BCC203 - Estrutura de Dados II

Seminário: Árvore R (R-Tree)

Caio Lucas, Felipe Braz, Lucas Chagas, Matheus Peixoto, Nicolas Mendes, Pedro Henrique Oliveira, Pedro Morais, Renato Franco e Vinícius Nunes

Universidade Federal de Ouro Preto, UFOP
Departamento de Computação, DECOM
Professor : **Guilherme Tavares de Assis**

Conteúdo

Apresentação

Origem

A Árvore R foi proposta pelo Antonin Guttman, doutor em Ciência da Computação pela Universidade da Califórnia (Estados Unidos da América), em 1984, com o objetivo de criar uma estrutura de dados que otimizasse consultas de intervalo em sistemas de gerenciamento de banco de dados multidimensionais.

Figura: Antonin Guttman

R-tree

As árvores R correspondem à estrutura de dados que são utilizadas para indexação de dados multidimensionais, como dados espaciais, por exemplo.

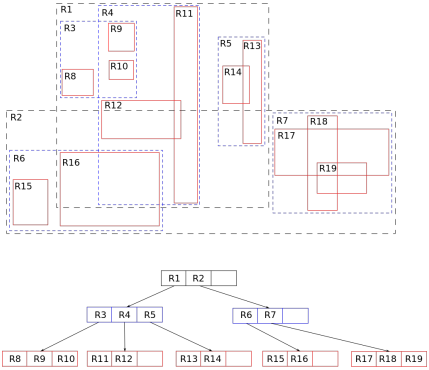


Figura: Árvore R retangular 2D

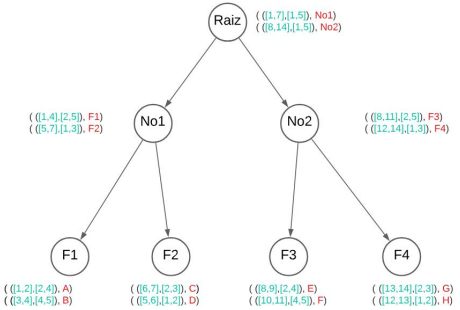


Figura: Árvore R com intervalos

O que são Dados multidimensionais?

Dados multidimensionais correspondem a um conjunto de informações organizadas em mais de uma variável ou dimensão. Cada dimensão diz respeito a um atributo do dado, podendo, por exemplo, ser algo de natureza escalar, como uma coordenada em um espaço, ou categórica, permitindo uma representação mais complexa da informação.

Exemplos de dados multidimensionais (dados espaciais)

- **Ponto**: coordenadas de uma unidade mínima em um espaço.
- **Linha**: sequência de pontos em uma mesma reta.
- **Linha poligonal**: sequência de pontos que não se encontram em uma mesma reta.
- **Poliedro**: sólido composto por faces.
- **Polígono**: sequência composta por linhas poligonais.

Estrutura geral

- Árvore R de ordem (m, M)
- Número máximo de entradas por nó: M
- Número mínimo de entradas por nó: m <= |M/2|
- Altura máxima da árvore: Hmax = $[\log_m N]$ - 1, onde N é o número de objetos inseridos

Estrutura em C

Representação da estrutura em C

```c
#define m 2
#define M 4

//Região
typedef struct{
    double limSupX;
    double limInfX;
    double limSupY;
    double limInfY;
}Regiao;

//Ponto
typedef struct{
    double x;
    double y;
}Ponto;

//MBR
typedef Regiao MBR[M];
```

Figura: Código em C - Parte 1

Representação da estrutura em C

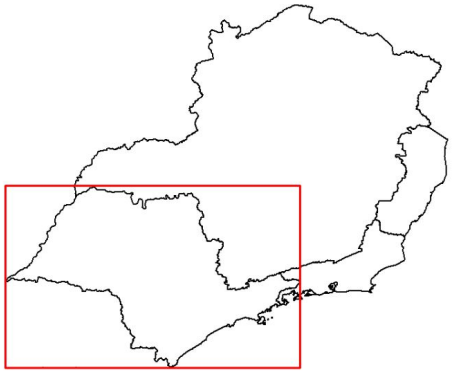
```c
//Ponteiro de MBR
typedef Regiao* PonteiroRegiao;

//ID
typedef int id;

//Nó Folha (contém dados)
typedef struct{
    Ponto pontos[M];
    id ids[M];
}Folha;

//Nós internos (índices)
typedef struct{
    id ids[M];
    PonteiroRegiao ponteiros[M+1];
    MBR mbr;
}Indice;
```

