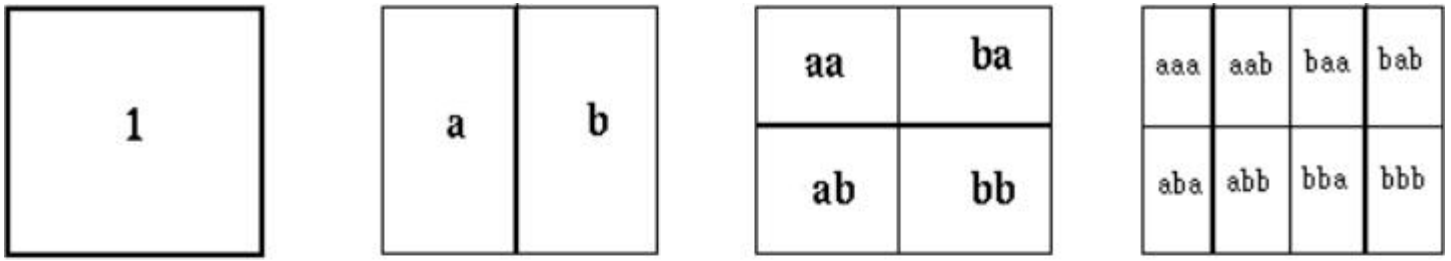
Árvores Point-Quad, Polygon-Quad e R

Eduardo Furlan Miranda

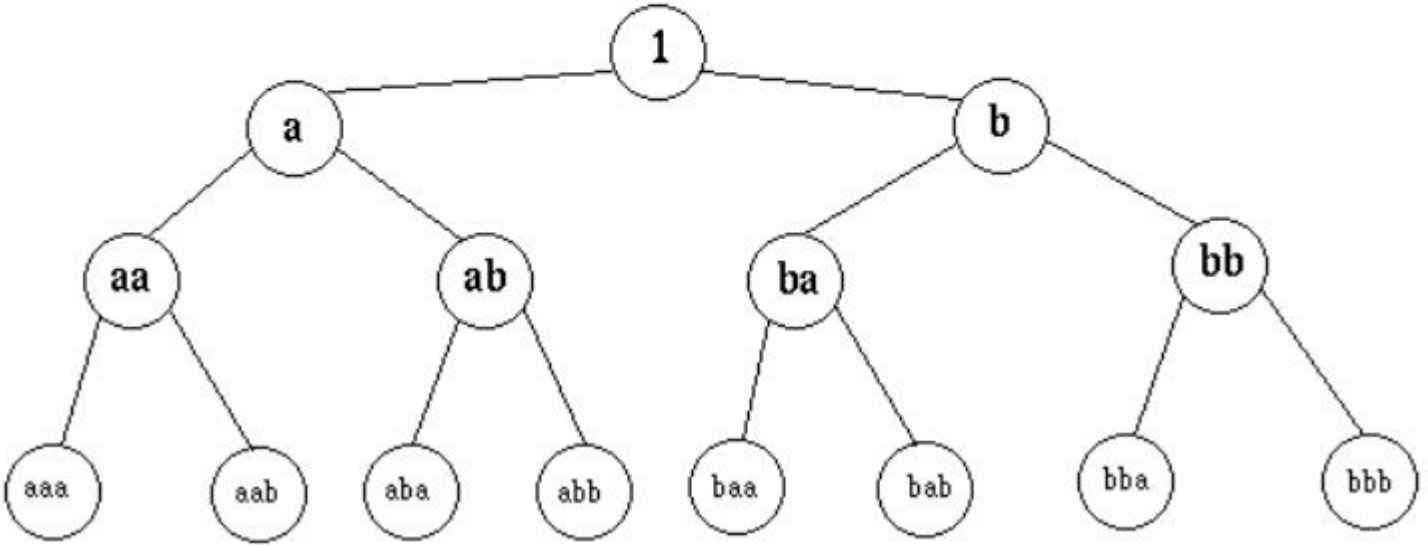
2024-02-04

Adaptado dos materiais dos Profs. J. Nikander, A. Conci, e J. Pei

Exemplo de divisão do espaço

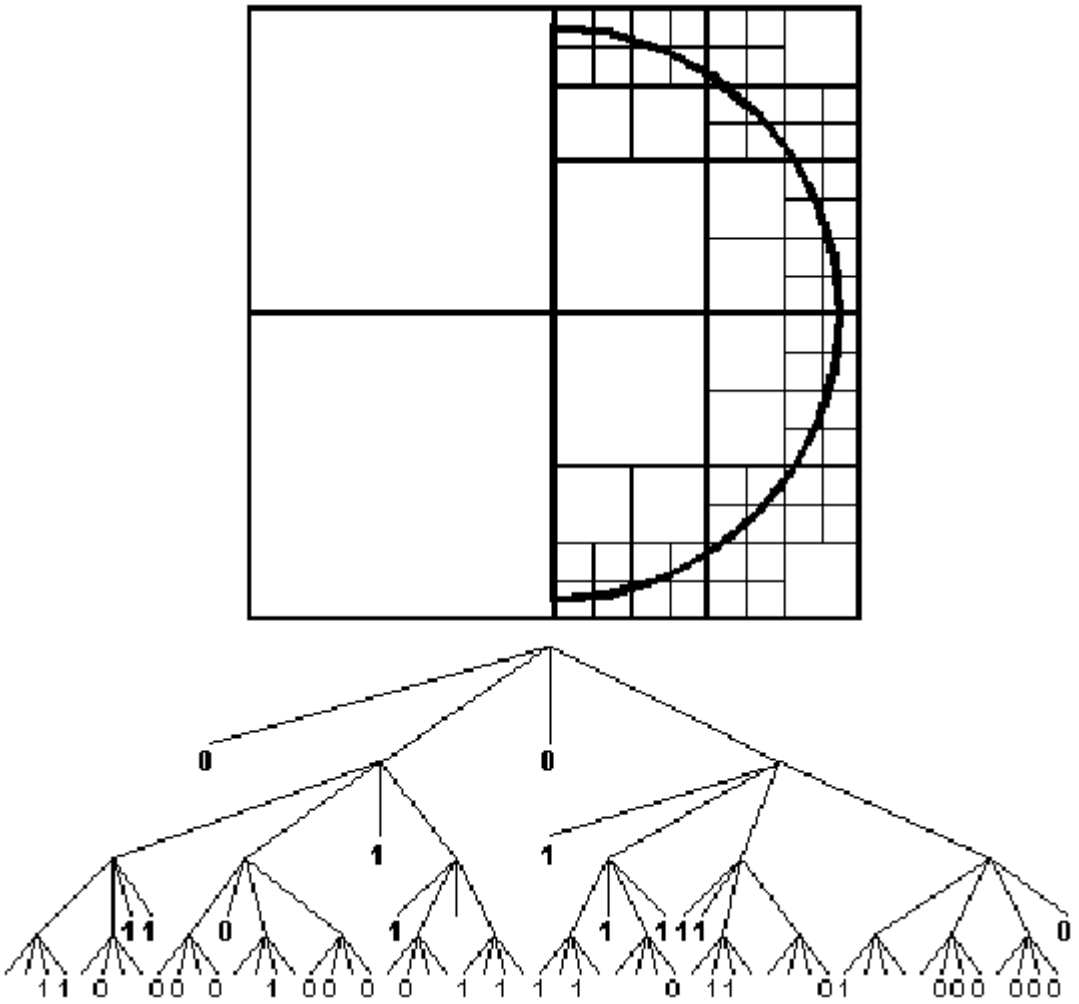


Árvore binária



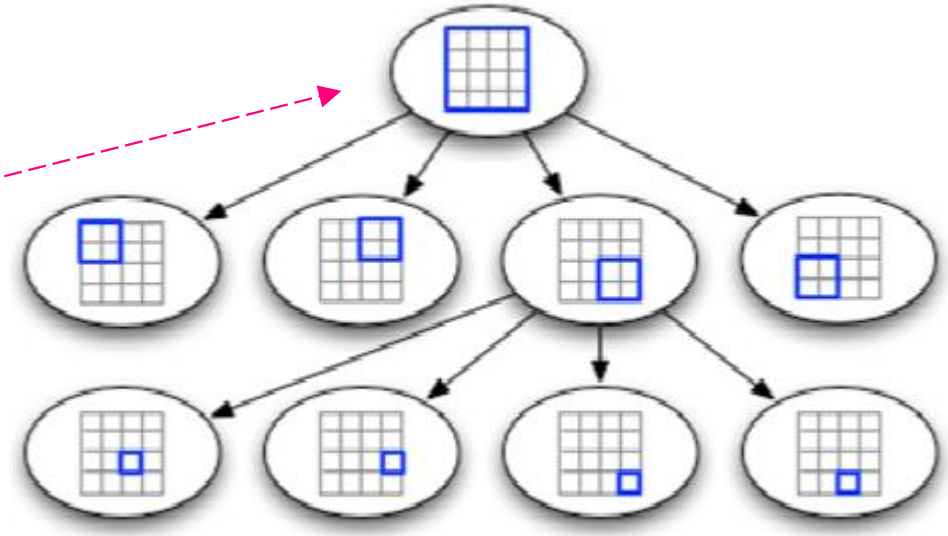
Divisão do espaço através de divisões sucessivas

