

Aula 23

Buscas ortogonais

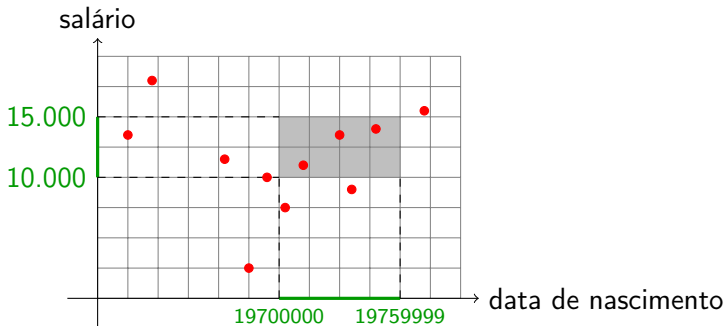
Cap 5 do livro de de Berg et al.

Consultas em bancos de dados

BD com informações sobre funcionários de uma empresa: nome, endereço, data de nascimento, salário, etc.

Exemplo de consulta: liste todos os funcionários nascidos entre 1970 e 1975 com salário entre R\$10.000,00 e R\$15.000,00.

Interpretação geométrica: cada funcionário é um ponto, com data de nascimento, codificada como um número, como x -coordenada, e salário como y -coordenada.

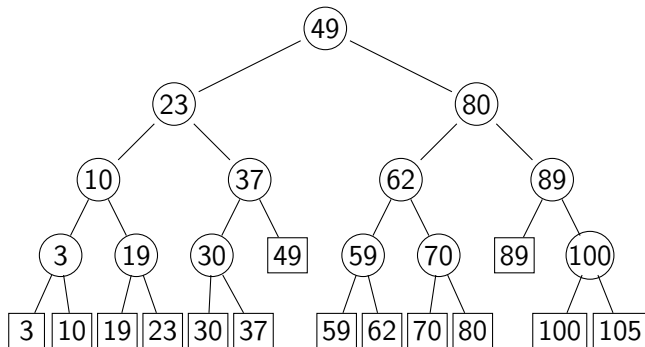


Busca unidimensional por intervalo: 1D range queries

S : conjunto de n números

ABB com números de S nas folhas

Nós internos guardam máximo da subárvore esquerda.



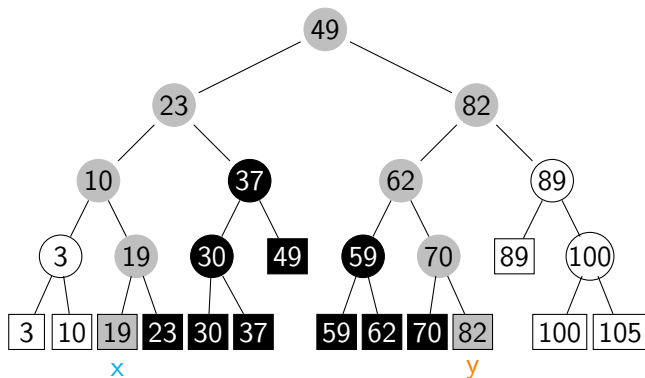
Valor num nó interno divide as folhas aproximadamente ao meio.

Busca unidimensional por intervalo

Busca por números de S no intervalo $[a, b]$:

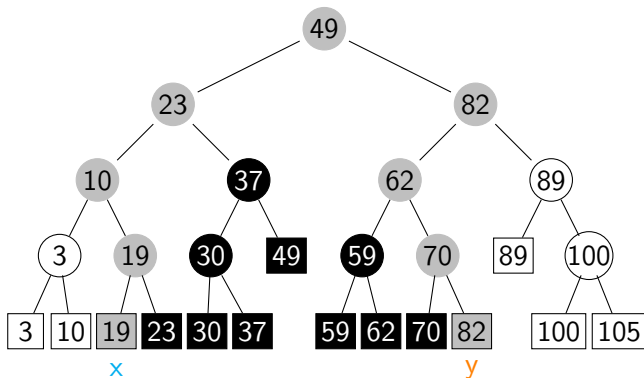
Busque a e b na ABB, terminando nas folhas x e y .

Exemplo: para $a = 15$ e $b = 80$, terminamos com $x = 19$ e $y = 82$.