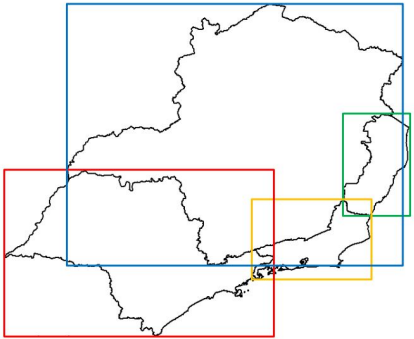
Figura: Código em C - Parte 2

Propriedades

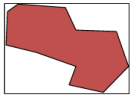
MBR (Minimum bounding rectangle)

MBR é um protocolo usado pela Árvore R, que visa construir os menores retângulos possíveis em torno dos dados da árvore.

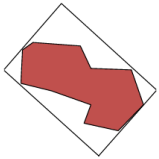




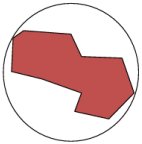
Outros protocolos para representação dos dados multidimensionais



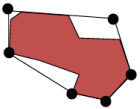
retângulo envolvente mínimo



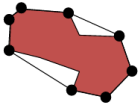
retângulo envolvente mínimo rotacionado



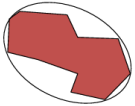
círculo envolvente mínimo



polígono envolvente mínimo com 6 vértices



casco convexo



elipse envolvente mínima

Figura: Outras formas conservativas

Propriedades e características

- Possui uma única raiz, nodos internos e nodos folha.
- A raiz aponta para a maior região no domínio espacial.
- Os nodos filhos têm suas regiões completamente sobrepostas às regiões de seus nodos pais.
- Os nodos folha contêm dados referentes ao MBR para os objetos atuais.
- Todos os nodos folha estão no mesmo nível, semelhante à árvore B.
- O número mínimo de entradas permitindo na raiz é 2, a menos que a raiz seja um nodo folha, nesse caso, ele pode conter uma ou nenhuma entrada, semelhante à árvore B*.

Propriedades e características

- Construção Bottom-Up: Todos os dados são inseridos nos nodos folha.
- Dinâmica: Permite inserção, remoção e edição dos dados.
- Trabalha com memória secundária: Nodos são páginas de disco com tamanho pré-determinado.

Aplicações

Aplicações da R-tree

- Sistemas de Informações Geográficas (GIS)
- Gerenciamento de Dados de Localização
- Banco de Dados Espaciais
- Detecção de Colisão e Simulações Físicas
- Planejamento de Rotas e Otimização Logística

Pesquisa

Pesquisa na R-Tree

Tipos de pesquisas

Figura: Tipos de pesquisas

Algoritmo de pesquisa

Pesquisando dado específico ou dados em um espaço arbitrário

- O processo inicia-se verificando se alguma área, presente na raiz, sobrepõe o espaço a ser pesquisado.
- Caso haja sobreposição, as áreas presentes nos nodos filhos também são verificadas e, caso haja sobreposição novamente, o processo se dá recursivamente.
- O processo recursivo encerra quando chega-se a um nodo folha. A partir daí, verifica-se todas as entradas que interceptam o espaço desejado e as retorna.

