ORTOGONÁLNÍ VYHLEDÁVÁNÍ

Minule tzv. kd stromy

Dnes <u>range trees</u>

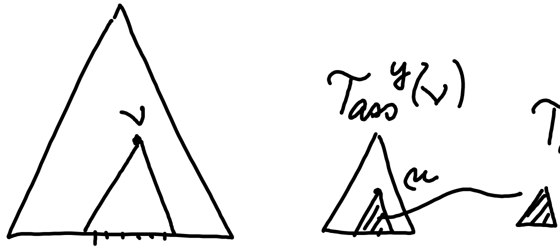
Range tree je složen z jistého binárního vyváženého stromu T, jehož listy jsou body množiny P v rovině seřazené podle souřadnice x (Pro začátek předpokládáme, že každé dva body mají různou souřadnici x i y.)

$[x, x']$

$T_{ass}(v)$

$[y, y']$

~~kd~~ range tree v dim 3



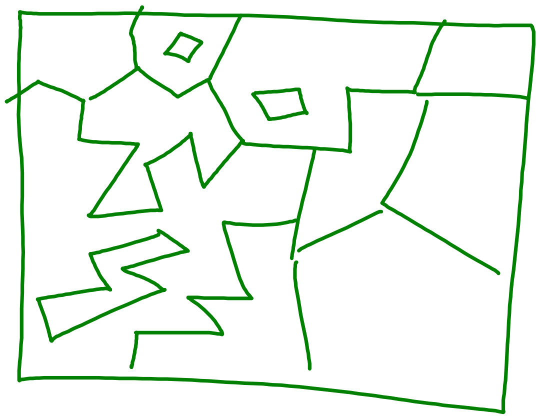
$T_{ass}^{y}(v)$ $T_{ass}(v, u)$

podle x podle y podle z

alg 27

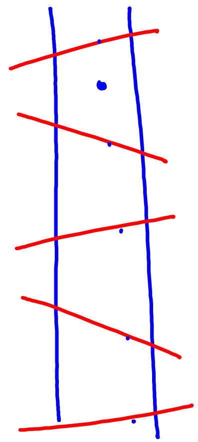
	kd stromy	range tree
vyhledání	$O(n^{\frac{1}{2}} + k)$	$O(\log^2 n + k)$
$n = 10^6$	$10^3 + k$	$36 + k$

LOKALIZACE BODU



Pro danou mapu vytvořit vyhledávací strukturu, která pro zadaný bod q najde oblast ve které bod leží.

1. pozorování – v dimenzi 1 se vyhledá dobře



Vyhledávání v tomto pásu je vlastně 1-dimenzionální

Předpoklady

S je množina n úseček $s_1, s_2, \dots s_n$.

Úsečky se protínají (pokud vůbec) pouze v koncových bodech.

Žádné dva koncové body úseček nemají stejnou souřadnici x

Množinu S umístíme do pravoúhelníku R.

Postupně vytvoříme lichoběžníkovou mapu T a vyhledávací

strukturu D pro množinu $\{s_1, s_2, \dots s_i\}$

Lichoběžníková mapa T(S).. koncovými body úseček vedeme vertikální úsečky k nejbližší vyšší úsečce a nejbližší nižší úsečce.

(Tyto úsečky jsou buď z S nebo z R.)

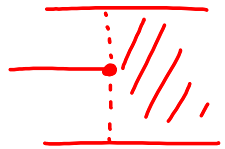
Lemma: Lichoběžníková mapa pro n úseček obsahuje nejvýše $6n+4$ uzlů a nejvýše $3n+1$ lichoběžníků.

Dk. Počet bodů $\le$ uzly R + konc. body úseček + nové vytvářené body

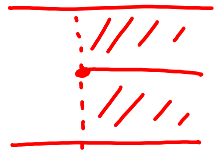
$$\underbrace{4 \qquad 2n \qquad 2(2n)}_{6n+4}$$

Kolik lichoběžníků je určeno leftpointem?

R



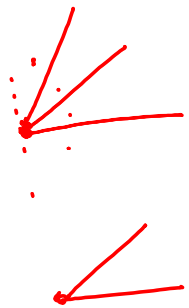
1



2

Celkově $1 + n\cdot 1 + n\cdot 2 = 3n+1$.

počítáme-li ho $k\times$ mění $k+1$ lichoběžníků



$k+1 < 2k$

Algoritmus vytváří lichoběžníkovou mapu ~~pozn~~ a vyhledávací strukturu
pro množinu $\{s_1, s_2, \dots s_i\}$ postupně

Lichoběžníková mapa pro prázdnou množinu obsahuje pouze pravoúhelník R
a vyhledávací struktura obsahuje pouze jediný list.

Algoritmus nám říká jak od $\mathcal{T}(s_1, \dots s_{i-1})$ a $\mathcal{D}(s_1 \dots s_{i-1})$
přejít ke $\mathcal{T}(s_1, \dots s_i)$ a $\mathcal{D}(s_1, \dots s_i)$.

p_i je levý koncový bod úsečky s_i

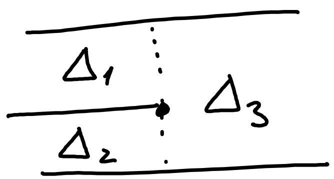
q_i je pravý koncový bod úsečky s_i

Celkový algoritmus (28)

Co se děje po přidání nové úsečky?

Potřebujeme pojem sousedního lichoběžníka

Sousední lichoběžníky jsou ty se společnou vertikální stranou



sousední Δ_1 a Δ_3 společný top

Δ_2 a Δ_3 společný bottom

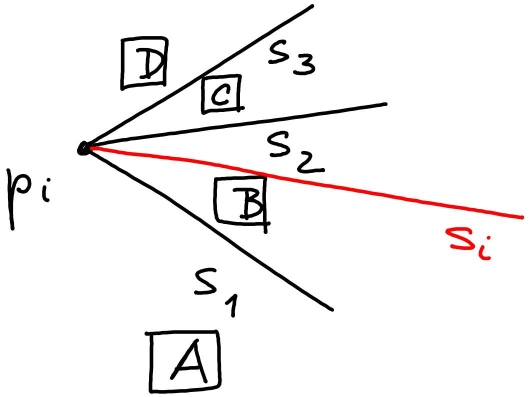
Δ_1 je horní ~~pravý~~ levý soused pro Δ_3

Δ_2 je dolní levý soused pro Δ_3

Δ_3 je pravý horní soused pro Δ_1,

Δ_3 je pravý dolní soused pro Δ_2

Nechť např. nastane taková situace



Porovname směrnici úsečky s_i se směrnicemi úseček s_1, s_2, s_3. Zjistíme, že

směrnice s_1 < směrnice s_i < směrnice s_2.

Můžeme tedy usoudit, že s_i protíná lichoběžník B jako první.