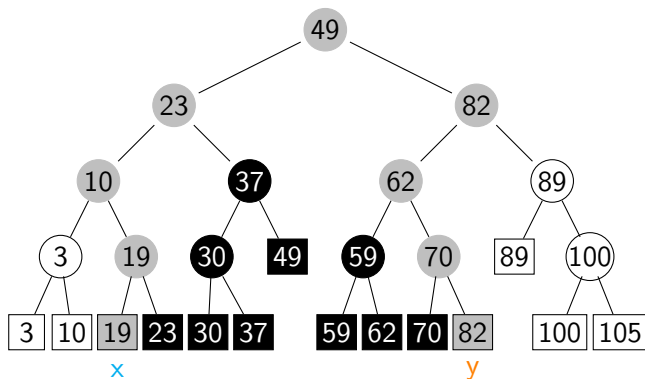
Resposta: folhas entre x e y , possivelmente incluindo x e y

Consumo de tempo



Ache o vértice s onde os caminhos para x e y bifurcam e, de s para x , imprima todas as folhas de subárvores à direita e, de s para y , imprima todas as folhas de subárvores à esquerda.

Consumo de tempo



Ache o vértice s onde os caminhos para x e y bifurcam e, de s para x , imprima todas as folhas de subárvores à direita e, de s para y , imprima todas as folhas de subárvores à esquerda.

$O(\lg n + k)$, onde k é o número de elementos de S em $[a, b]$.

KD-trees

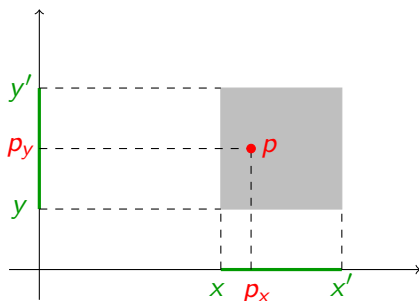
Espaço 2-dimensional, buscas ortogonais.

P : conjunto de pontos

Consulta retangular:

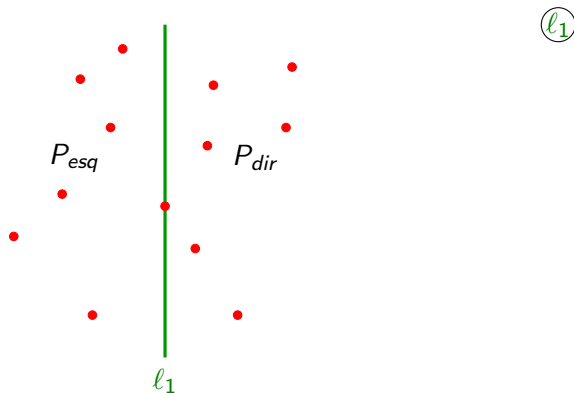
Quais pontos de P estão em $[x : x'] \times [y : y']$?

Ponto p está em $[x : x'] \times [y : y']$ se $p_x \in [x : x']$ e $p_y \in [y : y']$.



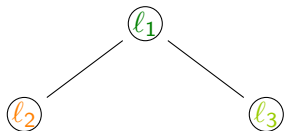
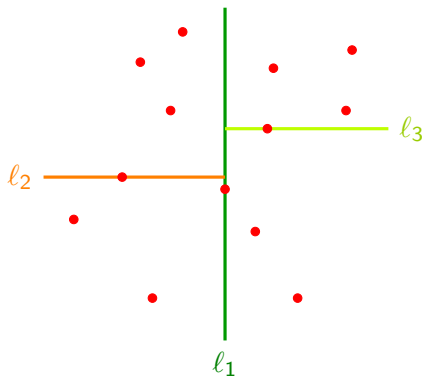
KD-trees

Árvore de busca binária que divide os pontos alternadamente, ora por uma linha vertical, ora por uma linha horizontal.



