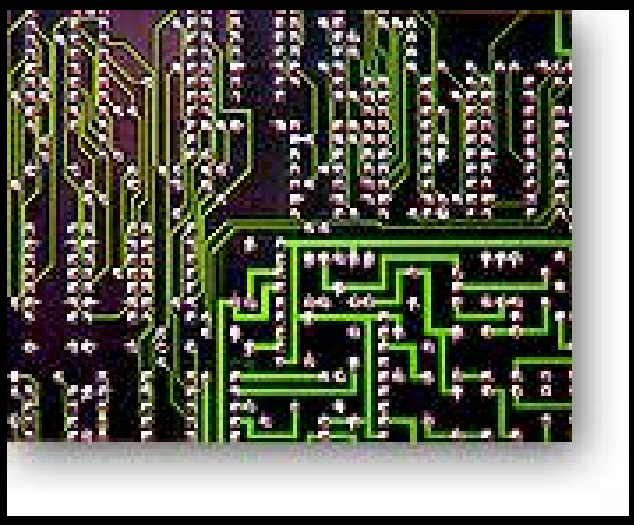
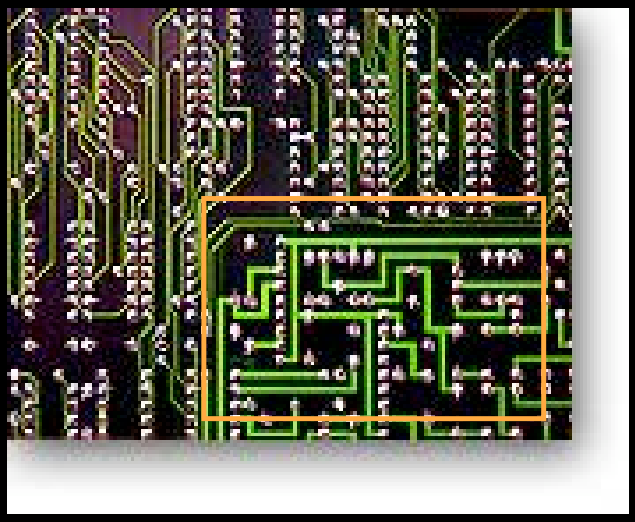
Interval Trees, Priority Search Trees, Segment Trees

João Comba

Pesquisas intervalares sobre segmentos de reta



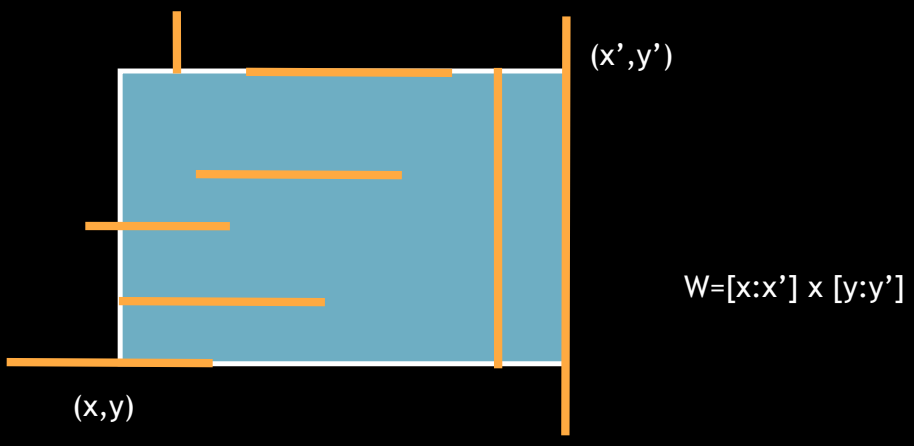
Pesquisas intervalares sobre segmentos de reta



Consultar uma coleção de segmentos de reta e reportar aqueles dentro de uma janela de busca

Interval Trees

- Restringir a segmentos de reta ortogonais aos eixos cartesianos



Interval Trees

Caso 1: Se um dos pontos extremos pertence ao interior da janela

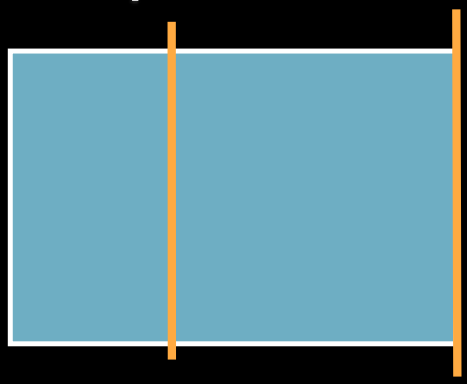
Interval Trees

Caso 1: Se um dos pontos extremos pertence ao interior da janela

Reportar em O(log n + k) com uma estrutura que usa O(nlogn) de memória e tempo de processamento (Range Tree + Fractional Cascading)

Interval Trees

Caso 2: Encontrar segmentos que não possuem pontos extremos dentro da janela



Propriedade : Linhas ou (a) cruzam duas vezes a janela ou (b) contém uma aresta da janela