Quad-Tree

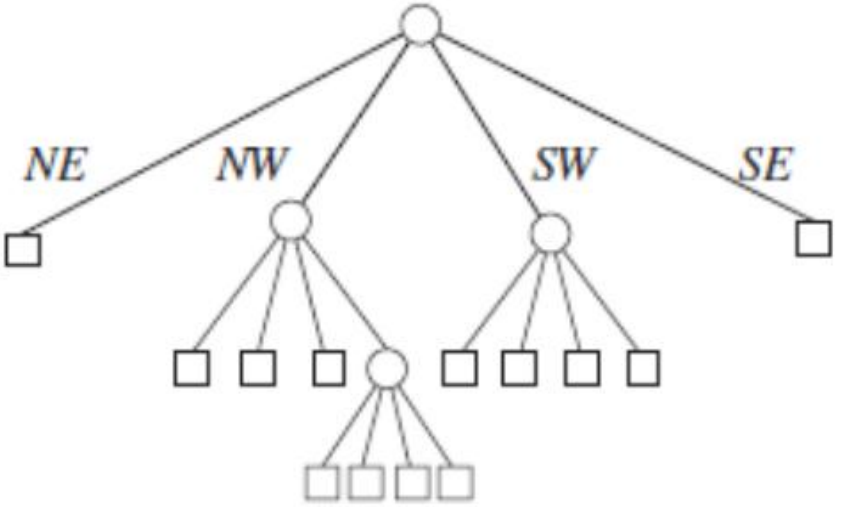


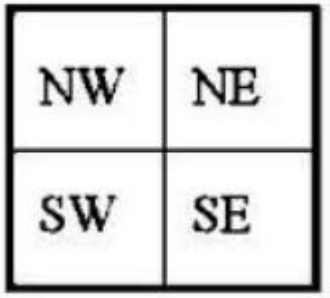
- Usado para representar
  - Pontos (vetores)
  - Área (raster ou bitmap)
- 2D: quadtree
- 3D: octree

Quad-Tree



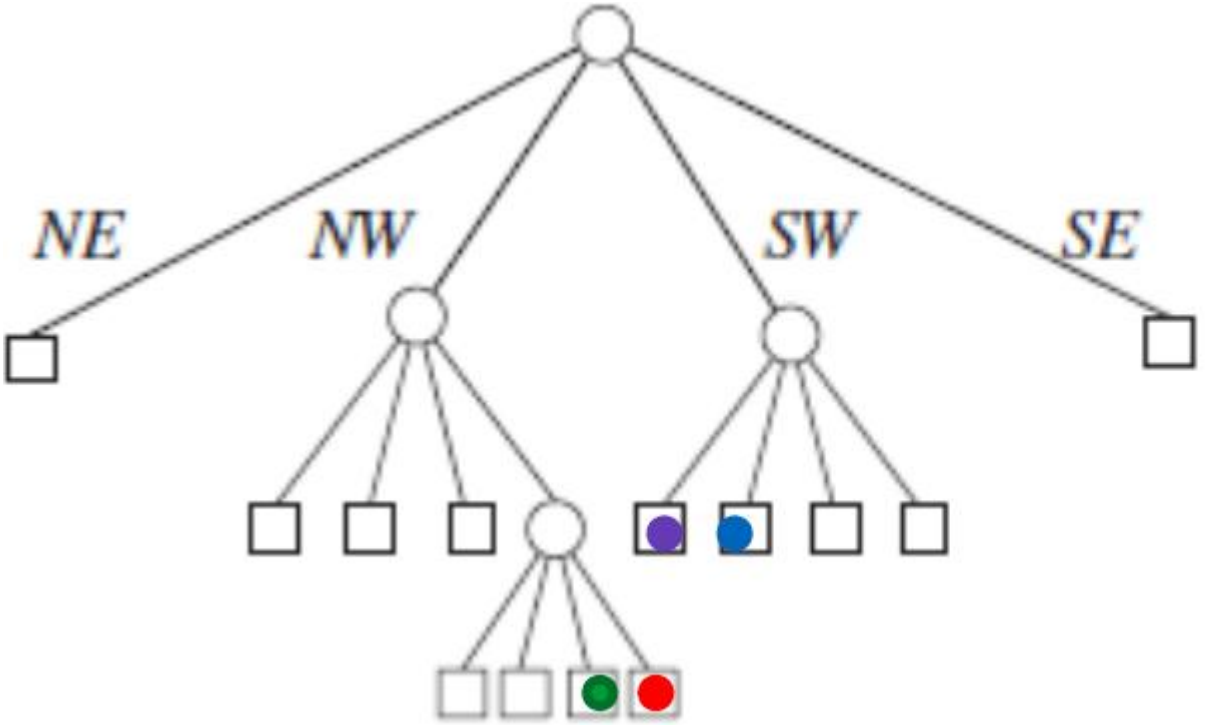
- Raiz = ordem completa de elementos da figura
- Cada filho de um nó representa um quadrante
  - NW (nordeste)
  - NE (noroeste)
  - SW (sudoeste)
  - SE (sudeste)

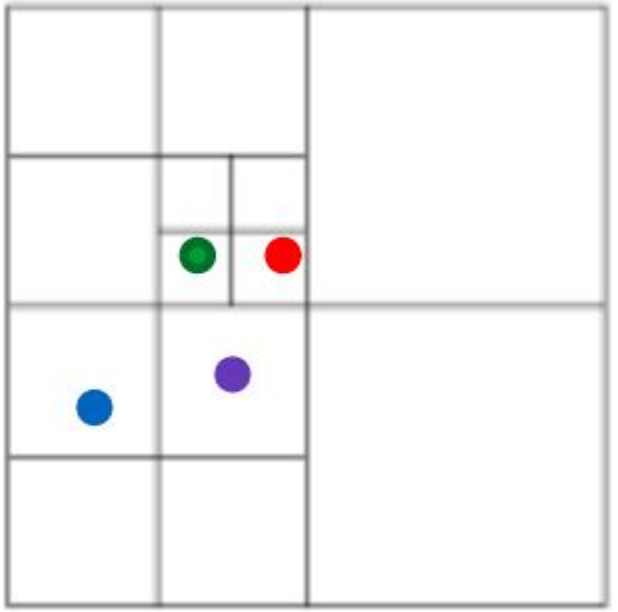




- Nós de folha da árvore
  - Correspondem a blocos para os quais nenhuma subdivisão adicional é necessária

Quad-Tree

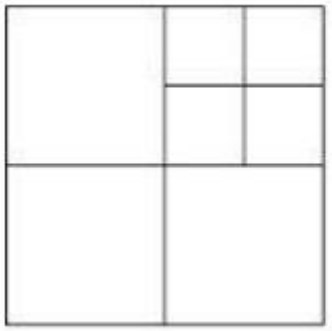


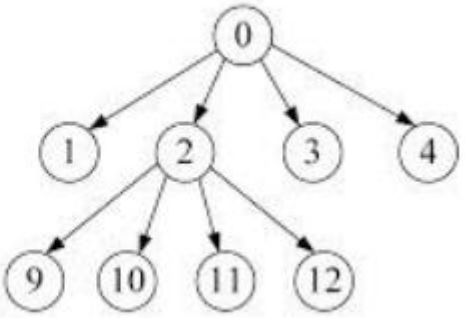


Quad-Tree

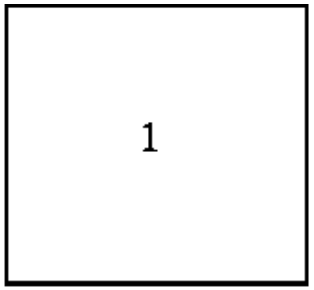
- O objeto é envolvido por um quadrado
  - Em seguida é dividido em quadrados menores (quadrantes)
- Cada um é classificado como
  - **Cheio**: o objeto ocupa todo o quadrante
  - **Vazio**: o objeto não ocupa nenhuma parte do quadrante
  - **Cheio-Vazio**: o objeto ocupa parte do quadrante
- Cheio-Vazio
  - É novamente dividido em 4 partes iguais e o processo repete até que só existam **cheio** ou **vazio**

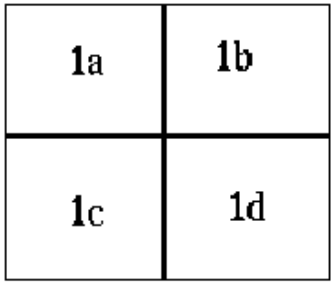
Codificação





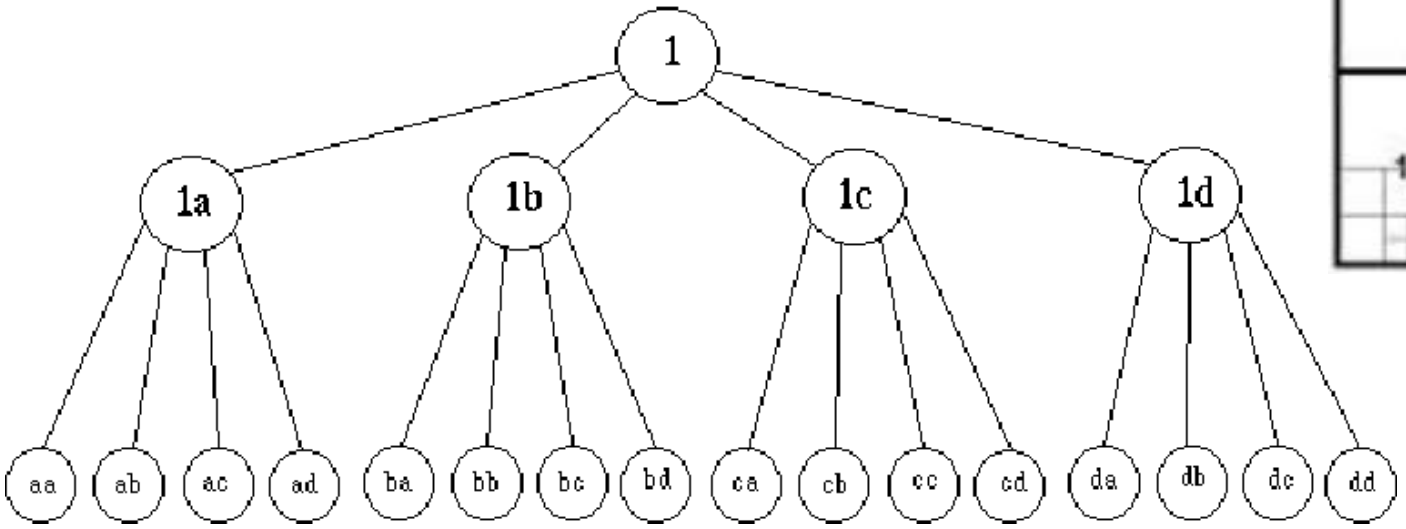
- Usa-se alguma forma de indexação e codificação da estrutura final