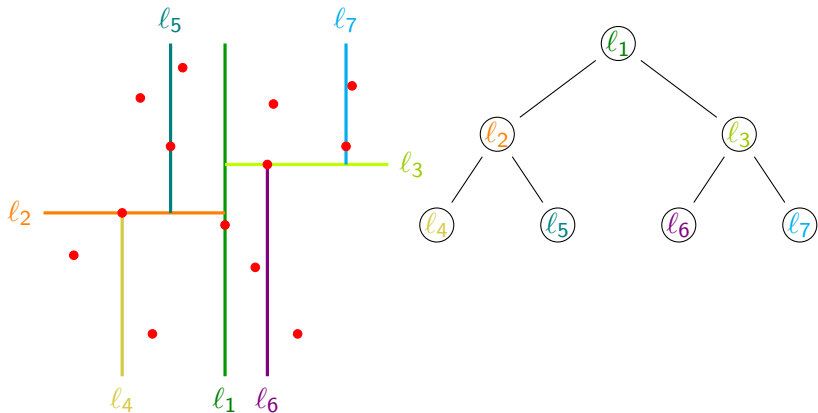
KD-trees

Árvore de busca binária que divide os pontos alternadamente, ora por uma linha vertical, ora por uma linha horizontal.



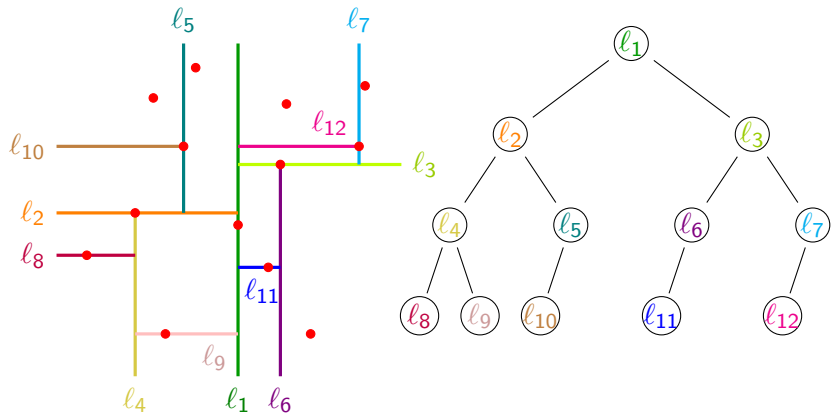
KD-trees

Árvore de busca binária que divide os pontos alternadamente, ora por uma linha vertical, ora por uma linha horizontal.



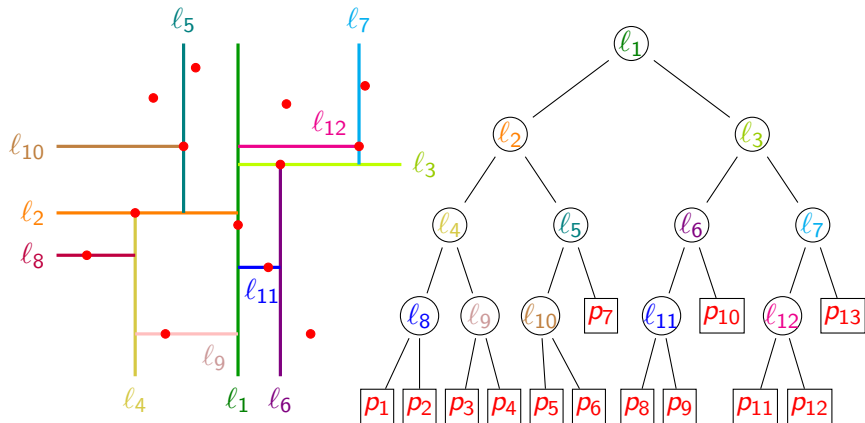
KD-trees

Árvore de busca binária que divide os pontos alternadamente, ora por uma linha vertical, ora por uma linha horizontal.



