Estrutura de dados espaciais e QuadTree

Carlos Eduardo Romaniello - 19.1.4001
Carlos Gabriel Freitas - 19.1.4009
Daniel Monteiro Valério - 19.1.4035
Fábio Henrique Alves - 19.1.4128
Gabriel Mace Ferreira - 19.1.4013
Marcus Vinícius Fernandes - 19.1.4046
Vinícius Gabriel Verona - 19.1.4005

Tópicos

01 Estrutura de dados Espaciais

02 QuadTree

03 Execução

04 Bibliografia

01
ESTRUTURA DE DADOS ESPACIAIS

Introdução e explicação sobre as estruturas de dados espaciais

Dados escalares

Como é o caso em muitos termos em computação, a origem do termo está relacionada a propriedades físicas. Uma grandeza escalar é definida quando precisamos de um valor numérico associado a uma unidade de medida para caracterizar uma grandeza física.

Em suma, um dado escalar é um valor numérico único e simples, como por exemplo os valores:

- 1
- ⅔
- 3.14

Geralmente estes dados se encontram em forma de um inteiro, ponto fixo ou flutuante.

Os dados escalares fazem oposição a uma matriz, estrutura, objeto, vetor, que contém mais de um único valor numérico, sendo algo pouco mais complexo e elaborado.

“Without geography you're nowhere.”

— Jimmy Buffett

Dados espaciais

Dados Espaciais, também referido como dados geoespaciais, é a informação que identifica a localização geográfica de objetos físicos na Terra. São dados que podem ser mapeados, pois são armazenados como coordenadas e topologia.

Em suma, dados espaciais consistem em objetos espaciais compostos de pontos, linhas, regiões, retângulos, superfícies, volumes e até mesmo dados de dimensão superior que inclui o tempo.

Ao trabalhar com estes dados, também é desejável anexar informações de atributos não espaciais, como alturas de elevação, nomes de cidades e etc, visando tornar a informação mais completa.

Eles possuem uma forma específica para serem armazenados, visando a otimização em consultas e manipulações, trataremos destas especificidades a seguir informando detalhes referentes ao banco de dados, como também os casos de usos para estes, revelando onde encontramos exemplos em nosso dia a dia.

Objeto Geográfico

Três características principais descrevem um objeto geográfico:

- Atributos não espaciais (o que): descrevem qualitativa ou quantitativamente uma entidade geográfica. Estes dados podem ser tratados por bancos de dados não espaciais;

- Atributo espacial (onde): se refere à localização e à representação do objeto geográfico, considerando sua geometria e sistema de coordenadas. Este aspecto exige um tipo de dado específico não disponível em SGBDs (Sistemas de Gerenciamentos de Bancos de Dados) convencionais (não espaciais);

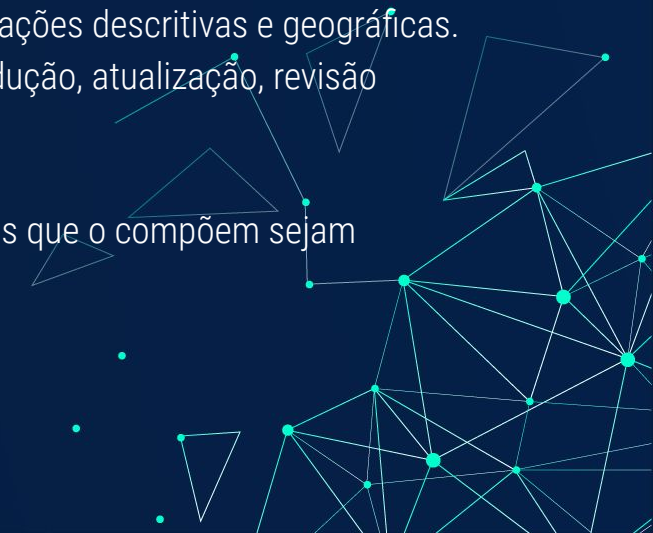
- Relacionamentos espaciais (como): relacionamentos de vizinhança (ex:. topologia, distância). Necessitam operações especiais que não são disponíveis em SGBDs convencionais.