Teste suficiente : reportar todos segmentos que intersectam a aresta mais a esquerda e mais abaixo da janela

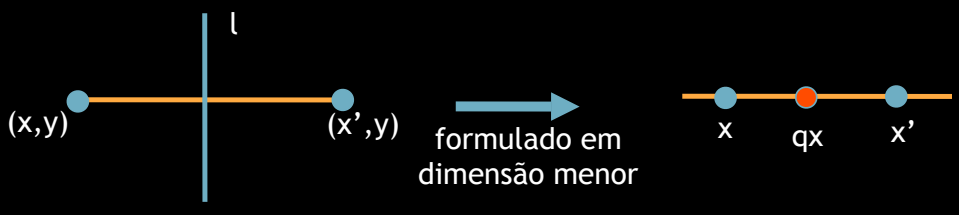
Interval Trees

Caso 2: Encontrar segmentos que não possuem pontos extremos dentro da janela

OBS 1: Lidar com interseções com a aresta mais a esquerda (para a mais abaixo é feito de forma similar trocando os papéis de X e Y)

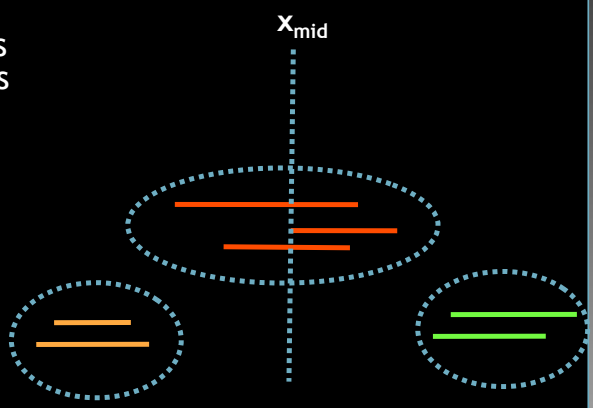
PROBLEMA: pré-processar um conjunto de segmentos de linha horizontais no plano de forma a reportar interseções com um segmento de linha vertical

OBS 2: Simplificar o problema para lidar com uma linha vertical ao invés de um segmento de reta



Interval Tree

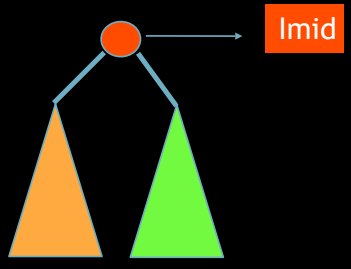
- Sejam $\{[x_1:x_1'],[x_2:x_2'],\ldots,[x_n:x_n']\}$ intervalos fechados correspondentes aos pontos extremos de n segmentos de reta horizontais
- Seja x_{mid} a mediana dos 2n pontos extremos
- Classificar os intervalos em 3 grupos:
  - Completamente a esquerda de x_{mid}
  - Completamente a direita de x_{mid}
  - Cruzando x_{mid}
- **Construir árvore binária de intervalos**



Interval Tree

Problema: todos intervalos podem conter x_{mid}

- evitar armazenar múltiplos intervalos
- usar estrutura associada a cada nodo contendo os intervalos cruzados pelo particionador

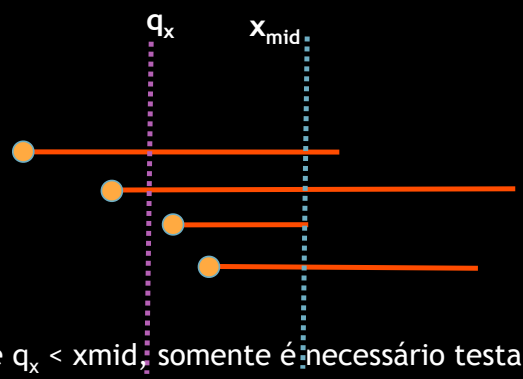


Interval Tree

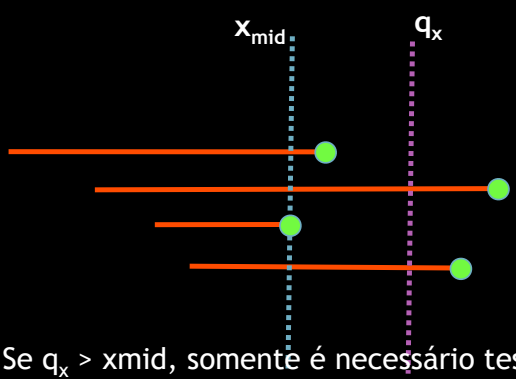
Problema: Mas se todos os intervalos são armazenados em um mesmo nodo, não voltamos ao problema original ?

Interval Tree

Problema: Mas se todos os intervalos são armazenados em um mesmo nodo, não voltamos ao problema original ?