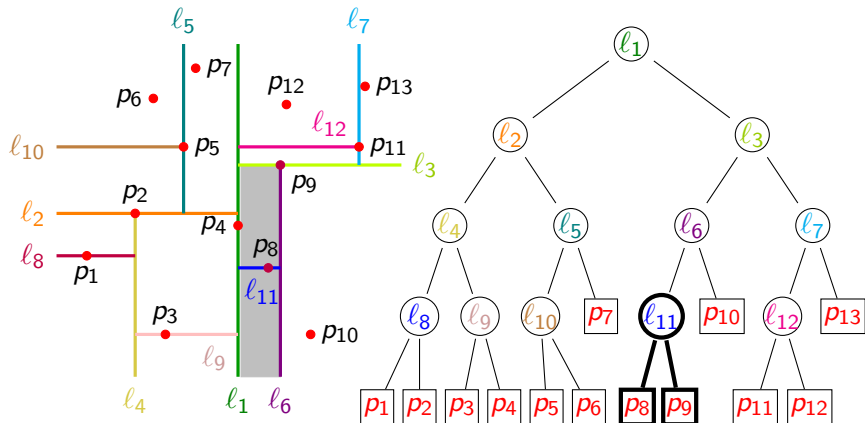
KD-trees

Árvore de busca binária que divide os pontos alternadamente, ora por uma linha vertical, ora por uma linha horizontal.



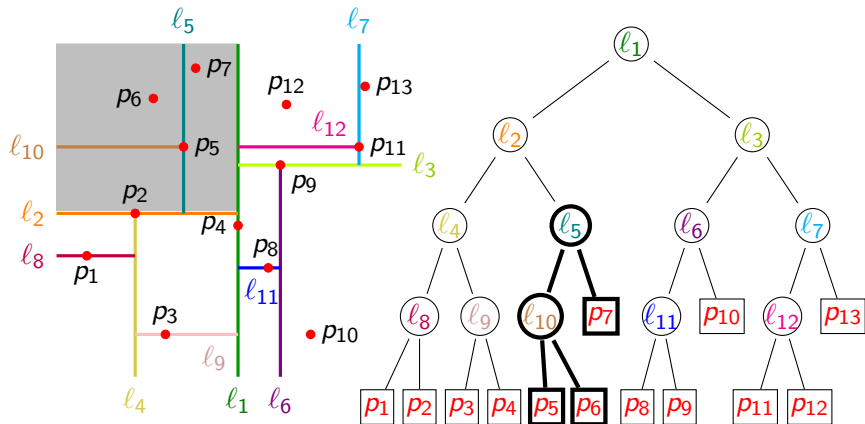
KD-trees

Subárvores e suas regiões



KD-trees

Subárvores e suas regiões



Construção de uma KD-tree

BuildKDTree(P , $depth$)

- 1 se $|P| = 1$
- 2 então cria e retorna uma folha com o único ponto em P
- 3 se $depth$ é par
- 4 então divide P em dois conjuntos por uma reta vertical ℓ
passando pela mediana das x -coordenadas dos pontos de P ;
seja P_1 o conjunto dos pontos de P à esquerda ou
em cima de ℓ e P_2 os pontos de P à direita de ℓ .
- 5 senão ▷ análogo para uma reta horizontal ℓ
- 6 $v_e \leftarrow \text{BuildKDTree}(P_1, depth + 1)$
- 7 $v_d \leftarrow \text{BuildKDTree}(P_2, depth + 1)$
- 8 cria nó v com conteúdo ℓ e filhos v_e e v_d
- 9 devolva v

Construção de uma KD-tree

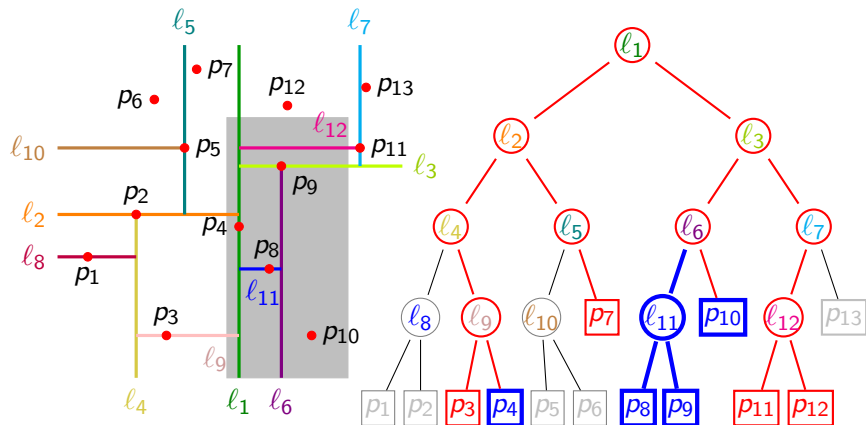
BuildKDTree(P , $depth$)