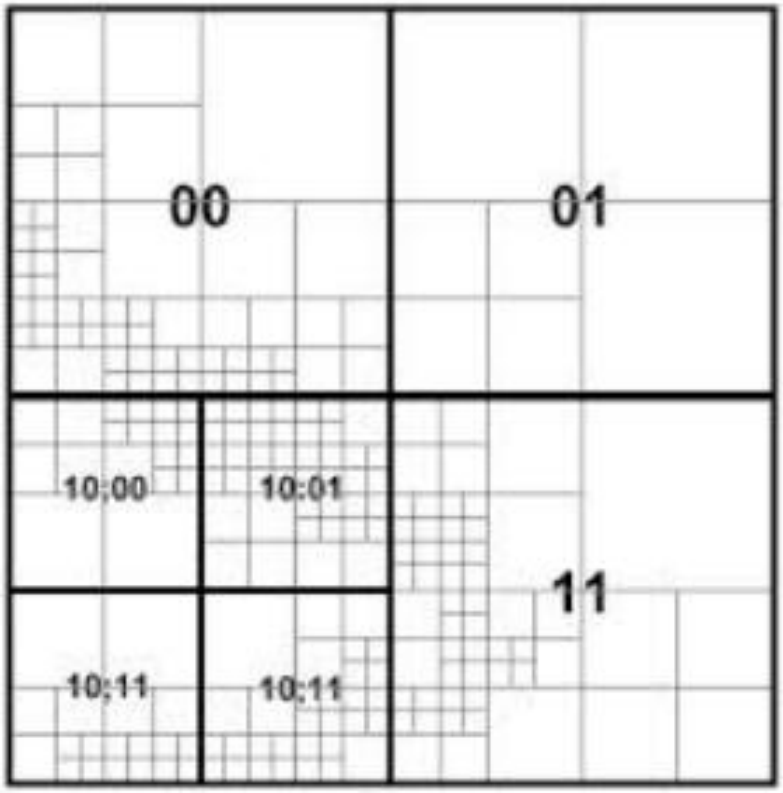




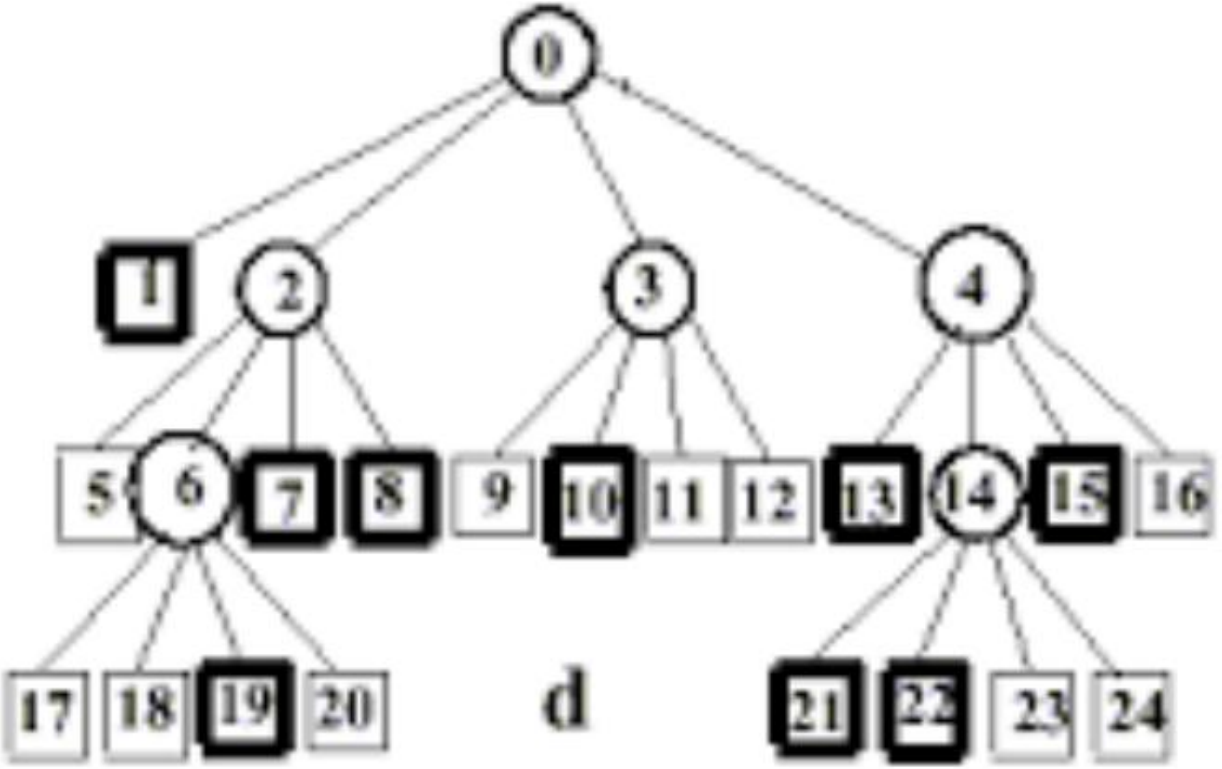
aa	ab	ba	bb
ac	ad	bc	bd
ca	cb	da	db
cc	cd	dc	dd

A quadtree





Codificação da estrutura final



Full

Empty

Partial

11	13	31	33
10	12	30	32
01	03	21	23
00	02	20	22

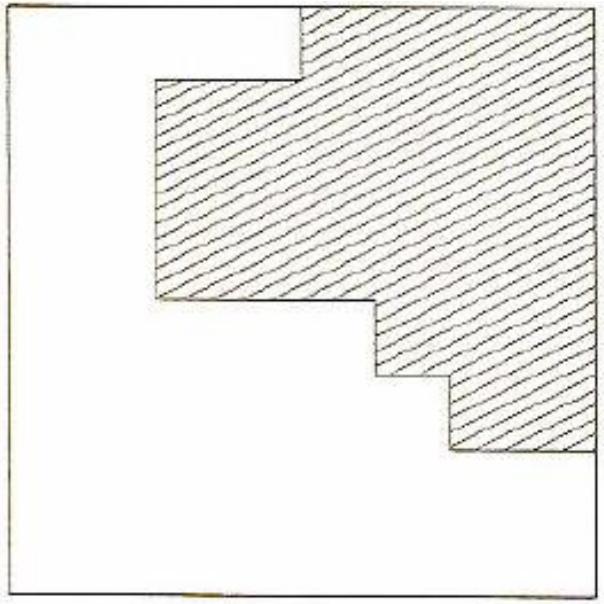
Level 0

Level 1

Level 2

Exemplo

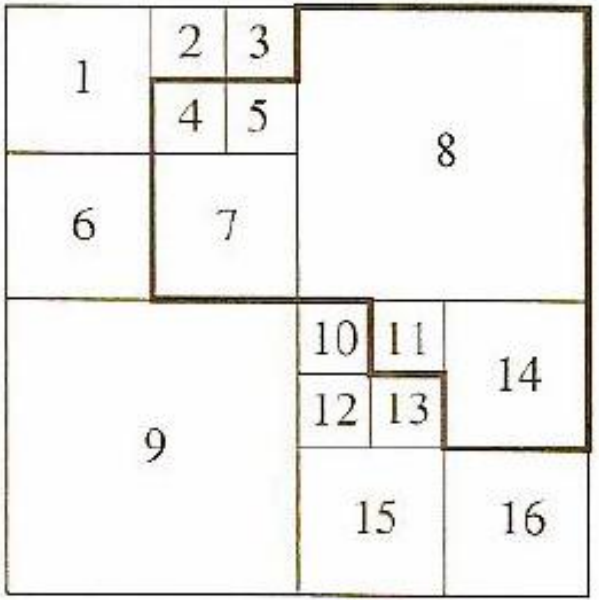
0 = Universo (está fora do objeto)
1 = Objeto



Forma do objeto original

0	0	0	0	1	1	1	1
0	0	1	1	1	1	1	1
0	0	1	1	1	1	1	1
0	0	1	1	1	1	1	1
0	0	0	0	0	1	1	1
0	0	0	0	0	0	1	1
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Representação binária



