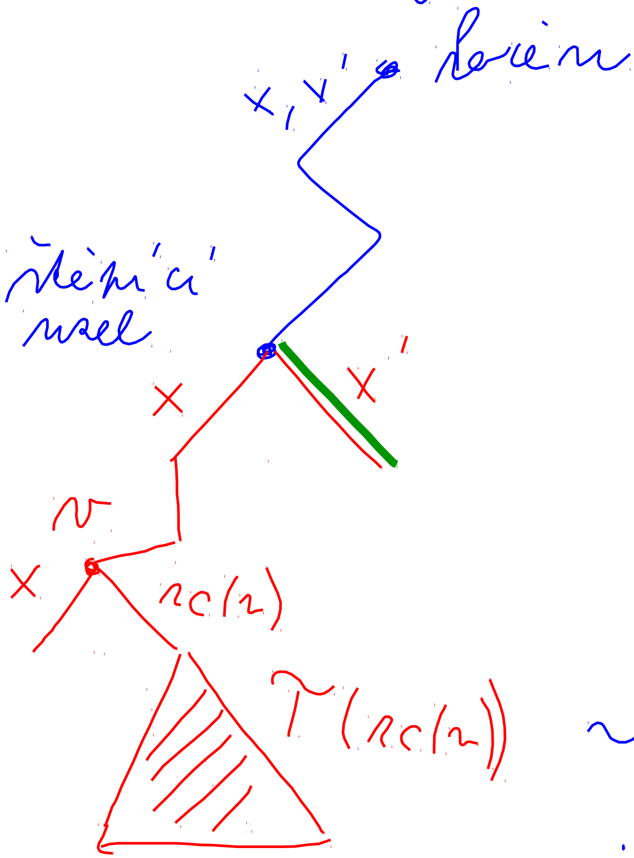
$$T(n) = 2T\left(\frac{n}{2}\right) + O(n) \Rightarrow T(n) = O(\log n \cdot n)$$

Na začátku uděláme uspořádání podle y.

3

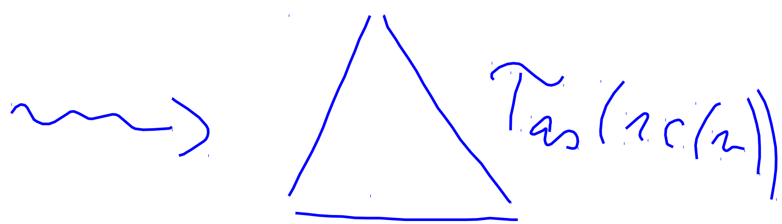
Vyhledávání v range tree – podobné jako v dim 1.

$[x, x'] \times [y, y']$ potom podle y



analogický postup pro x'

– viz algoritmus v e-learningu



jednodimenzionální vyhledávání v $T_{as}(rc(v))$ podle y

Časová náročnost vyhledávání

Procházíme cestu od kořene k listu přes vrcholy v zvlášť pro x, tak i pro x'.

V každém vrcholu může skončit podle y asociovaný pravý (pro x) nebo levý (pro x') podstrom, ve kterém najdeme k_v bodů v obdélníku.

Čas ve vrcholu v je nejvýše $O(k_v + \log n)$

Celkem $O\left(\sum_v k_v + \log n\right) = O(k_{v_1} + \dots + k_{v_{\log n}} + \log n \cdot \log n)$
$= O(k + \log^2 n)$