- 1 se $|P| = 1$
- 2 então cria e retorna uma folha com o único ponto em P
- 3 se $depth$ é par
- 4 então divida P em dois conjuntos por uma reta vertical ℓ
 passando pela mediana das x -coordenadas dos pontos de P ;
 seja P_1 o conjunto dos pontos de P à esquerda ou
 em cima de ℓ e P_2 os pontos de P à direita de ℓ .
- 5 senão ▷ análogo para uma reta horizontal ℓ
- 6 $v_e \leftarrow$ BuildKDTree(P_1 , $depth + 1$)
- 7 $v_d \leftarrow$ BuildKDTree(P_2 , $depth + 1$)
- 8 cria nó v com conteúdo ℓ e filhos v_e e v_d
- 9 devolva v

Consumo de tempo: $O(n \lg n)$, onde $n := |P|$.

Consumo de espaço: $O(n)$.

Consultas ortogonais em KD-trees



Consultas ortogonais em KD-trees

SearchKDTree(v, R)

- 1 se v é folha
- 2 então imprima o ponto guardado em v se ele pertence a R
- 3 senão se $region(esq(v)) \subseteq R$
- 4 então **ReportSubtree**($esq(v)$)
- 5 senão se $region(esq(v))$ intersecta R
- 6 então **SearchKDTree**($esq(v), R$)
- 7 se $region(dir(v)) \subseteq R$
- 8 então **ReportSubtree**($dir(v)$)
- 9 senão se $region(dir(v))$ intersecta R
- 10 então **SearchKDTree**($dir(v), R$)

Consultas ortogonais em KD-trees

SearchKDTree(v, R)

- 1 se v é folha
- 2 então imprima o ponto guardado em v se ele pertence a R
- 3 senão se $region(esq(v)) \subseteq R$
- 4 então **ReportSubtree**($esq(v)$)
- 5 senão se $region(esq(v))$ intersecta R
- 6 então **SearchKDTree**($esq(v), R$)
- 7 se $region(dir(v)) \subseteq R$
- 8 então **ReportSubtree**($dir(v)$)
- 9 senão se $region(dir(v))$ intersecta R
- 10 então **SearchKDTree**($dir(v), R$)

Pré-processe a árvore calculando $region(v)$ para todo v .

Consultas ortogonais em KD-trees

SearchKDTree(v, R)

- 1 se v é folha
- 2 então imprima o ponto guardado em v se ele pertence a R
- 3 senão se $region(esq(v)) \subseteq R$
- 4 então **ReportSubtree**($esq(v)$)
- 5 senão se $region(esq(v))$ intersecta R
- 6 então **SearchKDTree**($esq(v), R$)
- 7 se $region(dir(v)) \subseteq R$
- 8 então **ReportSubtree**($dir(v)$)
- 9 senão se $region(dir(v))$ intersecta R
- 10 então **SearchKDTree**($dir(v), R$)

Pré-processe a árvore calculando $region(v)$ para todo v .

Consumo de tempo: $O(\sqrt{n} + k)$, onde n é o número de pontos e k o número de pontos reportados na consulta.

Resumo

P : coleção de n pontos

Objetivo: responder consultas retangulares

Kd-trees:

Consumo de espaço: $O(n)$

Custo por consulta: $O(\sqrt{n} + k)$.

Resumo

P : coleção de n pontos

Objetivo: responder consultas retangulares

Kd-trees:

Consumo de espaço: $O(n)$

Custo por consulta: $O(\sqrt{n} + k)$.

Range trees:

Consumo de espaço: $O(n \lg n)$

Custo por consulta: $O(\lg^2 n + k)$.

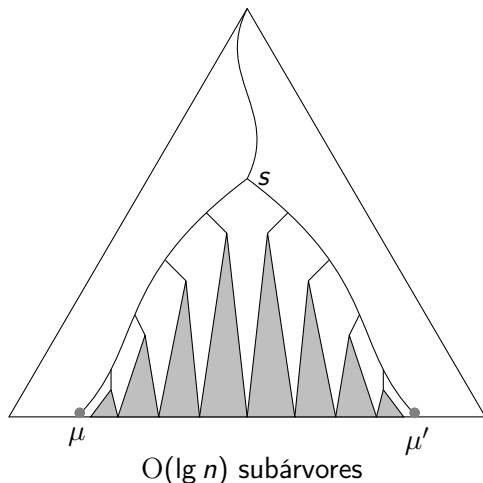
Range trees

Consulta retangular: quais pontos de P estão em $[x : x'] \times [y : y']$?