Depois da divisão em 4tree

Exemplo

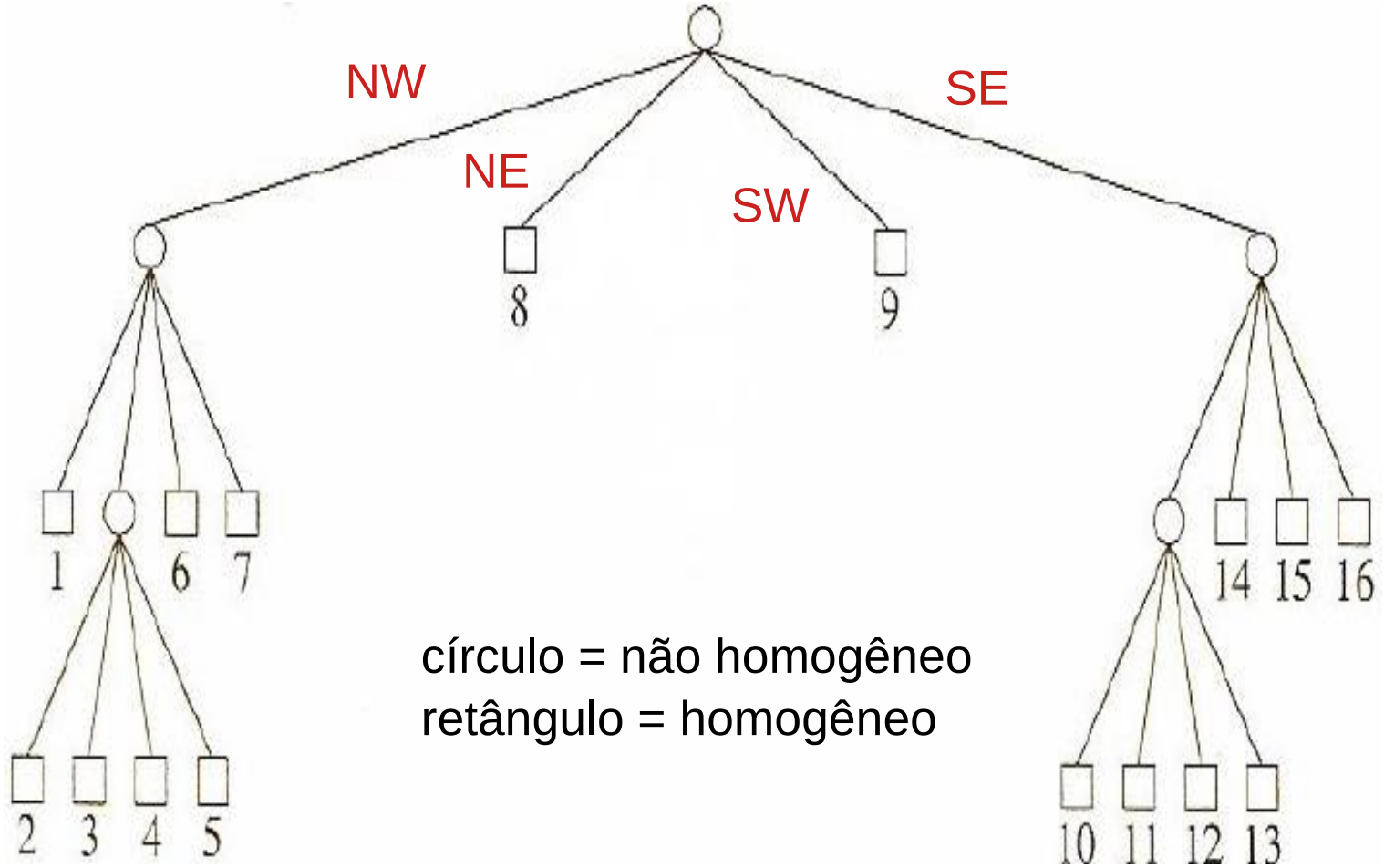
NW	NE
SW	SE

Nível 3

Nível 2

Nível 1

Nível 0



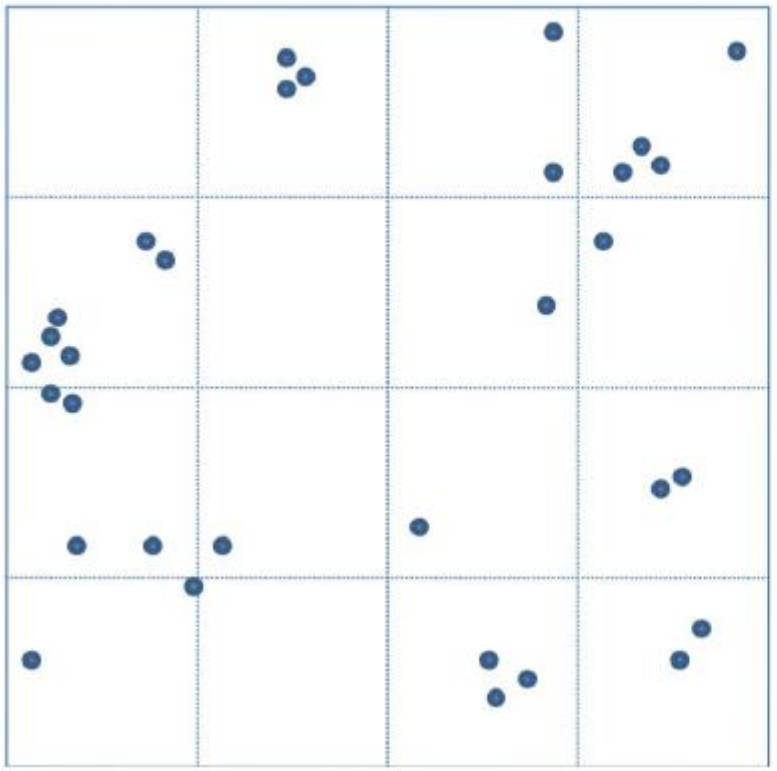
círculo = não homogêneo
retângulo = homogêneo

Quad-Tree

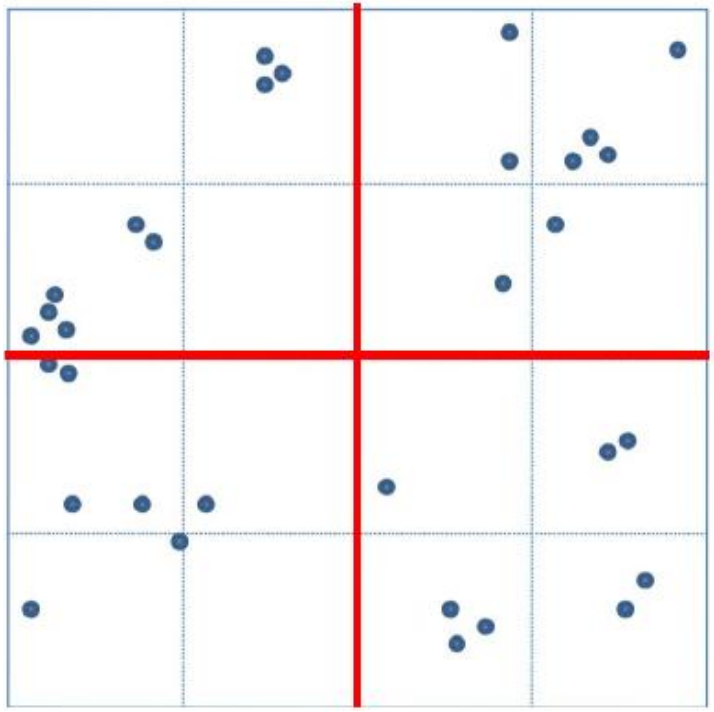
- A estrutura Quad-Tree não é adequada para armazenar dados distribuídos de forma desigual
- São necessários ‘truques’ de implementação para evitar que a árvore tenha um grande número de folhas vazias