Věta Celkový čas na vyhledávání je

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

n počet bodů množiny P
k počet bodů v obdélníku

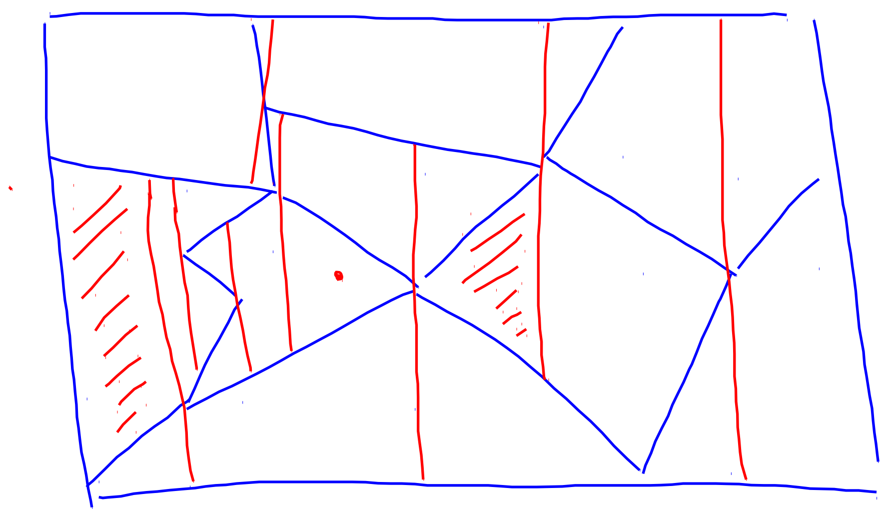
Obecně v dimenzi d je situace následující:

	kd-stromy	range trees
paměť	$O(n)$	$O(n \log n)$
čas konstrukce	$O(n \log n)$	$O(n \log^{d-1} n)$
čas vyhledání	$O(n^{1-\frac{1}{d}} + k)$	$O(\log^{d-1} n + k)$

6

LOKALIZACE BODU

K dané mapě chceme najít vyhledávací strukturu, která při zadání bodu určí, v které oblasti bod leží.



Příslušná vyhledávací struktura se nazývá lichoběžníková mapa.

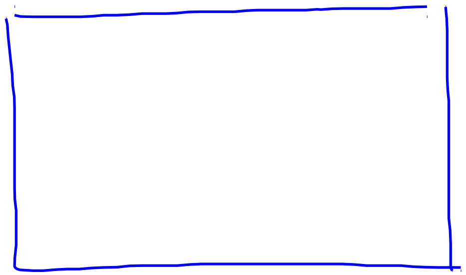
Z daného rovinného podrozdělení vytvoříme podrobnější podrozdělení, které jako oblasti bude mít jenom lichoběžníky a trojúhelníky.

Vrcholy, kromě vnějšího obdélníka, ve kterém se vše odehrává, mají různé x-ové souřadnice. Časem ukážeme, jak předpoklad odstranit.

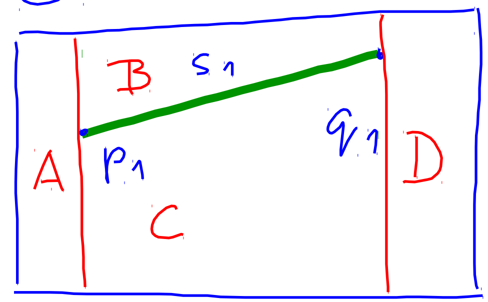
Konstrukci lichoběžníkové mapy a vyhledávání struktury provedeme pro množinu úseček $S = \{s_1, s_2, \dots s_n\}$ daného rovinného podrozdělení.

Konstrukce je přírůstková. Předpokládáme, že úsečky leží v pravoúhelníku R. Tento pravoúhelník je lichoběžníkovou mapou pro prázdnou množinu úseček. Je-li $S_i = \{s_1, \dots s_i\}$ a $T(S_i)$ lich. mapa na S_i, pak vytvoříme z této mapy přidáním úsečky s_{i+1} lich. mapu pro $S_{i+1} = \{s_1, s_2, \dots s_{i+1}\}$.