Range trees

Consulta retangular: quais pontos de P estão em $[x : x'] \times [y : y']$?

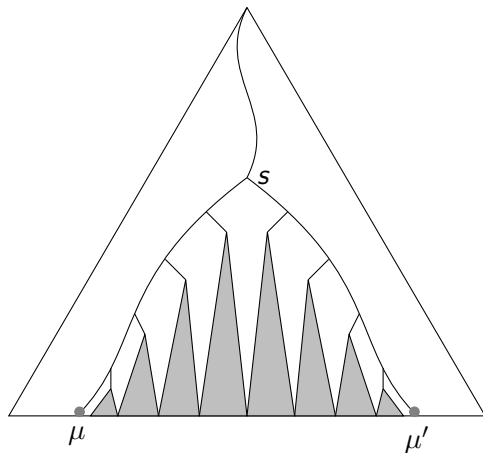
Ideia: primeiro buscar pontos cuja x -coordenada está em $[x : x']$.



Range trees

Consulta retangular: quais pontos de P estão em $[x : x'] \times [y : y']$?

Ideia: primeiro buscar pontos cuja x -coordenada está em $[x : x']$.



Buscar nestas subárvores pontos com y -coordenada em $[y : y']$.

Range trees: estrutura de dados com níveis

Como fazer estas segundas buscas eficientemente?

Range trees: estrutura de dados com níveis

Como fazer estas segundas buscas eficientemente?

Cada vértice v da ABB_x tem um conjunto canônico P_v ,
que consiste dos pontos que aparecem na subárvore enraizada em v .

Range trees: estrutura de dados com níveis

Como fazer estas segundas buscas eficientemente?

Cada vértice v da ABB_x tem um conjunto canônico P_v , que consiste dos pontos que aparecem na subárvore enraizada em v .

O conjunto dos pontos de P com x -coordenada em $[x : x']$ é a união disjunta de $O(\lg n)$ conjuntos canônicos.

Range trees: estrutura de dados com níveis

Como fazer estas segundas buscas eficientemente?

Cada vértice v da ABB_x tem um conjunto canônico P_v , que consiste dos pontos que aparecem na subárvore enraizada em v .

O conjunto dos pontos de P com x -coordenada em $[x : x']$ é a união disjunta de $O(\lg n)$ conjuntos canônicos.

Manteremos, para cada v , o conjunto P_v em uma ABB secundária, ordenada pela y -coordenada dos pontos.

Range trees: estrutura de dados com níveis

Como fazer estas segundas buscas eficientemente?

Cada vértice v da ABB_x tem um conjunto canônico P_v , que consiste dos pontos que aparecem na subárvore enraizada em v .

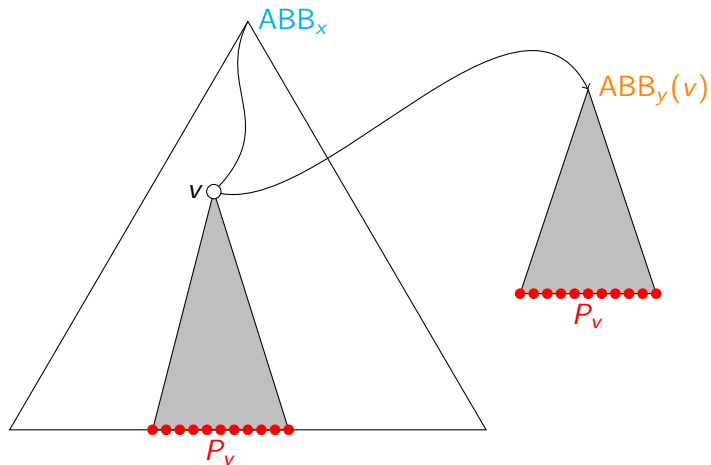
O conjunto dos pontos de P com x -coordenada em $[x : x']$ é a união disjunta de $O(\lg n)$ conjuntos canônicos.

Manteremos, para cada v , o conjunto P_v em uma ABB secundária, ordenada pela y -coordenada dos pontos.