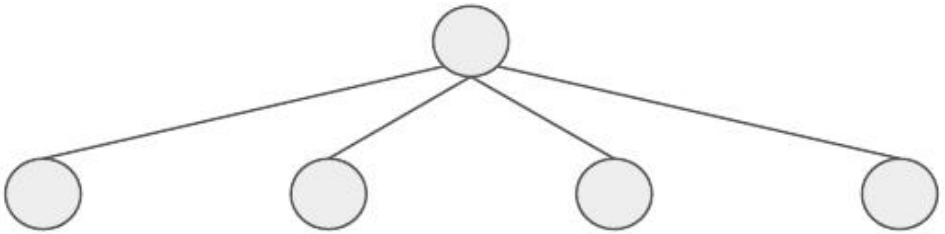
Exemplo

Raiz da árvore

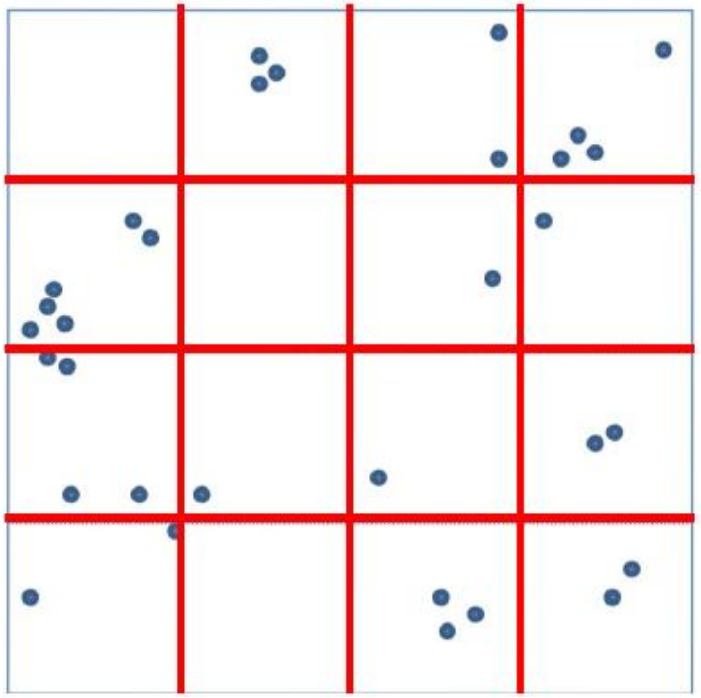


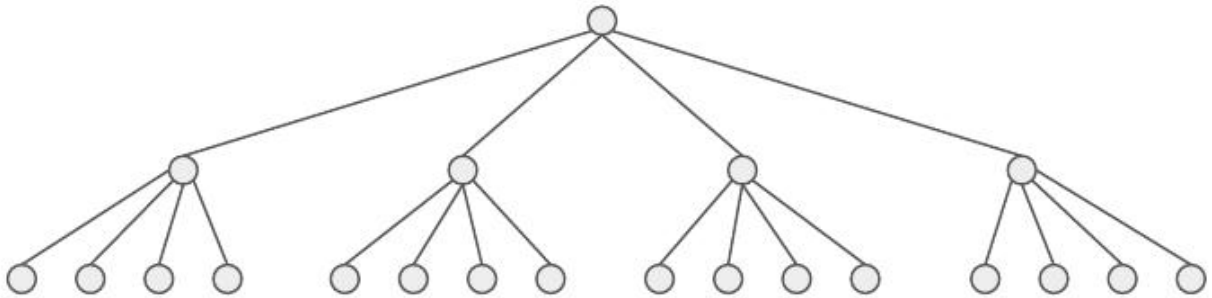
Exemplo - primeira divisão



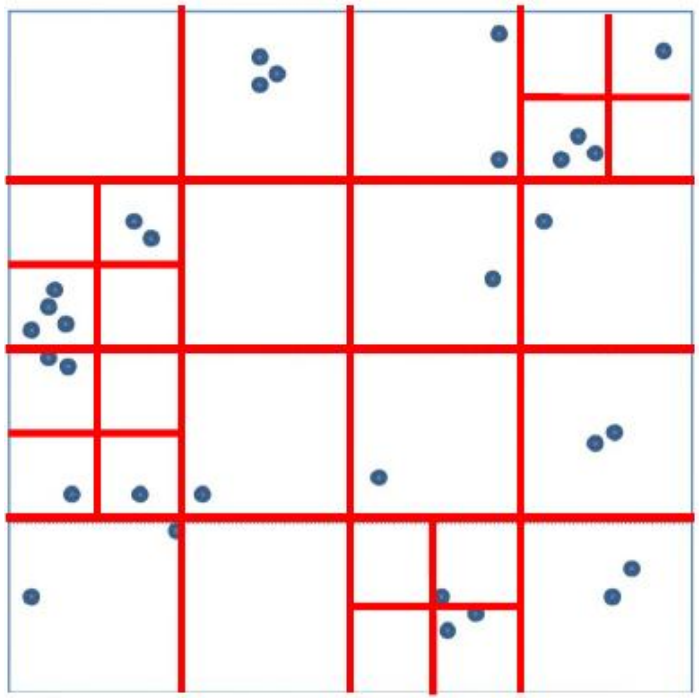


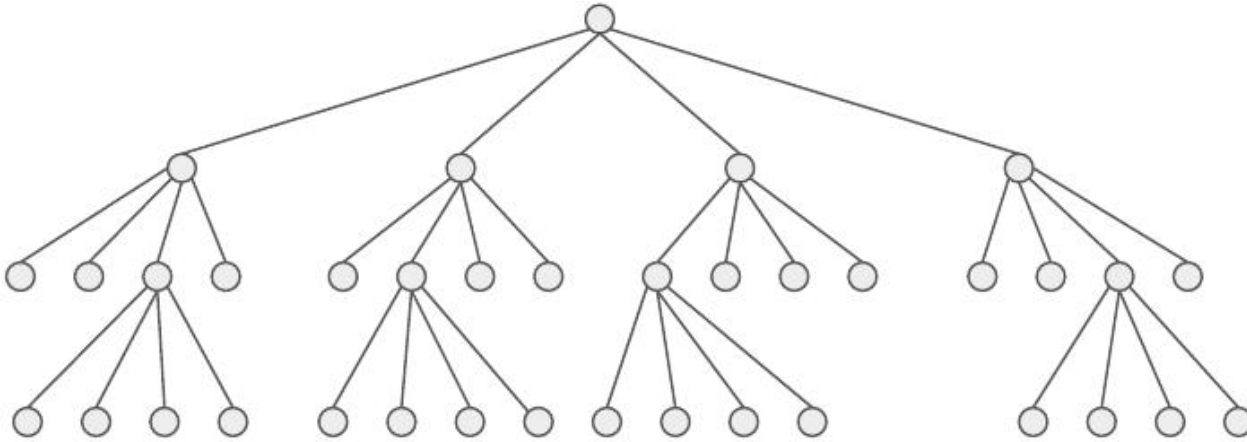
Exemplo - segunda divisão



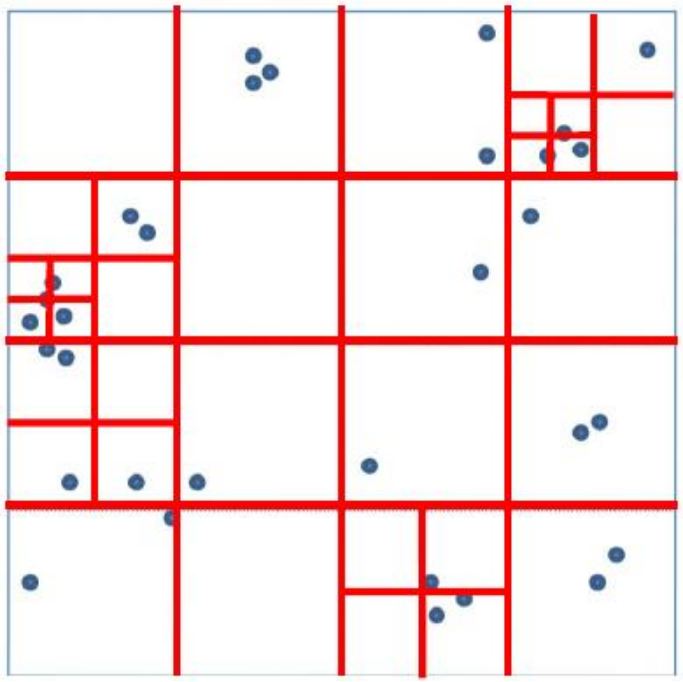


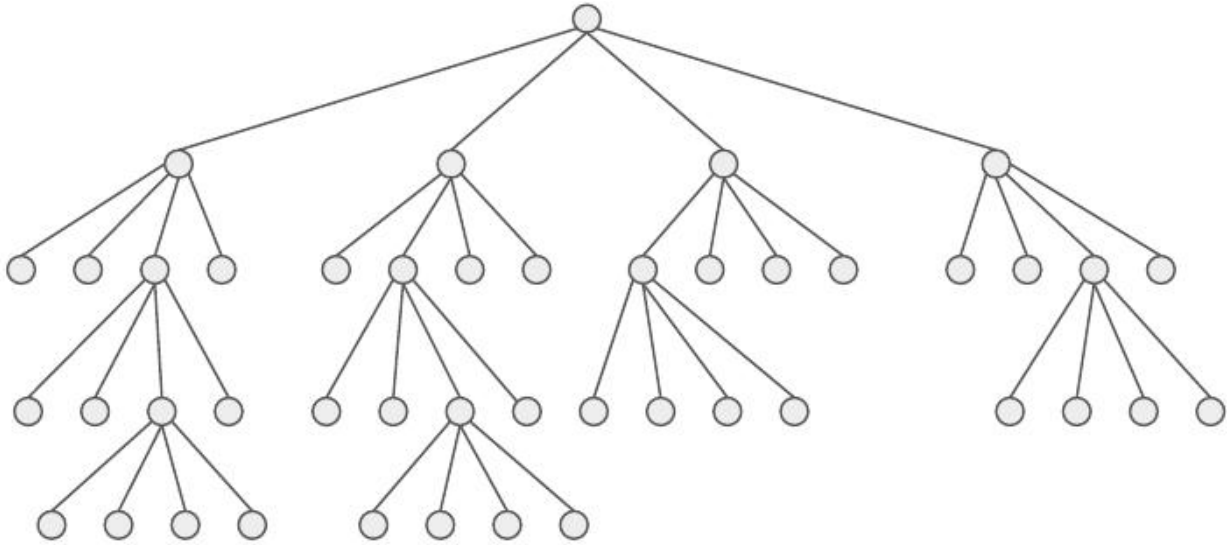
Exemplo - terceira divisão





Exemplo - quarta divisão





- Quad-Trees são desbalanceadas
- Cada nó tem exatamente 4 filhos

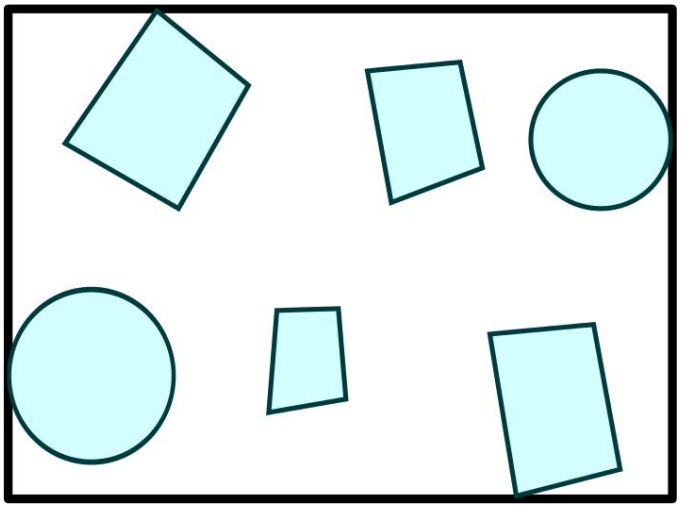
Cada partição contém pelo menos 2 objetos ← atingido o critério de parada