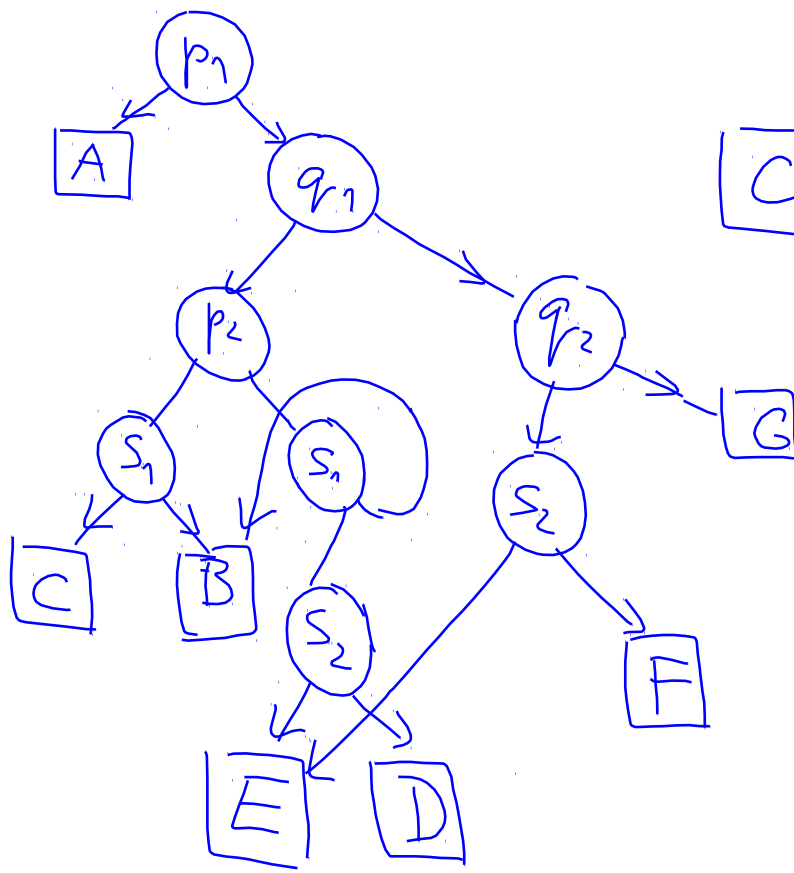
R



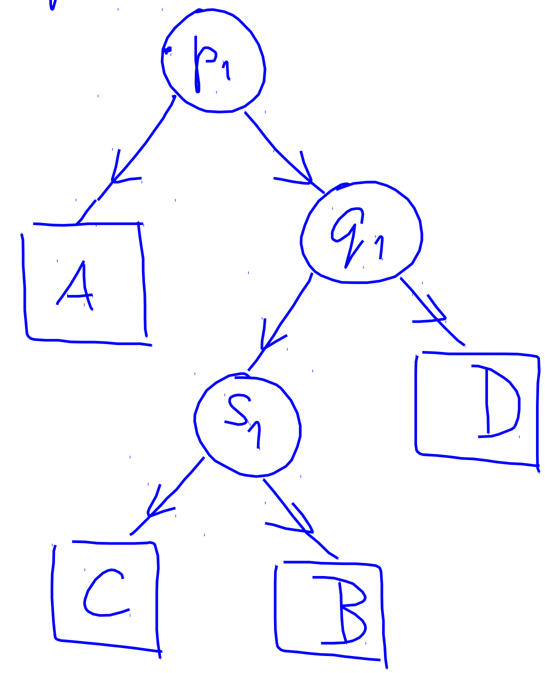
8

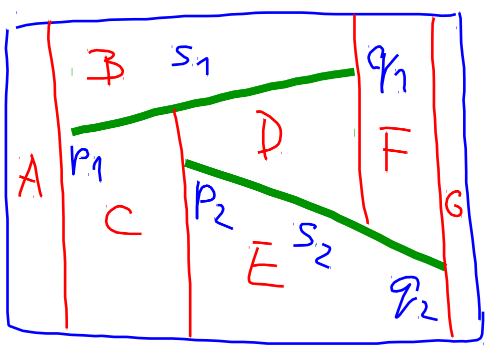


lich. mapa



vyhl. struktura





$\mathcal{T}(S_2)$

$\mathcal{D}(S_2)$

Vyhledávací struktura $\mathcal{D}(S)$ je orient. graf z každého uzlu vycházejí dvě hrany. Uzly jsou buď úsečky nebo jejich krajní body. V listech jsou lichoběžníky.

Úsečka s_i .... levý krajní bod p_i

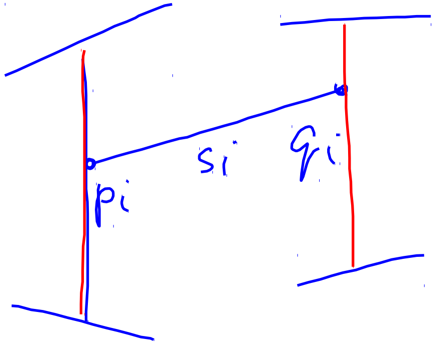
pravý krajní bod q_i

<u>Lemma</u>: Lichoběžníková mapa pro n úseček má nejvýše $6n+4$ vrcholů a $3n+1$ lichoběžníků.