Complexidade de tempo

- Caso médio = $O(\log_m N)$
- Pior caso = $O(n)$

Exemplo 1 de pesquisa

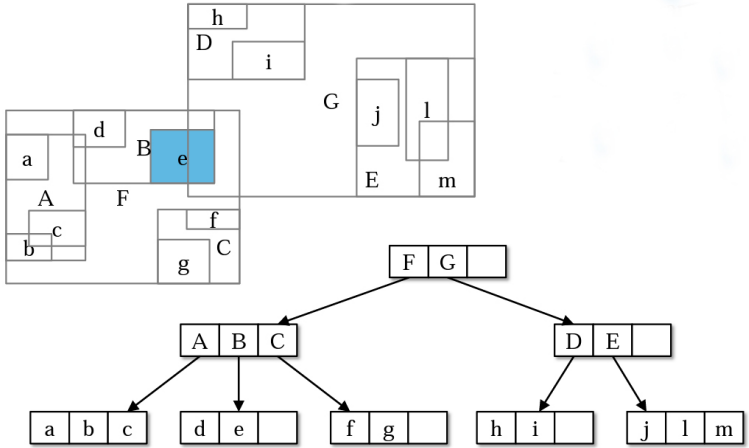


Figura: Pesquisar o subespaço E dentro da R-tree

Exemplo 2 de pesquisa

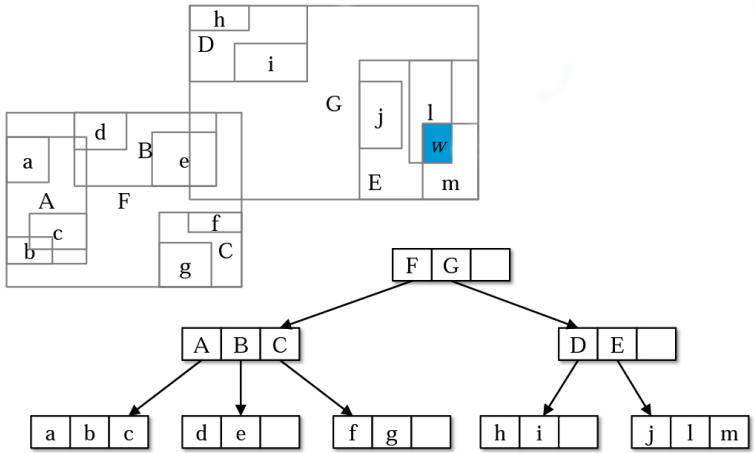


Figura: Pesquisar dados dentro de um subespaço w arbitrário

Inserção

Inserção na R-tree

- A árvore é percorrida, desde o nodo raiz até o nodo folha mais apropriado.
- Essa escolha é feita a partir do MBR que ocasionará no menor aumento possível da área, que ocorre caso a região do dado esteja fora dos limites de uma área.
- Em caso de empates, os métodos de desempate na ordem de prioridade são:
  - Escolher o de menor área
  - Escolher aquele com o menor número de entradas
  - Escolher arbitrariamente, qualquer um.
- Caso o nodo folha contenha espaço suficiente, a nova entrada é inserida no mesmo e o processo de inserção é encerrado.

Exemplo 1 de inserção parte 1

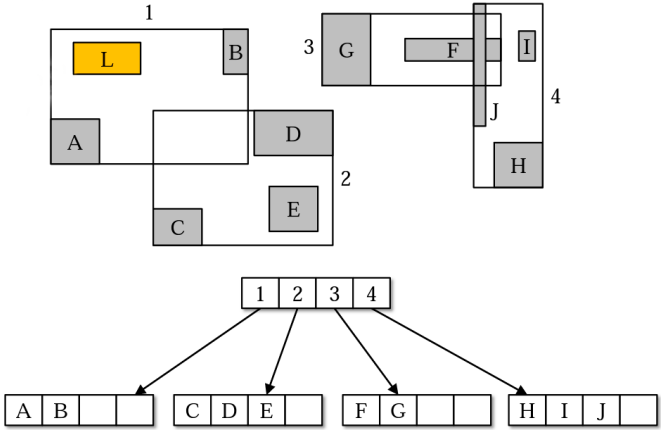


Figura: Inserir dado L em uma região já constituída (sem necessidade de criar ou dividir em novos espaços)

Exemplo 1 de inserção parte 2

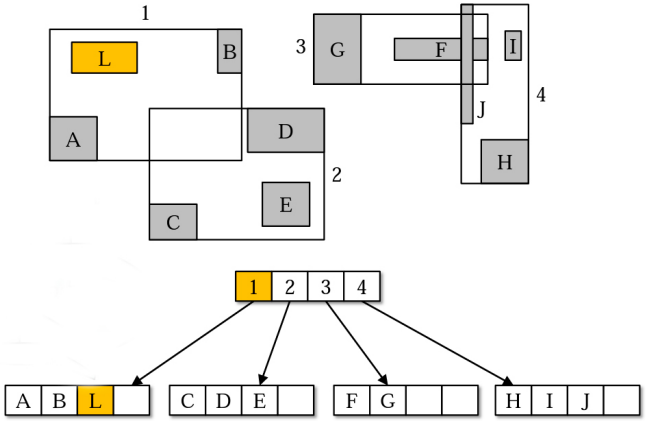


Figura: Inserir dado L em uma região já constituída (sem necessidade de criar ou dividir em novos espaços)