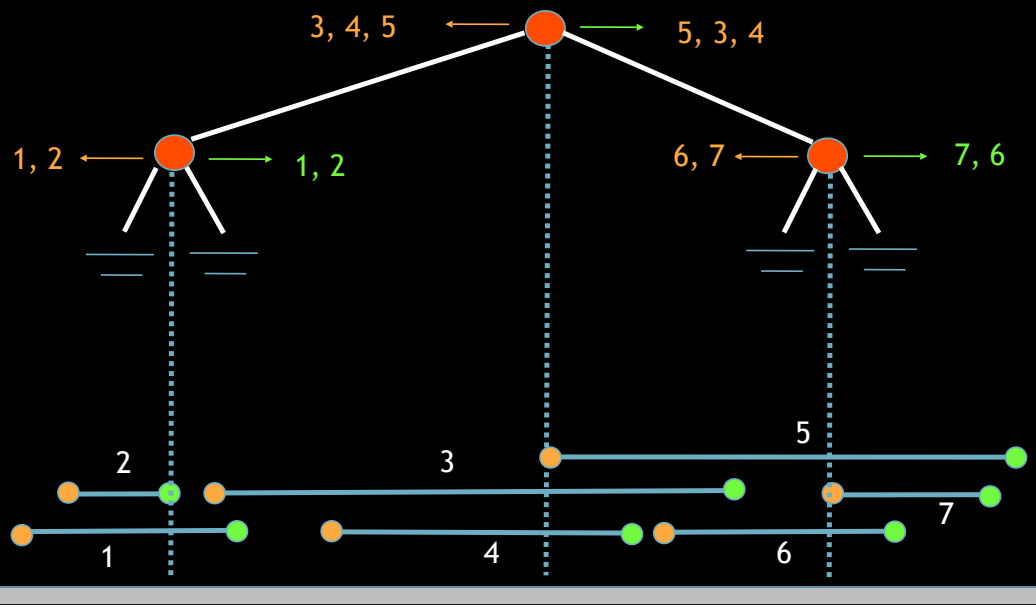
Se q_x < xmid, somente é necessário testar os pontos extremos esquerdos para verificar se qx cruza o intervalo



Se q_x > xmid, somente é necessário testar os pontos extremos direitos para verificar se qx cruza o intervalo

Interval Tree

Armazenar os intervalos associados ao nodo em 2 listas ordenadas (para os pontos extremos esquerdos e direitos)



Interval Trees

Algoritmo BuscaIntervalTree(v, qx)

Entrada: Raiz v da interval tree, ponto de busca qx

Saída: Todos intervalos contendo qx

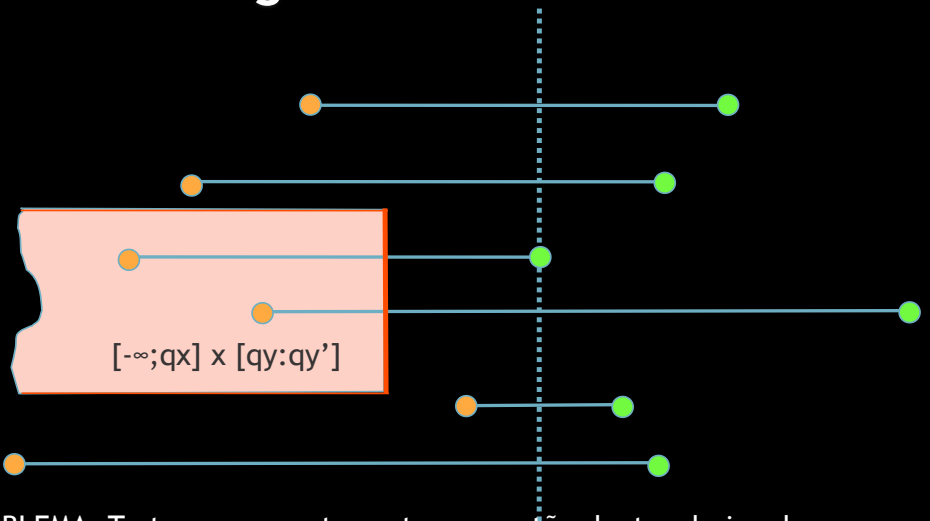
1. IF v não é folha
2. THEN IF qx < xmid(v)
3. THEN varre lista leftpoint(v), começando com o ponto extremo mais a esquerda, reportando todos intervalos contendo qx e parando no primeiro intervalo que não contiver qx.
4. BuscaIntervalTree(leftchild(v), qx)
5. ELSE varre lista rightpoint(v), começando com o ponto extremo mais a direita, reportando todos intervalos contendo qx e parando no primeiro intervalo que não contiver qx.
6. BuscaIntervalTree(rightchild(v), qx)

Interval Trees

Teorema: Uma interval tree para um conjunto I de n intervalos usa O(n) de memória, reporta todas interseções com um ponto em O(logn + k)