<u>Dk</u>: Počet vrcholů – ohraničující R má 4.

Konc. body úseček je nejvýše $2 \times n = 2n$

(Úsečky se neprotínají ve vnitřních bodech, ale mohou mít společné koncové body.)

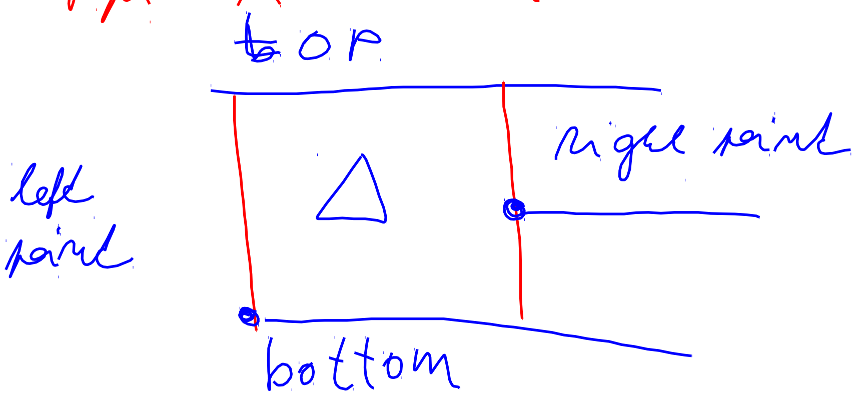


Vertikály pro každý konc. bod mohou vytvořit další dva body – celkem $2 \cdot 2n = 4n$.

Celkem uzlů $6n + 4$.

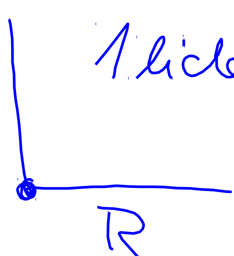
Před počítáním lichoběžníků zavedeme několik pojmů.

Popis lichoběžníka

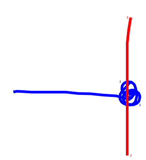


Každý lichoběžník je popsán těmito 4 údaji

Počítání lichoběžníků – kolik lichoběžníků máme pro daný levý bod

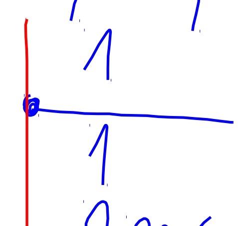


1 lichoběžník



1 lichoběžník

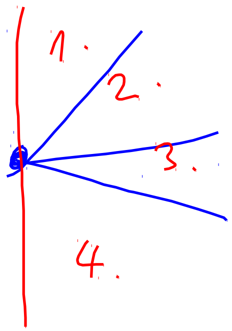
celkem n



1

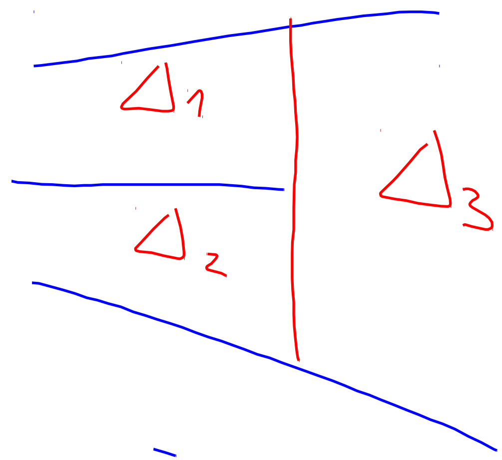
1

celkem $2n$ $\leq 3n+1$



odhad 2·3, složitost 4 lichoběžníky.

Sousední lichoběžníky mají společnou vertikální úsečku.



Δ_3 je pravý horní soused lich. Δ_1

Δ_2 je levý dolní soused lichoběžníku Δ_3

Algoritmus rámcově

Vezmeme náhodné pořadí úseček $s_1, s_2, \ldots, s_n$.

Z lich. mapy $T(S_i)$ a vyhledávací struktury $D(S_i)$ přidáním s_{i+1} vytvoříme $T(S_{i+1})$ a $D(S_{i+1})$.