Exemplo 2 de inserção parte 1

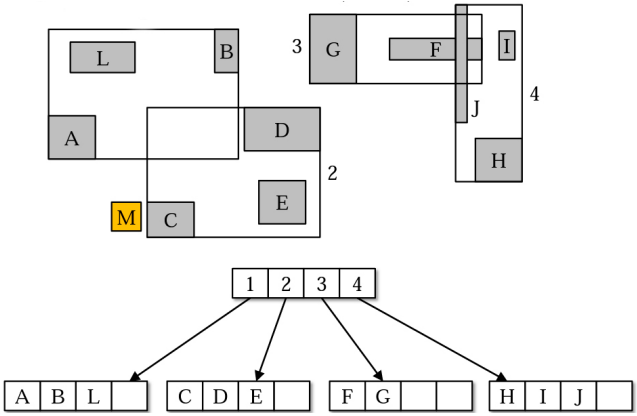


Figura: Inserir um dado M em que uma área terá que expandir para agrupá-lo)

Exemplo 2 de inserção parte 2

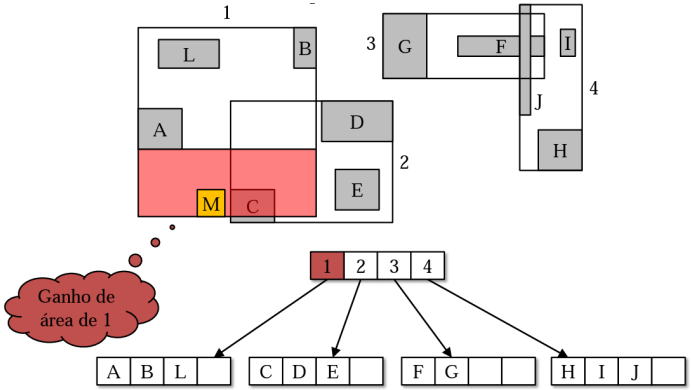


Figura: Inserir um dado M em que uma área terá que expandir para agrupá-lo)

Exemplo 2 de inserção parte 3

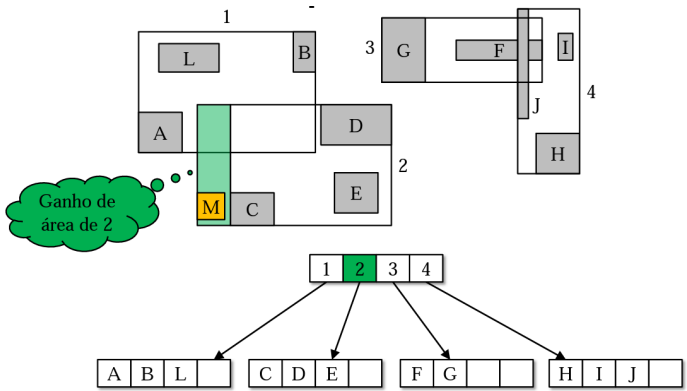


Figura: Inserir um dado M em que uma área terá que expandir para agrupá-lo)

Exemplo 2 de inserção parte 4

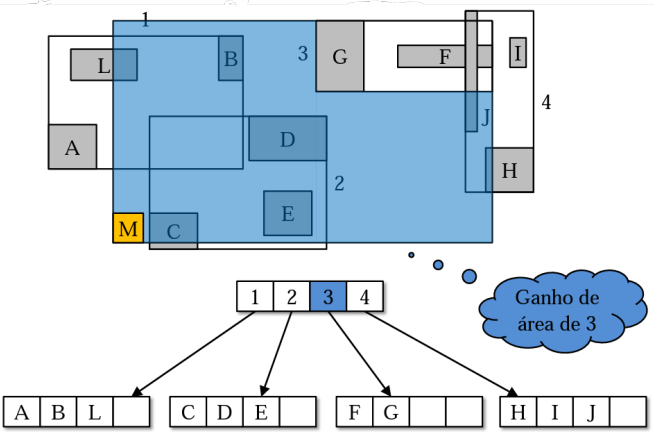


Figura: Inserir um dado M em que uma área terá que expandir para agrupá-lo)

Exemplo 2 de inserção parte 5

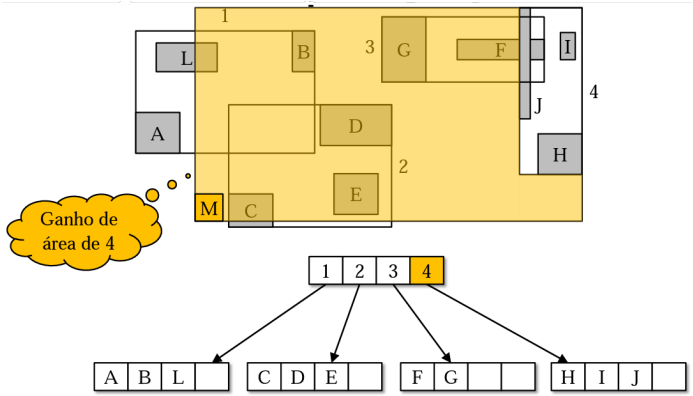


Figura: Inserir um dado M em que uma área terá que expandir para agrupá-lo)