Quadtree para polígonos

MBR - Retângulo Delimitador Mínimo

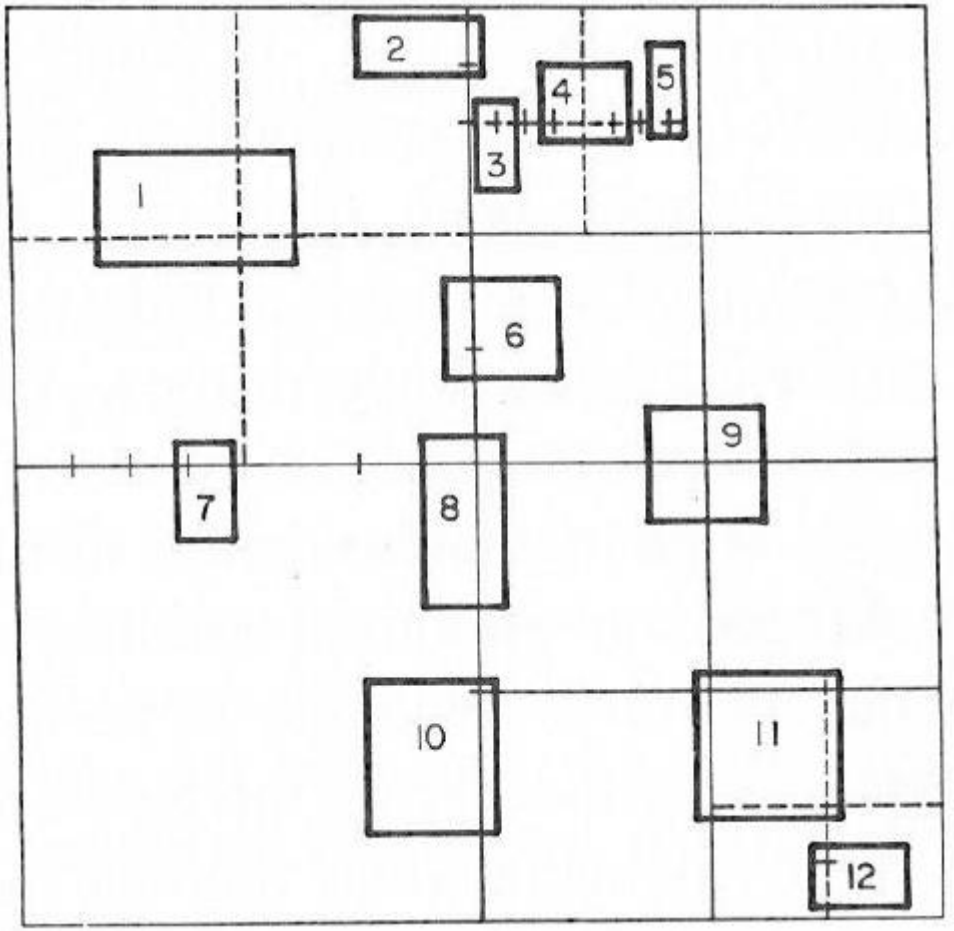
- É uma expressão das extensões máximas de um objeto bidimensional (p.ex., ponto, linha, polígono) ou conjunto de objetos dentro do seu sistema de coordenadas x-y

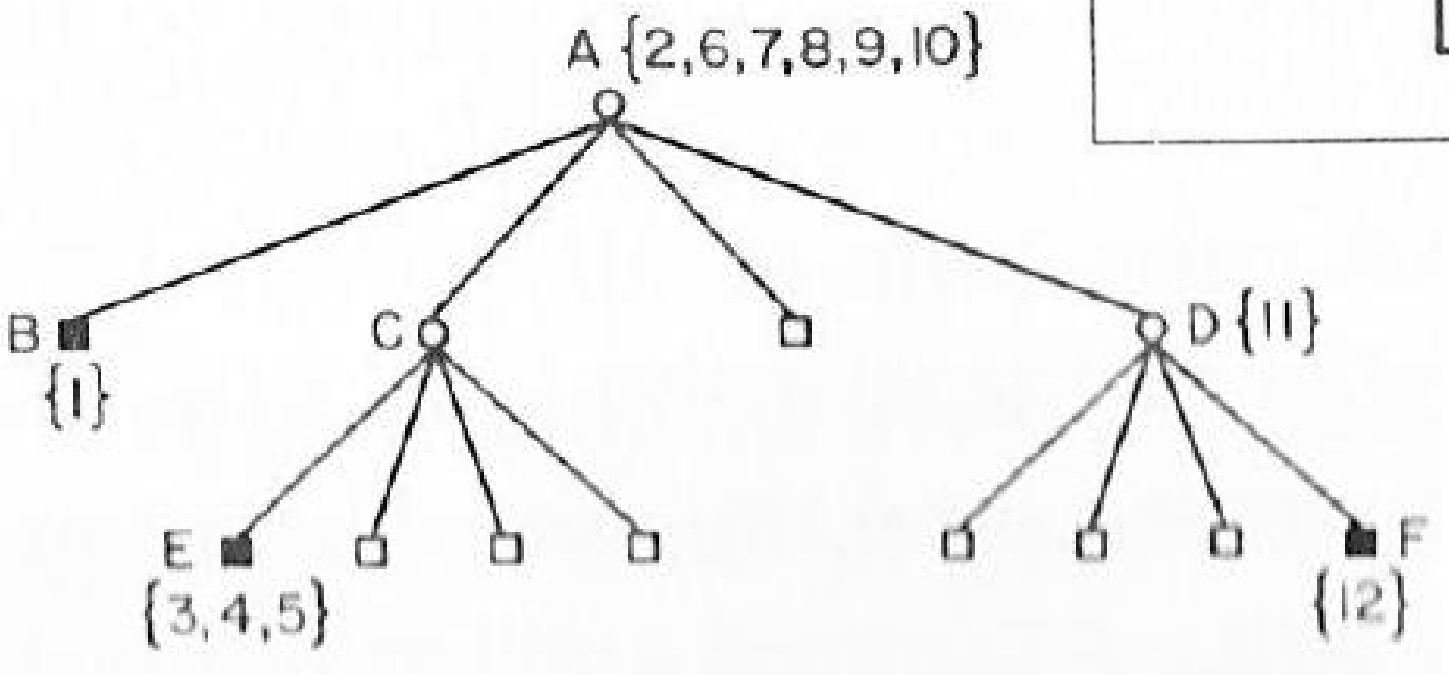
- É um caso bidimensional da caixa delimitadora mínima



- Usado como
  - Indicação da posição geral de uma característica geográfica ou conjunto de dados
  - Consulta espacial de primeira aproximação
  - Indexação espacial