Sistema de informação geográfica

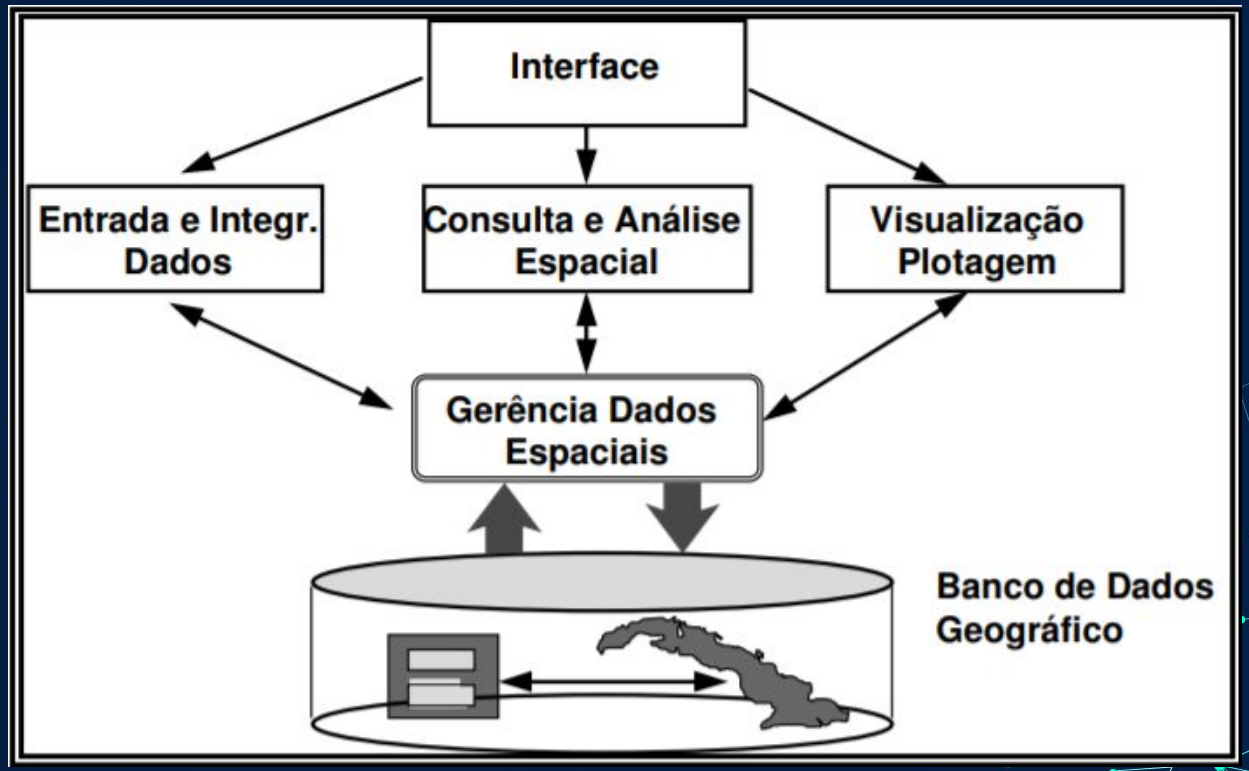
O termo SIG no campo de geoprocessamento é definido por Fitz (2008) como um sistema constituído por um conjunto de programas computacionais. Sua integração é feita por dados, equipamentos e pessoas. Tem o objetivo de coletar, armazenar, recuperar, manipular, visualizar e analisar dados espaciais relacionados com um sistema de coordenadas.

Sendo assim, consideramos o principal objetivo do SIG compatibilizar informações descritivas e geográficas. Isso permite produção de mapas com rapidez, facilidade, baixo custo de produção, atualização, revisão automática e análise quantitativa destes dados.

E por fim a compatibilidade entre os sistema exige ao menos os componentes que o compõem sejam próximos.



Arquitetura de SIG



Representação de Dados Espaciais

Os dados espaciais podem ser representados por meio de dados vetoriais e matriciais.

- Dados vetoriais:
  Comumente utilizados em mapas de tipo traço (ou linha), com feições geográficas associadas a pontos, linhas e polígonos.

- Dados matriciais:
  No formato matricial, o modelo ou imagem tem uma estrutura de células de grade. A cada célula da grade uma identidade de feição (característica do terreno) única é atribuída, normalmente um número (Ex.: montante de chuva; código numérico de uma categoria de uso do solo) ou um rótulo textual (um nome ou um código de letras).

Ponto

Usualmente o ponto é definido pelas coordenadas (x, y), e em certas situações, como mapeamento 3D, a coordenada z também pode ser utilizada.