Exemplo 2 de inserção parte 6

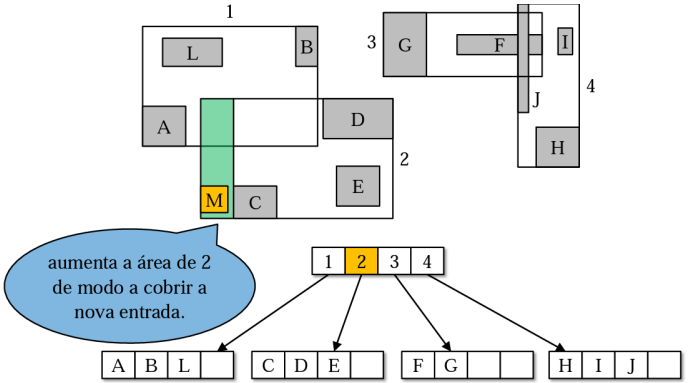


Figura: Inserir um dado M em que uma área terá que expandir para agrupá-lo)

Exemplo 2 de inserção parte 7

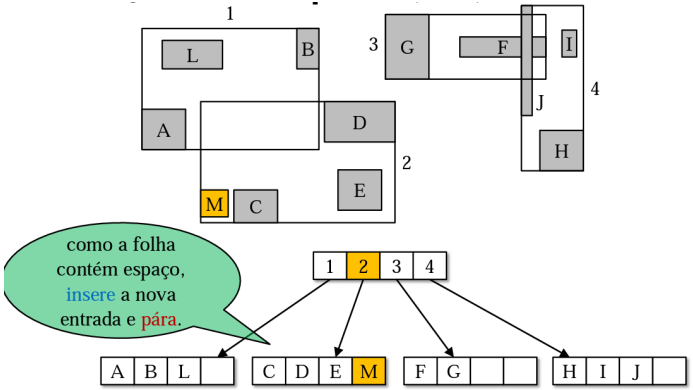


Figura: Inserir um dado M em que uma área terá que expandir para agrupá-lo)

Inserção na R-tree

- Caso o nodo folha F não tenha espaço, ele é *splitado* (dividido) em 2, F1 e F2.
  - A entrada de F no seu nodo pai (P) é ajustada de modo que seu MBR cubra apenas F1.
  - É adicionada uma entrada em P para F2. Este passo pode fazer com que o nó P seja *splitado* recursivamente.
- As alterações são propagadas para os níveis superiores.

Split: Algoritmo Quadrático

Parte 1: Seleção das seeds

- Seleciona dois objetos como seeds de modo que esses objetos, se colocados juntos, criam o maior dead space possível.
  - Dead space: área restante no MBR desconsiderando as seeds.
  - Dead space (A, B) = Área(MBR(A, B)) - Área (A) - Área (B)
- Complexidade: $O(M^2)$

Parte 2: Redistribuição das M-1 entradas

- Até que não reste mais entradas ($O(M)$), selecionar a entrada cuja diferença de dead space para cada um dos dois nodos N1 e N2 seja máxima ($O(M^2)$).
- Inserir a entrada no nodo que requer o menor aumento de seu MBR ($O(1)$).

Complexidade total de tempo: $O(M^2)$.

Split: Algoritmo Exaustivo

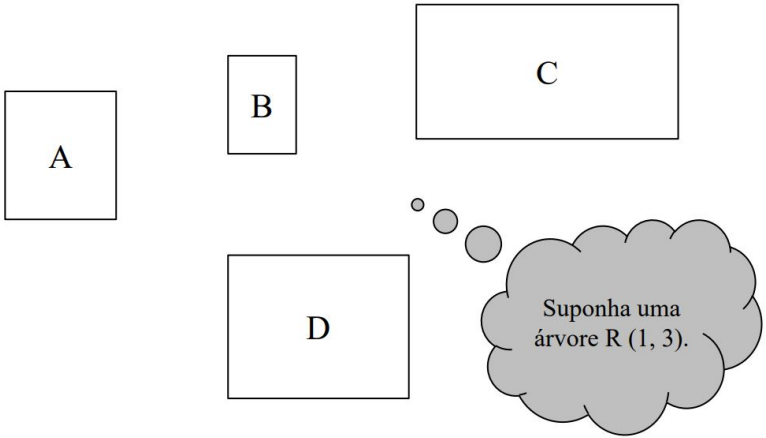
Tal algoritmo de divisão de nós pode-se fazer uma analogia com um algoritmo de força bruta, no qual ele irá testar todos os agrupamentos possíveis para encontrar o menor aumento de MBRs e área de sobreposição.