Chamemos essa ABB de $ABB_y(v)$ para cada v .

$ABB_y(v)$: árvore para busca unidimensional por intervalo.

Range trees: estrutura de dados com níveis



Construção de uma range tree 2D

BuildRangeTree(P)

- 1 Construa uma ABB_y T para P em relação a y
- 2 se $|P| = 1$
- 3 então cria uma folha v para este ponto
- 4 senão calcule x_{meio} para P
- 5 divida P em P_{esq} e P_{dir} em relação a x_{meio}
- 6 $v_{esq} \leftarrow \text{BuildRangeTree}(P_{esq})$
- 7 $v_{dir} \leftarrow \text{BuildRangeTree}(P_{dir})$
- 8 cria nó v com chave x_{meio} e filhos v_{esq} e v_{dir}
- 9 $ABB_y(v) \leftarrow T$
- 10 devolva v

Construção de uma range tree 2D

BuildRangeTree(P)

- 1 Construa uma ABB_y T para P em relação a y
- 2 se $|P| = 1$
- 3 então cria uma folha v para este ponto
- 4 senão calcule x_{meio} para P
- 5 divida P em P_{esq} e P_{dir} em relação a x_{meio}
- 6 $v_{esq} \leftarrow \text{BuildRangeTree}(P_{esq})$
- 7 $v_{dir} \leftarrow \text{BuildRangeTree}(P_{dir})$
- 8 cria nó v com chave x_{meio} e filhos v_{esq} e v_{dir}
- 9 $ABB_y(v) \leftarrow T$
- 10 devolva v

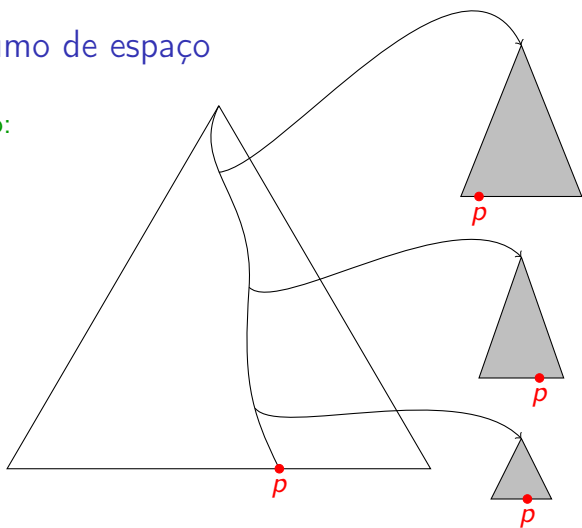
Consumo de tempo:

Se pré-ordenarmos por y e garantirmos que as chamadas a **BuildRangeTree** sempre recebem os pontos ordenados por y , o consumo de tempo será $O(n \lg n)$, onde $n := |P|$, pois a construção de cada ABB_y custará tempo linear neste caso.

Range trees: consumo de espaço

Consumo de espaço:

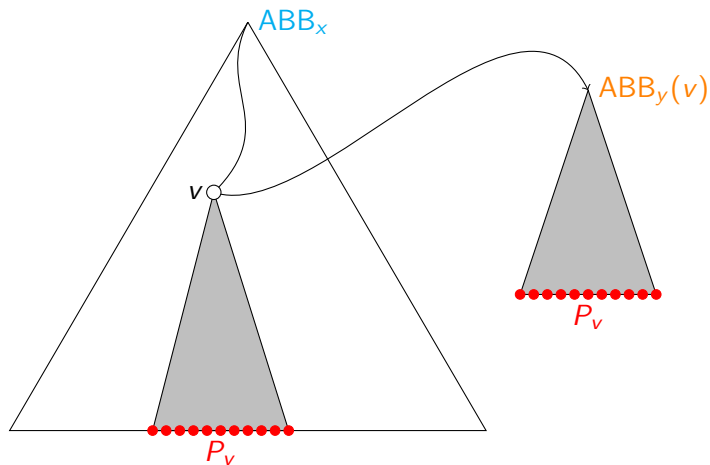
Pela ABB_x : $O(n)$



Cada ponto aparece em $ABB_y(v)$ para $O(\lg n)$ vértices v da ABB_x .
No total então $O(n \lg n)$ pelas $ABB_y(v)$ para todo v da ABB_x .