Polygon Quad-Tree





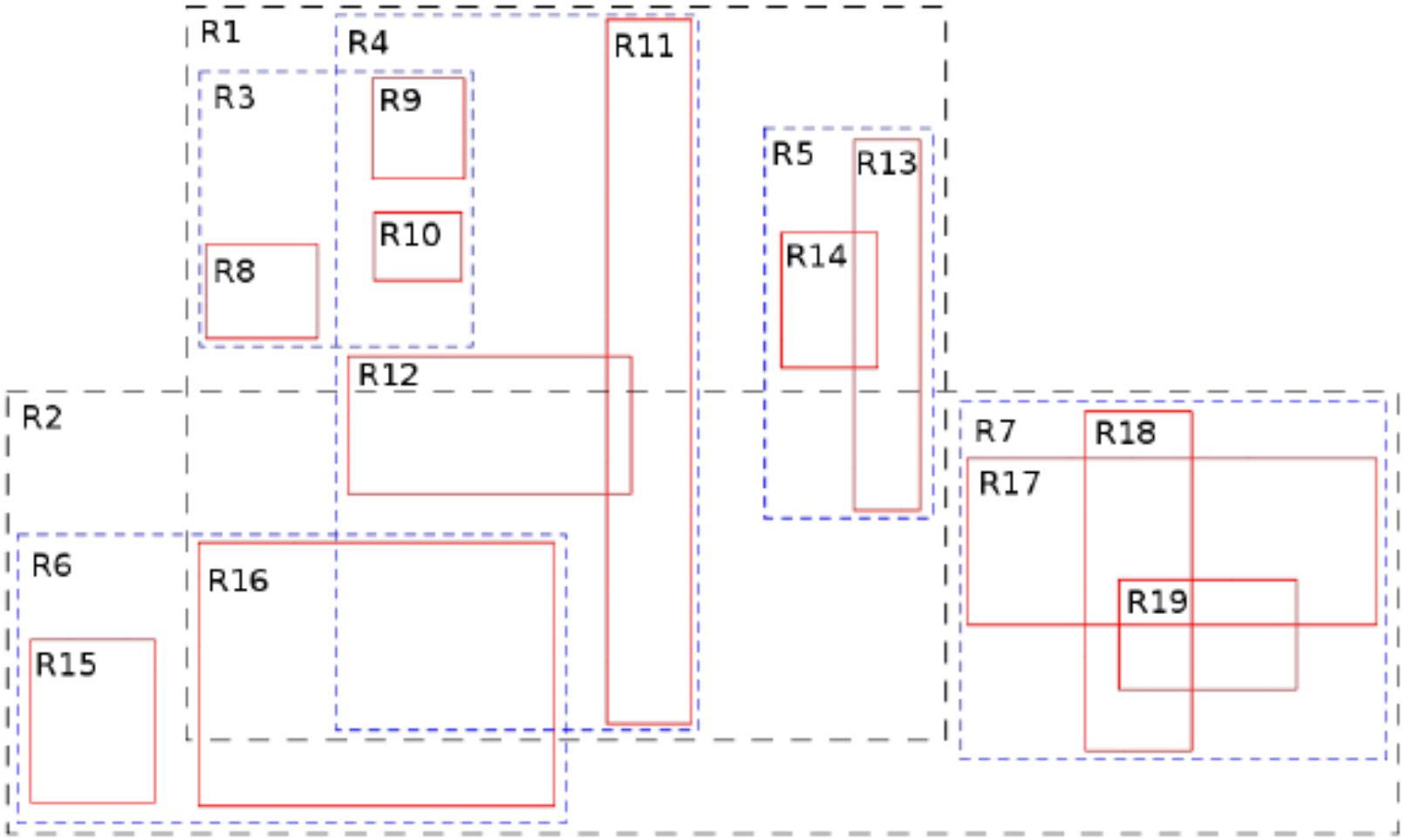
Quad-Tree para polígonos

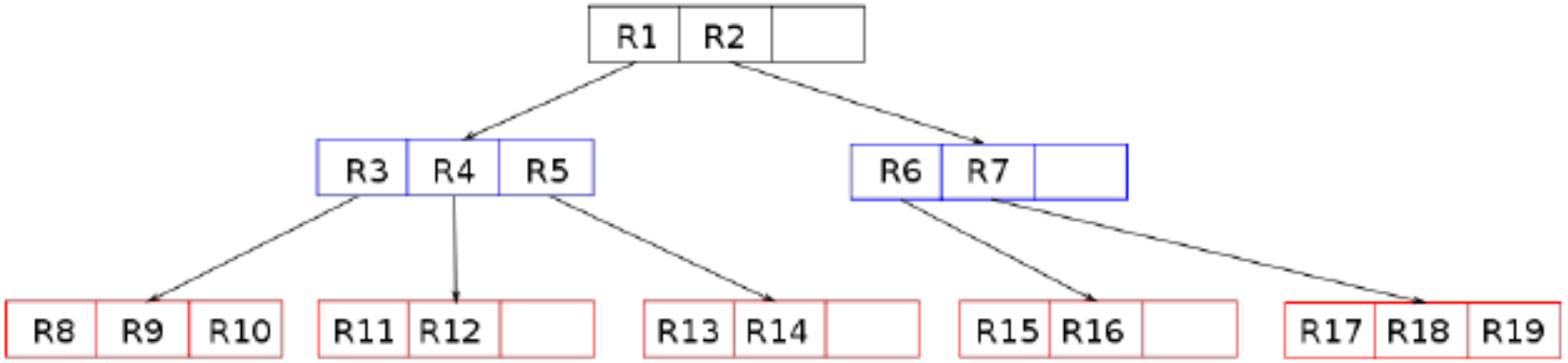
- A quad-tree básica contém elementos que sempre se ajustam completamente dentro de um nó
- Polígonos podem abranger a área coberta por vários nós
- Polígonos são aproximados por seus MBRs
- Cada item de dados é armazenado no nó quad-tree
  - Correspondente à menor subdivisão que cobre o MBR
  - Cada item de dados é armazenado apenas uma vez
- Um nó pode conter vários itens de dados

Quad-Tree para polígonos

- Em cada nó
  - O número de itens de dados não está vinculado
    - É necessário um índice para os itens no nó
  - Precisa de uma estrutura de dados auxiliar para indexar os dados
- Qualquer consulta de janela que se sobreponha um determinado nó
  - Deve pesquisar a estrutura auxiliar para ver quais itens de dados no nó se sobrepõem à janela de consulta

R-Tree


R1
R4
R11
R3
R9
R5
R13
R10
R14
R8
R12
R2
R7
R18
R17
R6
R16
R19
R15


R1
R2
R3
R4
R5
R6
R7
R8
R9
R10
R11
R12
R13
R14
R15
R16
R17
R18
R19

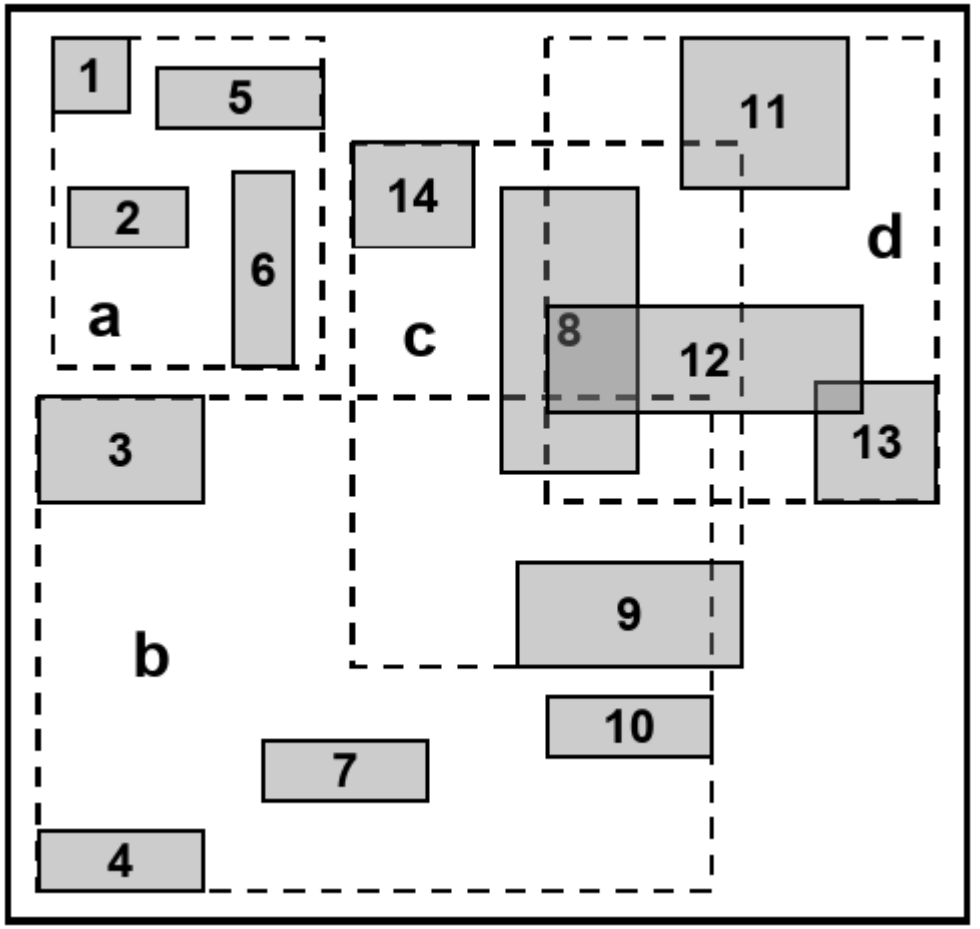
R-Tree

- Árvore balanceada para valores-chave multidimensionais
- Os itens de dados são aproximados usando MBRs
- Projetado para ser usado com dispositivos de armazenamento externos
- É uma estrutura de indexação comum em bancos de dados que armazenam dados multidimensionais
  - P. ex. bancos de dados espaciais
- Diferente de Quad-Tree, é construído de baixo para cima