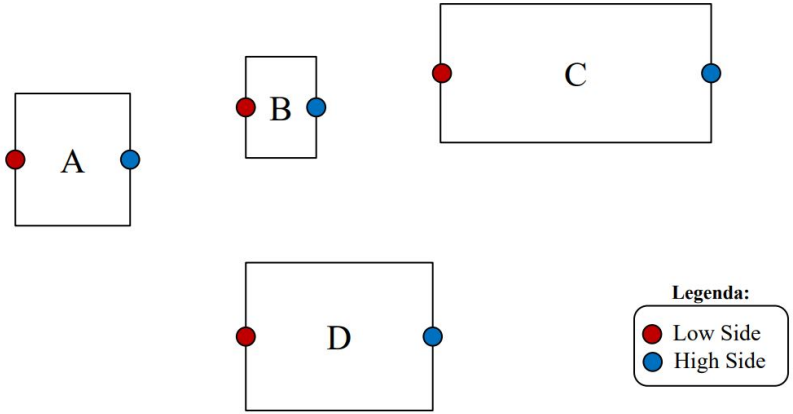
Possui uma complexidade de tempo de $O(2^{M-1})$

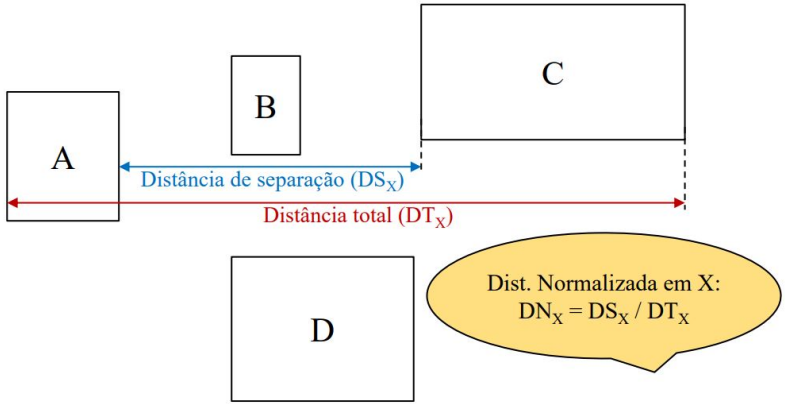
Split: Algoritmo Linear

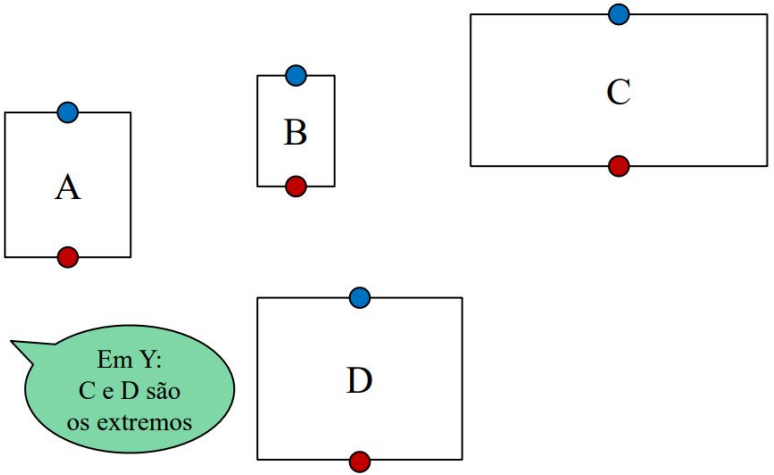
- Para cada dimensão, encontra a entrada com o maior tamanho lateral e o menor tamanho vertical, normalizando os valores
- Escolhe os dois objetos mais distantes entre si no nodo cheio, ou seja, os com o maior valor normalizado. Esses objetos serão os representantes dos novos nodos após a divisão.
- Distribua os objetos restantes entre os dois grupos, atribuindo cada objeto ao grupo com o representante mais próximo.
- Crie dois novos nodos e coloque os objetos nos grupos correspondentes nesses nodos.