Removendo simplificação

Reportar interseções com segmento de reta vertical no lugar de linha vertical



PROBLEMA: Testar se os pontos extremos estão dentro da janela

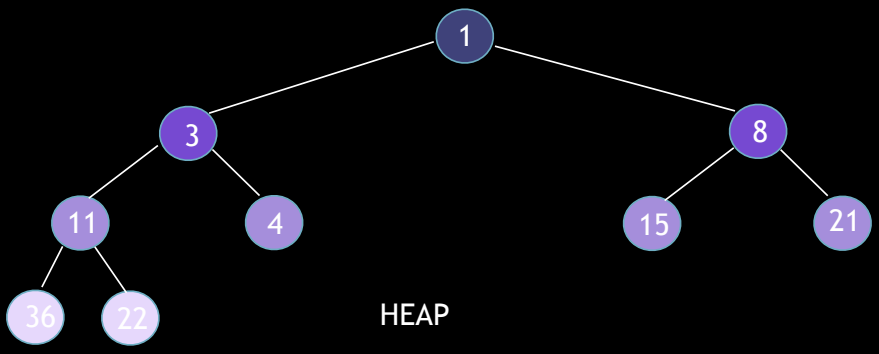
® Usar uma 2D Range Tree com fractional cascading no lugar das listas ordenadas

Priority Search Trees

Estrutura adequada para responder buscas [x,y] Є [-∞;qx] x [qy:qy']

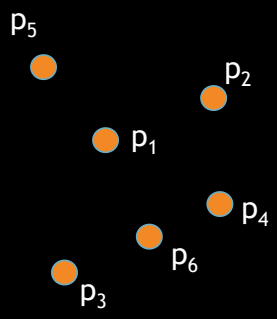
Baseado em Heaps:

- árvores balanceadas
- menor (ou maior armazenado no topo)
- sub-árvores possuem nós com todos valores menores (ou maiores) que o nodo

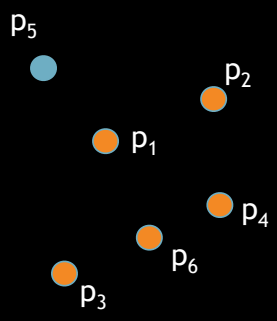


HEAP

Priority Search Trees

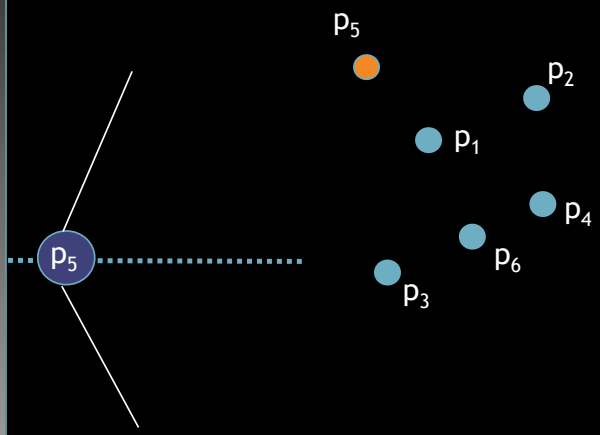


Priority Search Trees



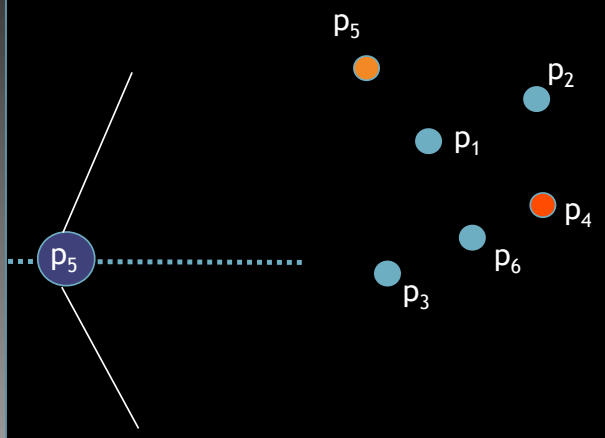
1. Encontrar ponto de menor coordenada x

Priority Search Trees



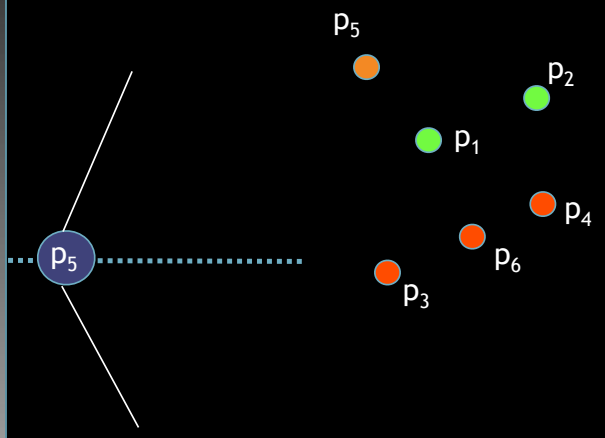
1. Encontrar ponto de menor coordenada x
2. Analisar os pontos restantes