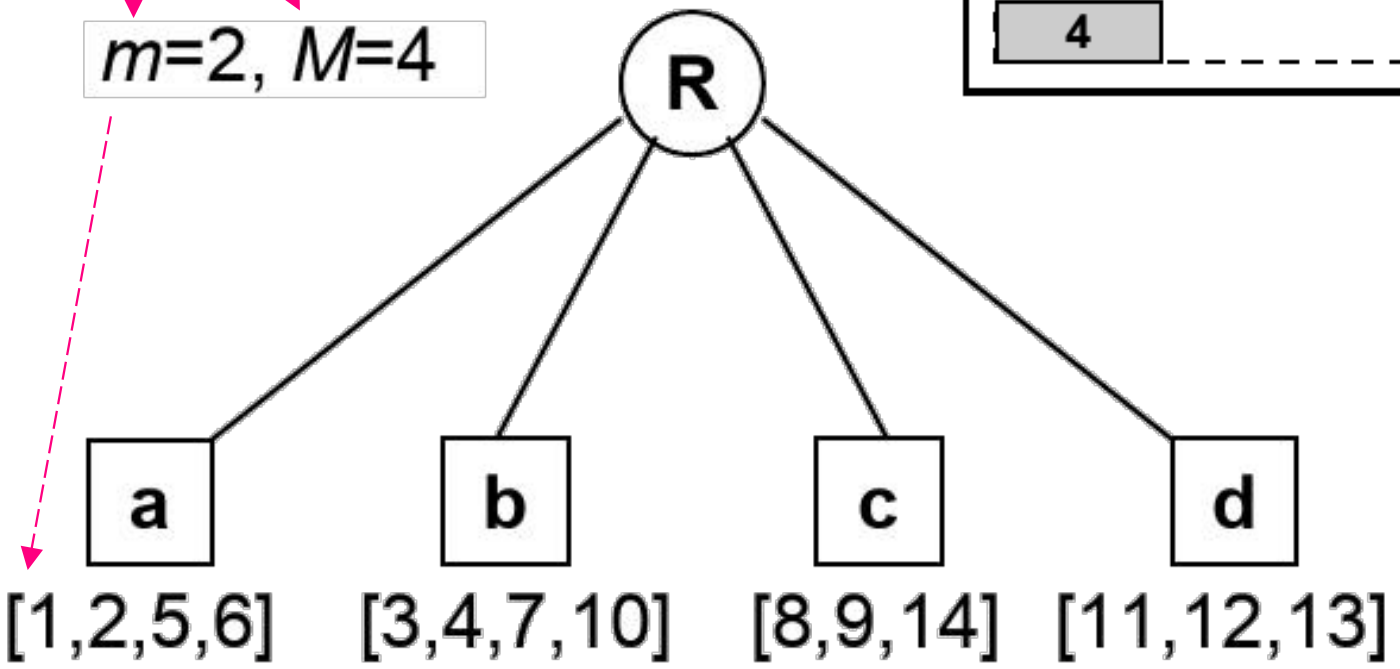
R-Tree

- R-Tree é uma árvore de pesquisa não binária balanceada
  - Cada nó (exceto raiz) possui entre m e 2m(=M) elementos de dados
  - Cada nó interno (exceto raiz) tem entre m e 2m nós filhos
  - A raiz tem 0 a 2m elementos e 0 a 2m filhos
  - Cada folha está no mesmo nível
- Como um único nó contém um grande número de elementos
  - A árvore é larga, mas curta
    - Apenas alguns links da raiz à folha

R-Tree



Quantidade de elementos de dados
- mínimo
- máximo



R-Tree

- Os nós R-Tree, com exceção da raiz, normalmente estão em armazenamento externo
- Como a árvore é curta, apenas algumas operações de E/S são necessárias por operação da árvore
- Cada nó possui um grande número de elementos
  - São necessários meios de pesquisa em um nó

R-Tree

- A eficiência da árvore R depende da distribuição de dados e da ordem de inserção (tem grande efeito nos MBRs)
  - **Cobertura**: a área total coberta por nós em um determinado nível
  - **Sobreposição**: a quantidade de área coberta por mais de um nó
  - Minimizar ambos tornará a árvore mais eficiente
  - Especialmente a minimização da sobreposição é importante

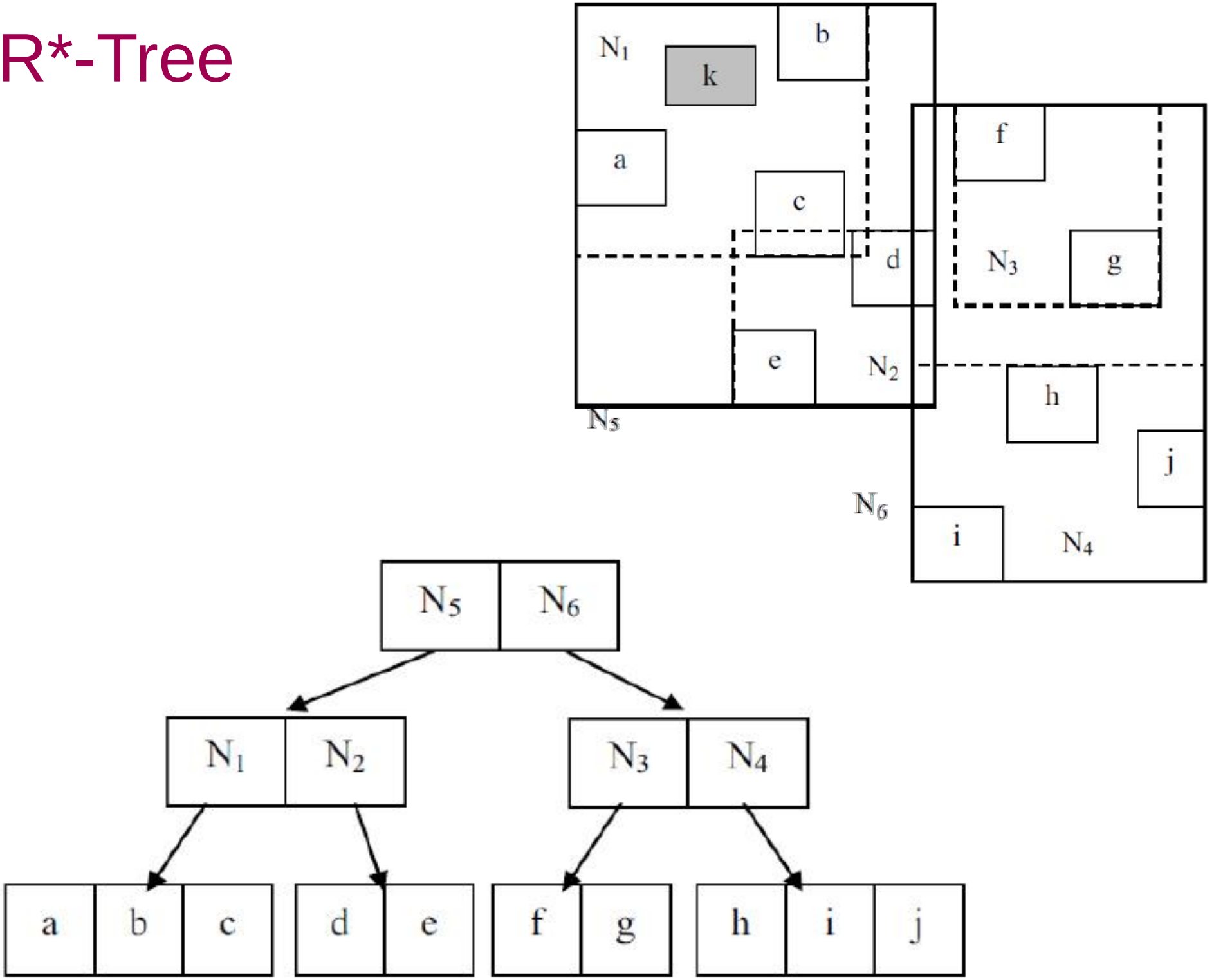
R*-Tree

- O R-Tree básico é muito ruim em minimizar a cobertura e a sobreposição devido à forma como os elementos são inseridos e excluídos
- R*-Tree é uma variação do R-Tree que melhora a versão básica
  - Minimizando a área de MBRs de nós
  - Minimizando a sobreposição
  - Minimizando o perímetro dos MBRs dos nós
  - Maximizando a utilização do armazenamento (elementos por nó)

R*-Tree

- Naturalmente, estes não podem ser todos ótimos ao mesmo tempo
  - É necessário encontrar um bom equilíbrio
- Isto é conseguido através de um método de inserção mais sofisticado
  - Em alguns casos, vários elementos são removidos e reinseridos

R*-Tree



Uso de Quad-Tree e R-Tree

- O banco de dados gráfico Neo4j pode usar indexação de árvore quádrupla
- O banco de dados espacial PostGIS pode usar R-tree como estrutura de indexação espacial
  - A árvore R é implementada usando uma estrutura chamada Generized Search Tree (GiST)
- O banco de dados SQLite inclui um índice de árvore R*
- Existe uma implementação Python para ambas as estruturas

Referências

ASSIS, G. T. de. **Introdução à Estrutura de Dados Espaciais & QuadTree**. [S. l.]: Universidade Federal de Ouro Preto, UFOP, 2018. http://www.decom.ufop.br/guilherme/BCC203/geral/ed2_introducao-estruturas-dados-espaciais_victor.pdf.

CONCI, A. **Solid modeling em C. G.** [S. l.]: Universidade Federal Fluminense, UFF, 2014. http://profs.ic.uff.br/~aconci/Solidos.pdf.

MACHUSAK, E. **The Notorious PM Quadtree**. [S. l.: s. n.], 2003. http://www.cs.umd.edu/~meesh/420/ContentBook/FormalNotes/PMQuadtree/pm_quadtree_local.pdf.

NIKANDER, J. **Data structures and algorithms for geometric models 2**. [S. l.]: Aalto University, 2019. https://mycourses.aalto.fi/pluginfile.php/926315/mod_folder/content/0/data%20structures%20and%20algorithms%202.pdf?forcedownload=1.

PEI, J. **R-Tree**. [S. l.]: Simon Fraser University, SFU, 2008. https://www2.cs.sfu.ca/CourseCentral/454/jpei/slides/R-Tree.pdf.