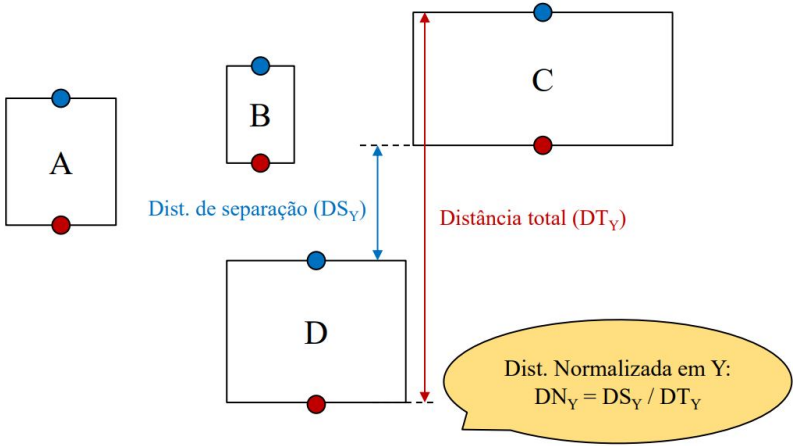
Algoritmo Linear

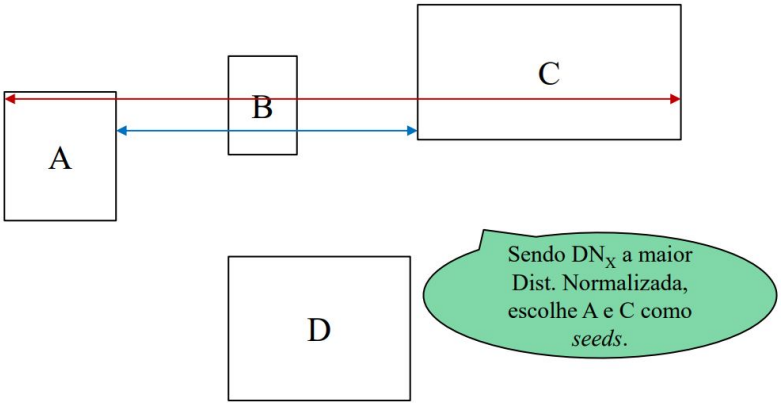


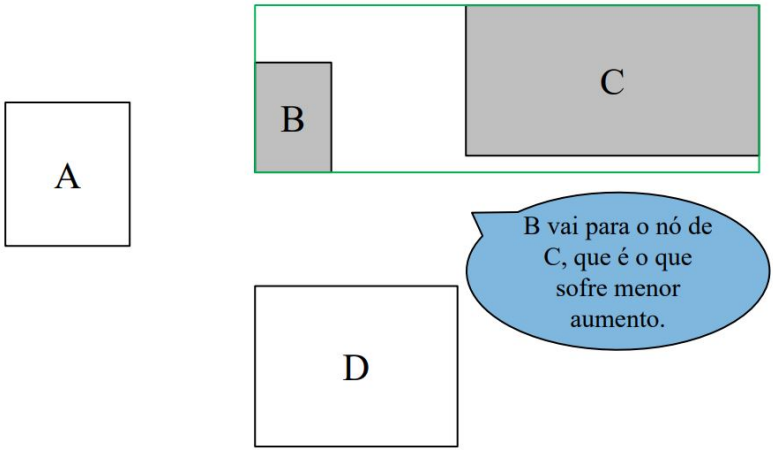

A
B
C
D
Legenda:
Low Side
High Side

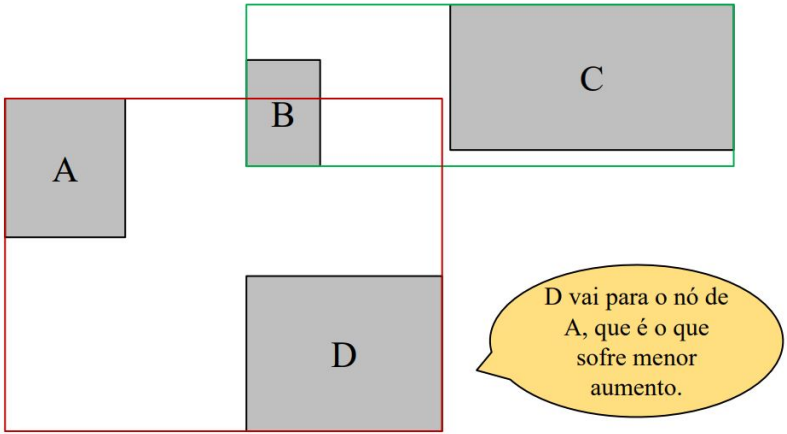

A
B
C
D
Distância de separação (DS_X)
Distância total (DT_X)
Dist. Normalizada em X:
$DN_X = DS_X / DT_X$