Priority Search Trees



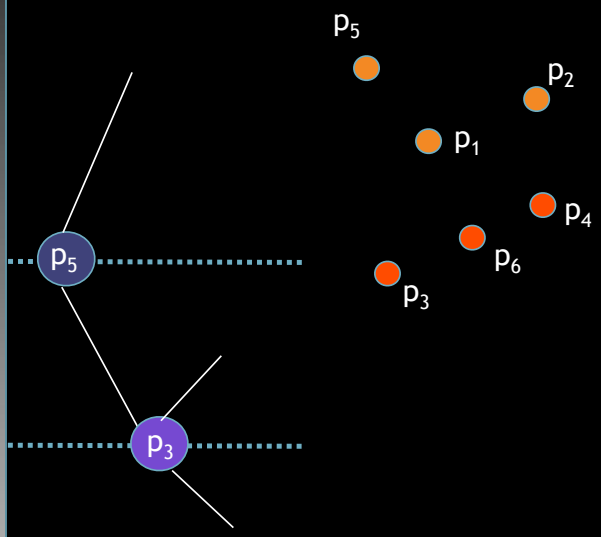
1. Encontrar ponto de menor coordenada x
3. Analisar os pontos restantes
4. Encontrar a mediana em y dos pontos restantes
5. Criar nodo com o valor y deste nodo

Priority Search Trees



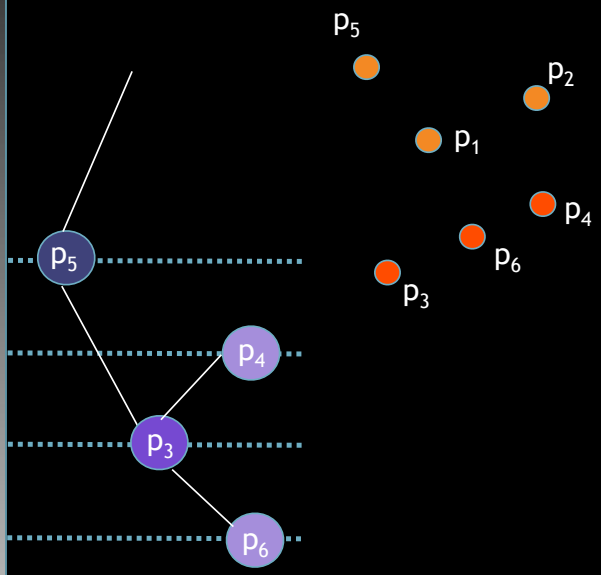
1. Encontrar ponto de menor coordenada x
3. Analisar os pontos restantes
4. Encontrar a mediana em y dos pontos restantes
5. Usar a mediana para criar dois conjuntos: acima e abaixo da mediana

Priority Search Trees



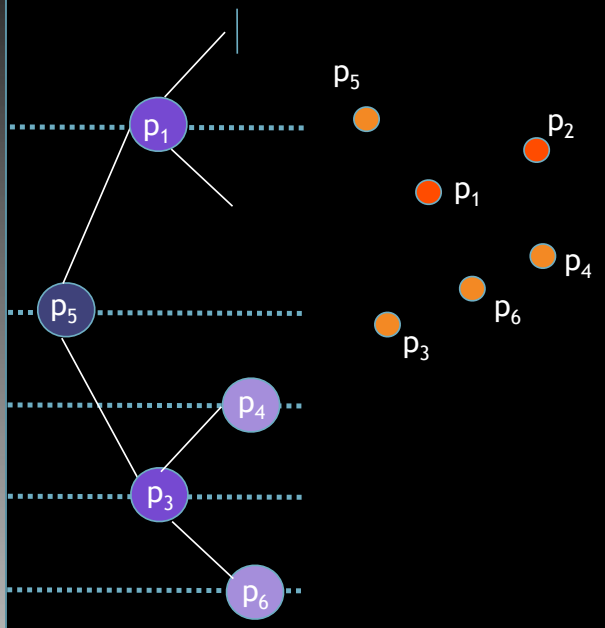
1. Encontrar ponto de menor coordenada x
3. Analisar os pontos restantes
4. Encontrar a mediana em y dos pontos restantes
5. Usar a mediana para criar dois conjuntos: acima e abaixo da mediana
6. Repetir o processo na subárvore esquerda com os pontos abaixo da mediana