Consumo total de espaço: $O(n \lg n)$

Range trees: consulta



Consulta: busca na ABB_x e, para cada $ABB_y(v)$ encontrada na consulta por x , busca na $ABB_y(v)$ por y .

Range trees: consulta

Query2DRangeTree($T, [x, x'] \times [y, y']$)

- 1 $s \leftarrow$ **FindSplitNode**(T, x, x')
- 2 se s é folha
- 3 então imprime o ponto em s se está em $[x, x'] \times [y, y']$
- 4 senão \triangleright siga o caminho até x e processe subárvores à direita
- 5 $v \leftarrow$ *esq*(s)
- 6 enquanto v não é folha faça
- 7 se $x \leq x_v$
- 8 então **Query1DRangeTree**($T^y(v), [y, y']$)
- 9 $v \leftarrow$ *esq*(v)
- 10 senão $v \leftarrow$ *dir*(v)
- 11 \triangleright siga o caminho até x' e processe subárvores à esquerda
- 12 ...

Range trees: consulta

$\text{Query2DRangeTree}(T, [x, x'] \times [y, y'])$

- 1 $s \leftarrow \text{FindSplitNode}(T, x, x')$
- 2 se s é folha
- 3 então imprime o ponto em s se está em $[x, x'] \times [y, y']$
- 4 senão \triangleright siga o caminho até x e processe subárvores à direita
- 5 $v \leftarrow \text{esq}(s)$
- 6 enquanto v não é folha faça
- 7 se $x \leq x_v$
- 8 então $\text{Query1DRangeTree}(T^y(v), [y, y'])$
- 9 $v \leftarrow \text{esq}(v)$
- 10 senão $v \leftarrow \text{dir}(v)$
- 11 \triangleright siga o caminho até x' e processe subárvores à esquerda
- 12 ...

Consumo de tempo por consulta: $O(\lg^2 n)$

$O(\lg n)$ pela busca unidimensional em T por $[x : x']$ e

$O(\lg n)$ buscas unidimensionais por $[y : y']$, custo $O(\lg n)$ cada.

Range trees: análise

Consumo de tempo por consulta:

$O(\lg n)$ pela busca unidimensional na ABB_x por $[x : x']$ e

$O(\lg n)$ buscas unidimensionais por $[y : y']$,
uma em cada $ABB_y(v)$ encontrada na busca acima.

Tempo total por consulta: $O(\lg^2 n)$

Range trees: análise

Consumo de tempo por consulta:

$O(\lg n)$ pela busca unidimensional na ABB_x por $[x : x']$ e

$O(\lg n)$ buscas unidimensionais por $[y : y']$,
uma em cada $ABB_y(v)$ encontrada na busca acima.

Tempo total por consulta: $O(\lg^2 n)$

Consumo de espaço:

Pela ABB_x : $O(n)$

Cada ponto aparece em $ABB_y(v)$ para $O(\lg n)$ vértices v da ABB_x .
No total então $O(n \lg n)$ pelas $ABB_y(v)$ para todo v da ABB_x .

Consumo total de espaço: $O(n \lg n)$

Hipótese simplificadora

Como garantir que todas as x -coordenadas são distintas e todas as y -coordenadas são distintas?

Hipótese simplificadora

Como garantir que todas as x -coordenadas são distintas e todas as y -coordenadas são distintas?

Seja $p = (x, y)$.

Considere que p é na verdade $((x, y), (y, x))$.

Defina $(a, b) < (c, d)$ sse $a < b$ ou $a = b$ e $c < d$.

Hipótese simplificadora

Como garantir que todas as x -coordenadas são distintas e todas as y -coordenadas são distintas?

Seja $p = (x, y)$.

Considere que p é na verdade $((x, y), (y, x))$.

Defina $(a, b) < (c, d)$ sse $a < c$ ou $a = c$ e $b < d$.

Ou seja, dados dois pontos $p = (x, y)$ e $q = (w, z)$,

$$p_x < q_x \text{ sse } x < w \text{ ou } x = w \text{ e } y < z$$

$$p_y < q_y \text{ sse } y < z \text{ ou } y = z \text{ e } x < w$$