O ponto é utilizado para representar a localização de um fenômeno geográfico ou para representar um aspecto do mapa que é muito pequeno para ser mostrado com as demais estruturas (Polígono e linha).

- Exemplo de código:

```
{
        “type”: “Point”,
        “coordinates”: [
           -87  .653274,
           41.936172
        ]
}
```

Linha

Definida por meio de dois pontos (dois pares de coordenadas (x, y)). Usada para representar feições geográficas que em teoria não possuem uma espessura, logo não podem ser representadas por uma área (polígono).

Vale ressaltar que árvores binárias e n-árias, por tratarem de apenas uma dimensão, podem ser utilizadas para representar um vetor.

- Exemplo de código:

```
{
        “type”: “LineString”,
        “coordinates”: [
          [102.0, 0.0],
          [103.0, 1.0],
          [104.0, 0.0],
          [105.0, 1;0]
        ]
}
```

Polígono

Definido por uma série de coordenadas (x, y) que formam segmentos de linhas, consequentemente fechando uma área definida.

- Exemplo de código:

```
{
        “type”: “LineString”,
        “coordinates”:[
          [100.0, 0.0],
          [101.0, 1.0],
          [101.0, 1.0],
          [100.0, 1;0],
          [100.0, 0.0]
        ]
}
```

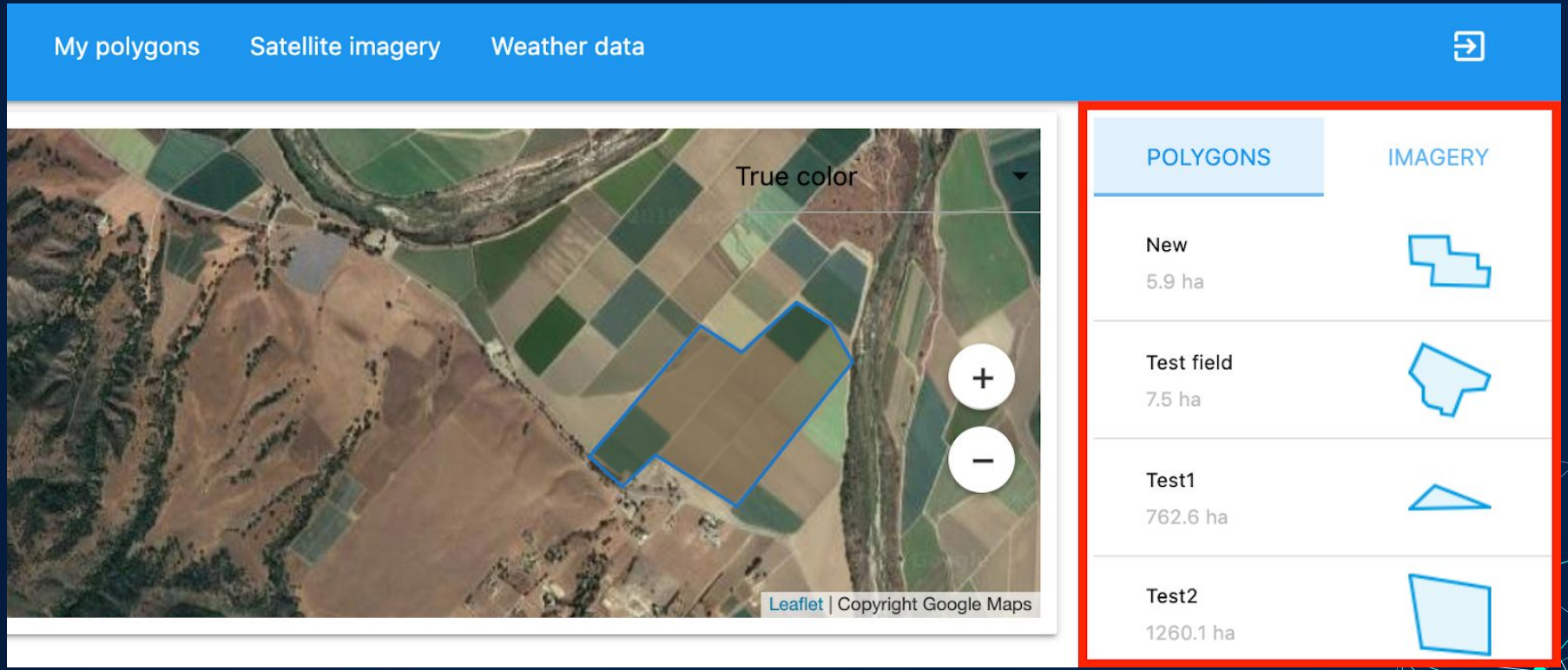


Imagem 1 - Divisão de terreno usando polígonos

Imagem 2 - Estruturas vetoriais compondo um mapa de linhas.