Priority Search Trees



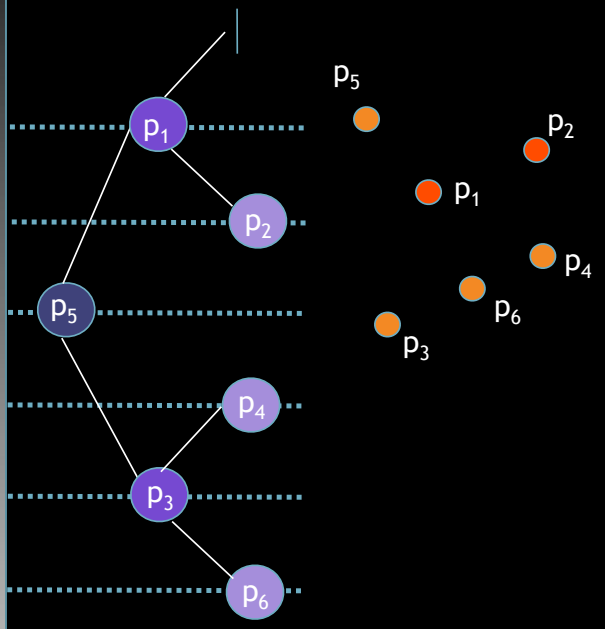
1. Encontrar ponto de menor coordenada x
3. Analisar os pontos restantes
4. Encontrar a mediana em y dos pontos restantes
5. Usar a mediana para criar dois conjuntos: acima e abaixo da mediana
6. Repetir o processo na subárvore esquerda com os pontos abaixo da mediana

Priority Search Trees



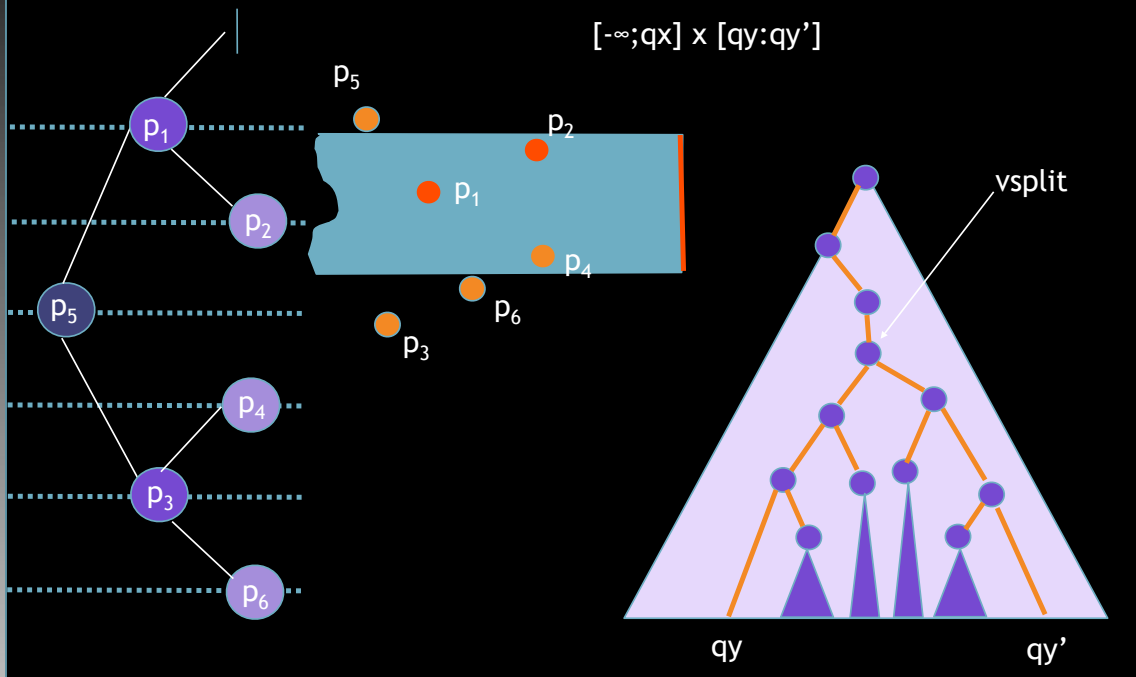
1. Encontrar ponto de menor coordenada x
3. Analisar os pontos restantes
4. Encontrar a mediana em y dos pontos restantes
5. Usar a mediana para criar dois conjuntos: acima e abaixo da mediana
6. Repetir o processo na subárvore esquerda com os pontos abaixo da mediana
7. Repetir o processo na subárvore direita com os pontos acima da mediana