(12)

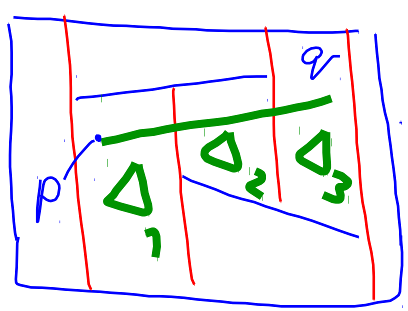
<u>Postup</u>: (1) Najdeme lichoběžníky v $T(S_i)$, kterými prochází úsečka s_{i+1}.

(2) V $T(S_i)$ zrušíme tyto lichoběžníky a nahradíme novými.

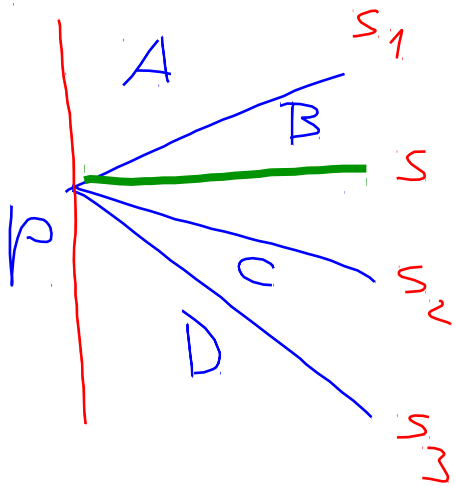
(3) V $\mathcal{D}(S_i)$ zrušíme listy pro lichoběžníky nalezené v (1) a na jejich místo doplníme graf pomocí podgrafů s listy nalezenými v (2).

<u>Algoritmus na (1)</u>   FOLLOW SEGMENT

– máme $T(S_i)$ a $\mathcal{D}(S_i)$. Přidáme $s = s_{i+1}$, s konc. body $p = p_{i+1}$ a $q = q_{i+1}$.



S prochází lichoběžníky $\Delta_1, \Delta_2, \ldots \Delta_k$. Jak je najdeme.
Pomocí $\mathcal{D}(S_i)$ zjistíme, ve kterém lichoběžníku leží bod p, pokud p není nalevo některé úsečky z S_i.
Pokud p je nalevo (nějaké) své úsečky, postupujeme následovně

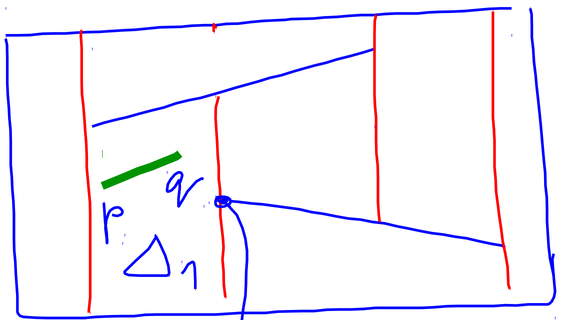


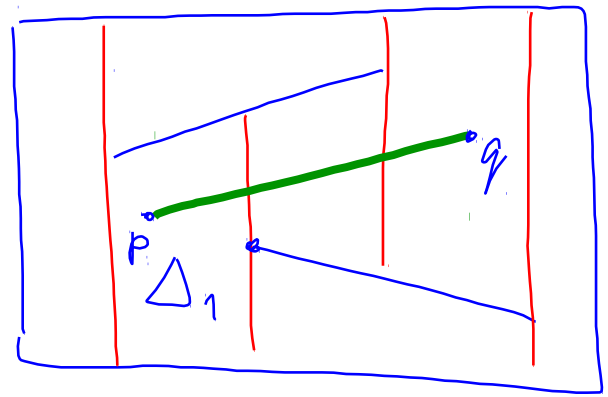
$\Delta_1 = B$ neboť směrnice úsečky s leží mezi směrnicemi úseček s_1 a s_2.

směrnice s je

$$\frac{q_{2y} - p_{2y}}{q_{2x} - p_{2x}} < \frac{q_y - p_y}{q_x - p_x} < \frac{q_{1y} - p_{1y}}{q_{1x} - p_{1x}}$$

Hledáme Δ_2





jestliže $q_x <$ x-ová souřadnice $rp(\Delta_1)$

tak s leží celá v Δ_1

$q_x > rp(\Delta_1)_x$

Δ_2 je pravý soused Δ_1

horní, jestliže $rp(\Delta_1)$ leží pod s

dolní, jestliže $rp(\Delta_1)$ leží nad s

atd.