Raster (matriciais)

Formados por grades e células (*pixels*), que contém uma área do terreno, é importante ressaltar que a área contida depende da resolução espacial.

As imagens são armazenadas em linhas e colunas numa *grid* com um valor numérico digital para cada célula.

A célula é a unidade mínima de mapeamento, o que significa que é o menor tamanho com que qualquer feição da paisagem pode ser representada e mostrada. Todas as feições na área de uma célula são reduzidas a uma simples identificação de célula. Isso significa que todos os objetos presentes na região coberta por uma célula é acumulada e combinada em uma única identificação, ou seja, é uma generalização da paisagem e de suas feições.

Um *raster* é criado a partir da introdução de um *raster datasheet*, que define as características da matriz, ou seja tamanho dos *pixels*, número de linhas e colunas e etc, em uma imagem escolhida pelo usuário.

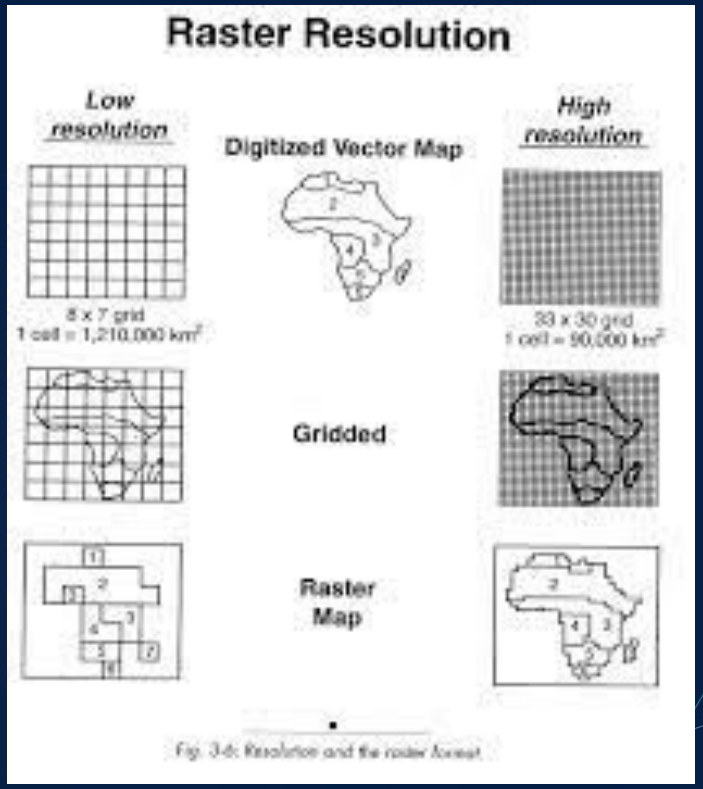


Imagem 3 - Demonstração de um *raster* com diferentes resoluções

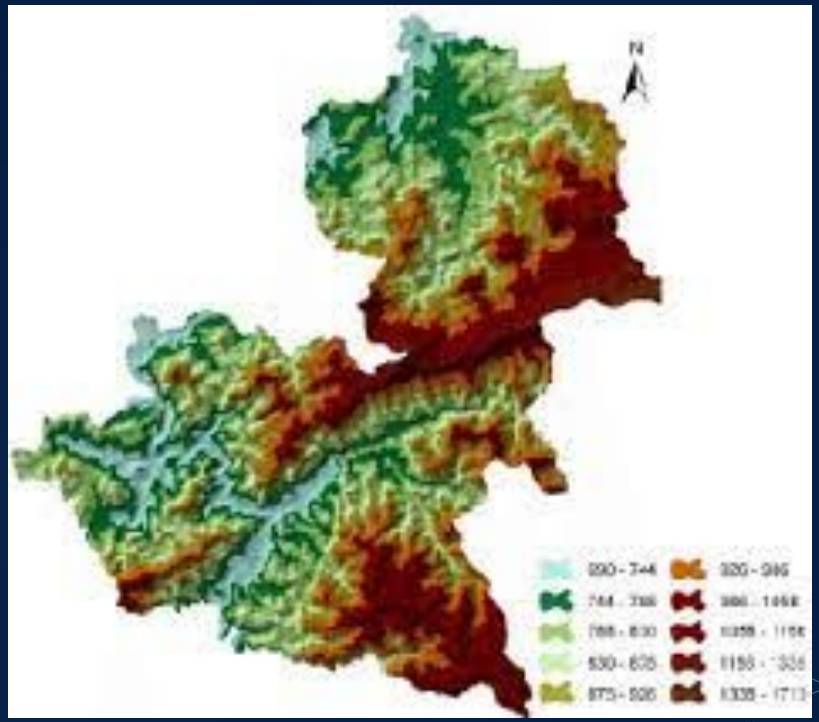


Imagem 4 - Estruturas matriciais utilizadas na composição de um mapa de altitude

02

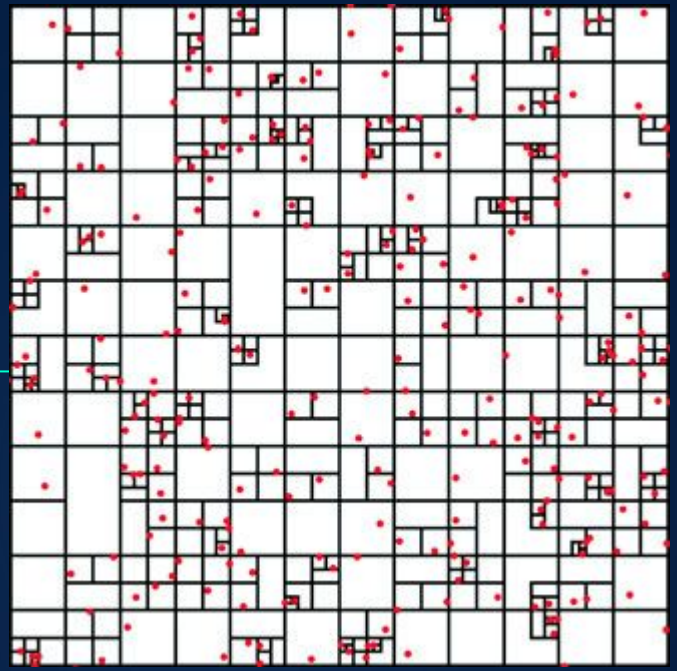
QuadTree

Introdução a estrutura de dado espacial QuadTree

QuadTree

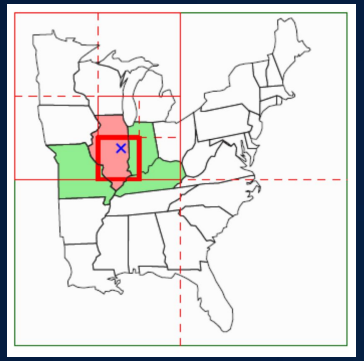
O que é?

A QuadTree é uma estrutura de dados do tipo árvore que armazena tipo de dados espaciais.



QuadTree

Pra que serve?

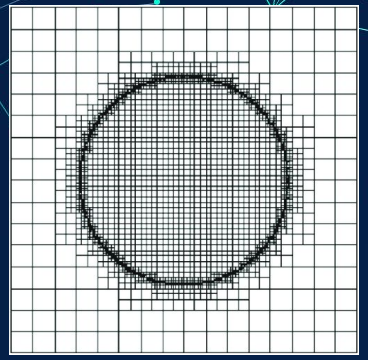


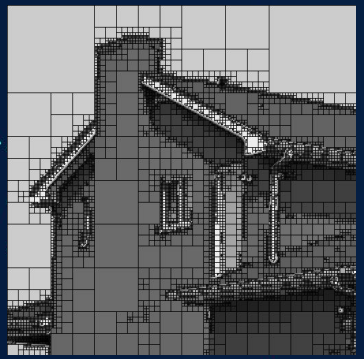
Representação de espaço real

Representação de um espaço real bidimensional demarcando pontos de interesse.

Mesh Generation

Subdivisão contínua de um espaço geométrico.





Processamento de imagens

Processo onde os dados de entrada e saída são imagens.

Detecção de colisão

Recurso presente nos jogos para tratamento de colisão entre objetos.