Priority Search Trees

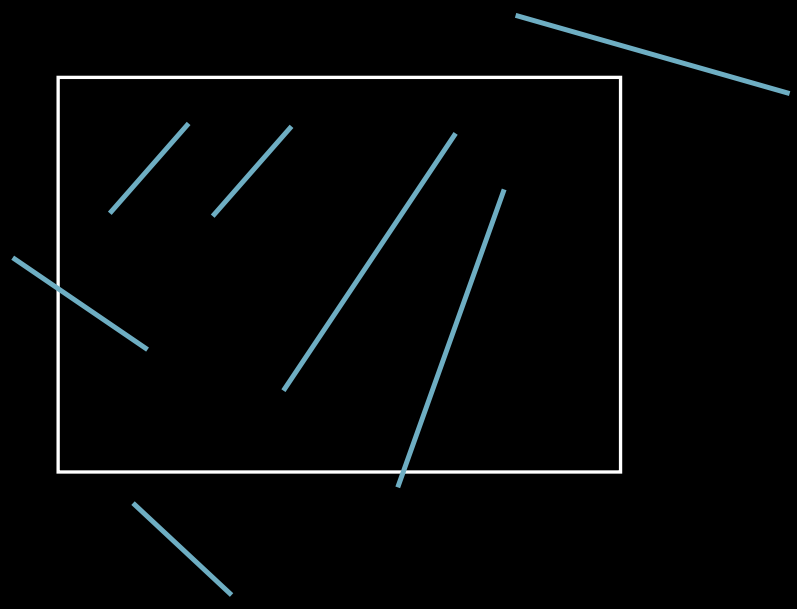


1. Encontrar ponto de menor coordenada x
3. Analisar os pontos restantes
4. Encontrar a mediana em y dos pontos restantes
5. Usar a mediana para criar dois conjuntos: acima e abaixo da mediana
6. Repetir o processo na subárvore esquerda com os pontos abaixo da mediana
7. Repetir o processo na subárvore direita com os pontos acima da mediana