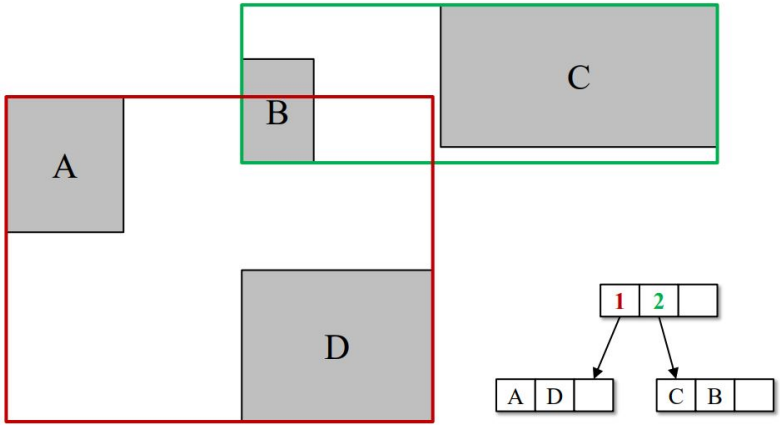

A
B
C
D
Em Y:
C e D são
os extremos


A
B
C
D
Dist. de separação (DS_Y)
Distância total (DT_Y)
Dist. Normalizada em Y:
$DN_Y = DS_Y / DT_Y$


A
B
C
D
Sendo DN_X a maior Dist. Normalizada, escolhe A e C como *seeds*.


A
B
C
D
B vai para o nó de C, que é o que sofre menor aumento.


C
B
A
D
D vai para o nó de A, que é o que sofre menor aumento.


A
B
C
D
1
2
A
D
C
B

Variações da R-tree

Variações da Árvore R

Com o tempo, várias variações e aprimoramentos da árvore R foram desenvolvidas para deixar ela top de linha. Dentre elas:

- Árvore R de prioridade
- Árvore R*
- Árvore R+
- Árvore R Hilbert
- Árvore X

Desvantagens

Limitações da R-tree

- Quando ocorre a sobreposição de caixas delimitadores, é preciso verificar mais ramificações para encontrar o item desejado e isso aumenta o custa da consulta, levando a um processo mais lento.
- A implementação do algoritmo da R-Tree é complexo, já que é necessário usar algumas estratégias para manter o balanceamento da arvore e assim evitar ao máximo a sobreposição.
- Há uma dificuldade desta estrutura de dados, em lidar com dados de alta dimensão. Apesar de teoricamente a R-Tree suportar tais dados, isso diminui a performance porque aumenta o numero de dimensões da arvore, aumentando a possibilidade de sobreposição das caixas.

Referências

Referências utilizadas

- MANOLOPOULOS, Y.; NANOPOULOS, A.; PAPADOPOULOS, A. N.; THEODORIDIS, Y. R-Trees: Theory and Applications. Springer, 2006.
- KAO, M.-Y. Encyclopedia of Algorithms. Springer, 2008.
- Introduction to R tree. GeeksforGeeks. Disponível em: <https://www.geeksforgeeks.org/introduction-to-r-tree/>. Acesso em: 17 ago. 2023.
- WIKIPEDIA CONTRIBUTORS. R-tree. Wikipedia. Disponível em: <https://en.wikipedia.org/wiki/R-tree>. Acesso em: 17 ago. 2023.

Referências utilizadas

- Árvore R Luiz Olmes Carvalho SCC 5789 -Base de Dados Profa. Dra. Cristina Dutra de Aguiar Ciferri. [s.l.: s.n., s.d.]. Disponível em: <http://wiki.icmc.usp.br/images/c/cb/SCC578920131-Rtree.pdf>. Acesso em: 17 ago. 2023.
- R-Trees. Cglab.ca. Disponível em: <https://cglab.ca/ cdillaba/comp5409_project/R_Trees.html>. Acesso em: 17 ago. 2023.