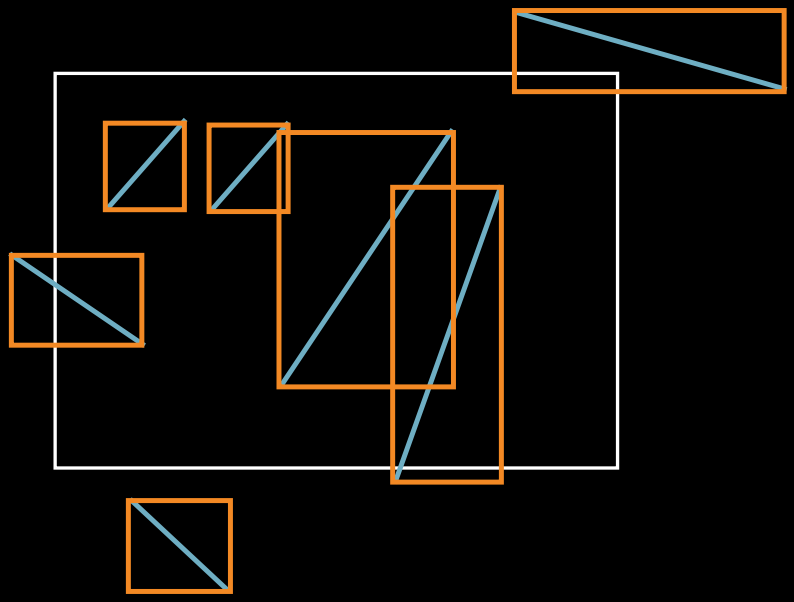
Priority Search Trees: Busca



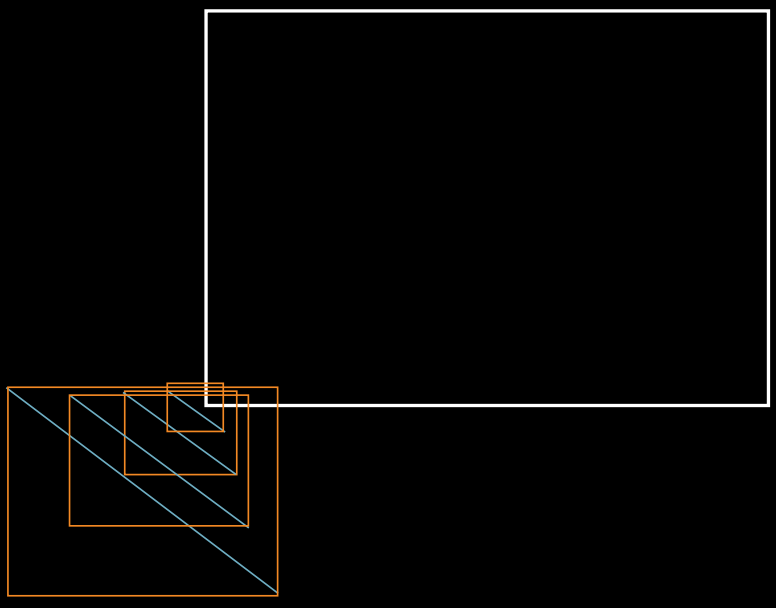
Segment Trees



Segment Trees



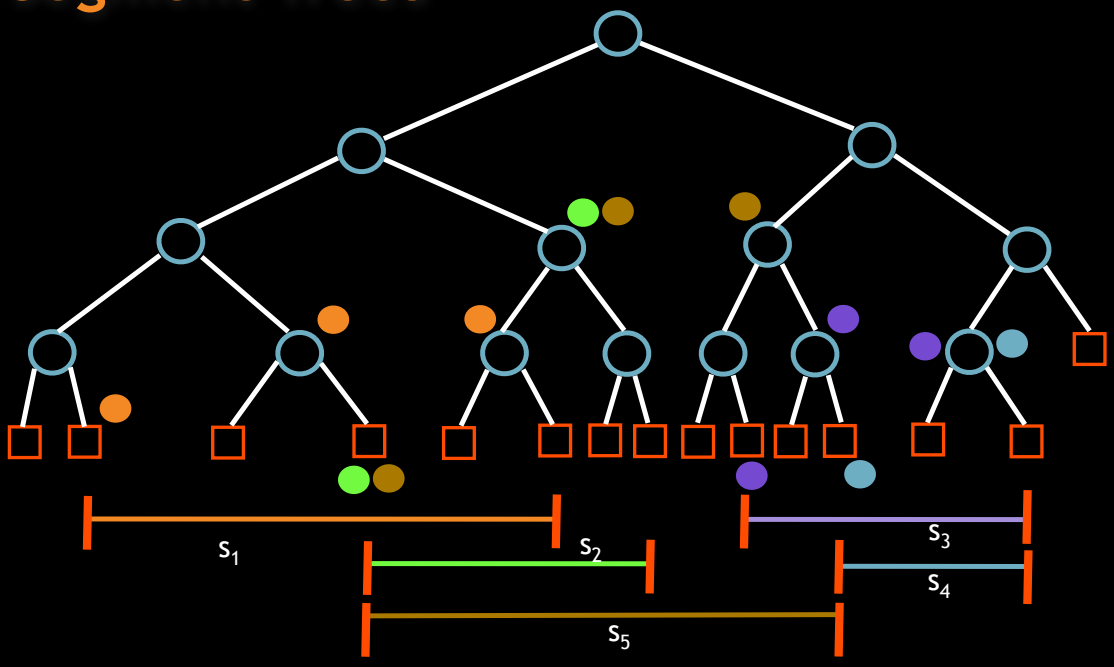
Segment Trees



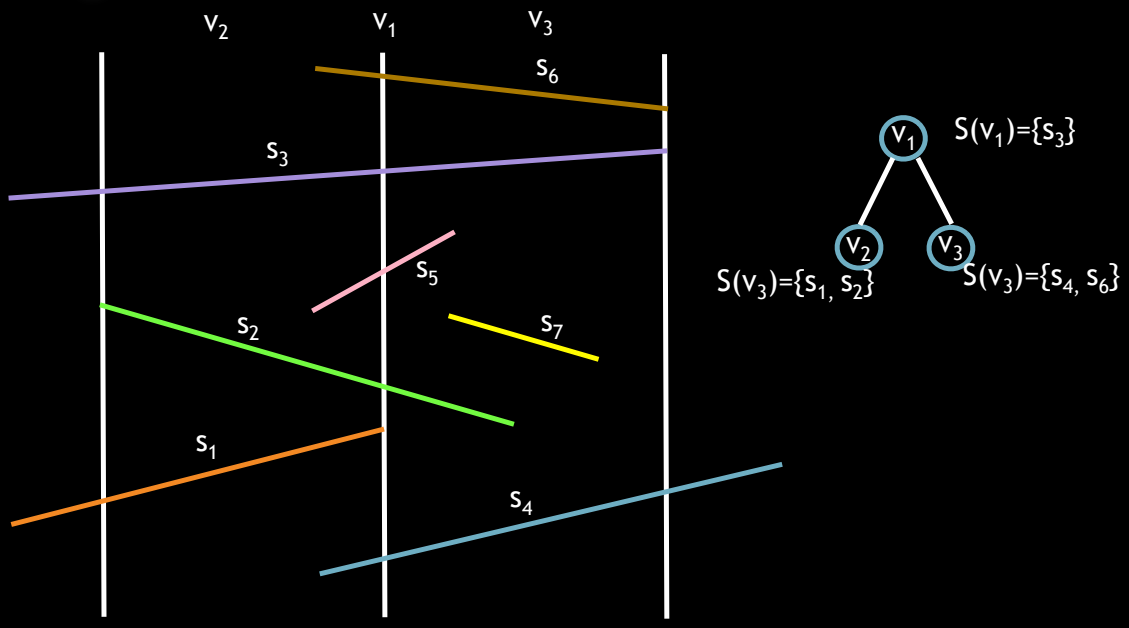
Segment Trees

$(-\infty:p_1),[p_1:p_1],(p_1:p_2),[p_2:p_2],\ \ldots\ ,(p_{m-1}:p_m),[p_m:p_m],(p_m:+\infty)$

Segment Trees



Segment Trees



Segment Trees

- Conjunto S de n segmentos (sem intersecões) é armazenado numa segment-tree nos intervalos em x dos segmentos
- O subconjunto canônico de cada nodo que contém os segmentos cortando a faixa é armazenado numa árvore binária balanceada

Segment Trees

Teorema: Seja S um conjunto de n segmentos de reta no plano. Segmentos interceptando uma janela retangular paralela aos eixos pode ser reportado em $O(\log^2 n + k)$ com uma estrutura de dados que usa $O(n \log n)$ de memória, construída em $O(n \log n)$.