QuadTree

Point QuadTree

Existem vários tipos de QuadTree que se diferenciam no tipo de dado que armazenam

Region QuadTree

Edge QuadTree

QuadTree

Apesar de existirem diversos tipos de QuadTree, existem alguns padrões que devem ser seguidos

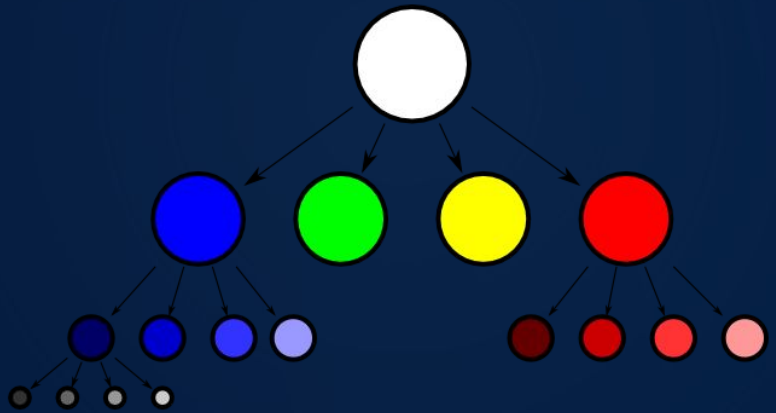
1º Decompor o espaço em células adaptáveis

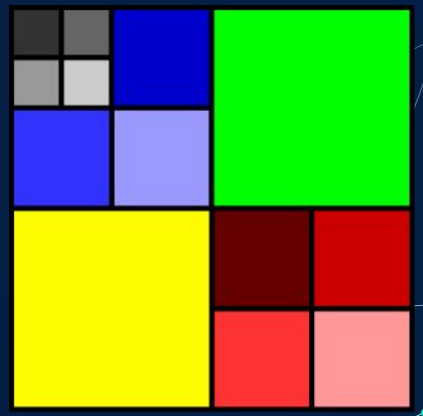
2º Cada nó tem uma capacidade máxima de dados

3º A ordem da árvore segue a decomposição espacial

QuadTree
Como funciona

- Dado um espaço, ele é subdividido recursivamente em quadrantes (NE, NO, SE, SO)
- Cada nó da árvore armazena os dados espaciais do seu respectivo quadrante

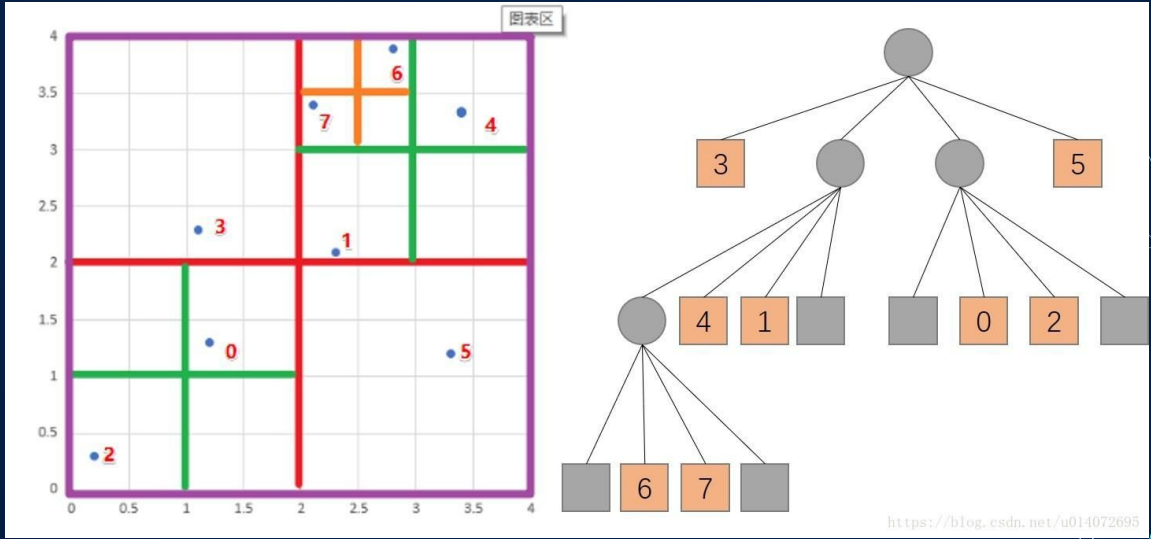




QuadTree

Como funciona

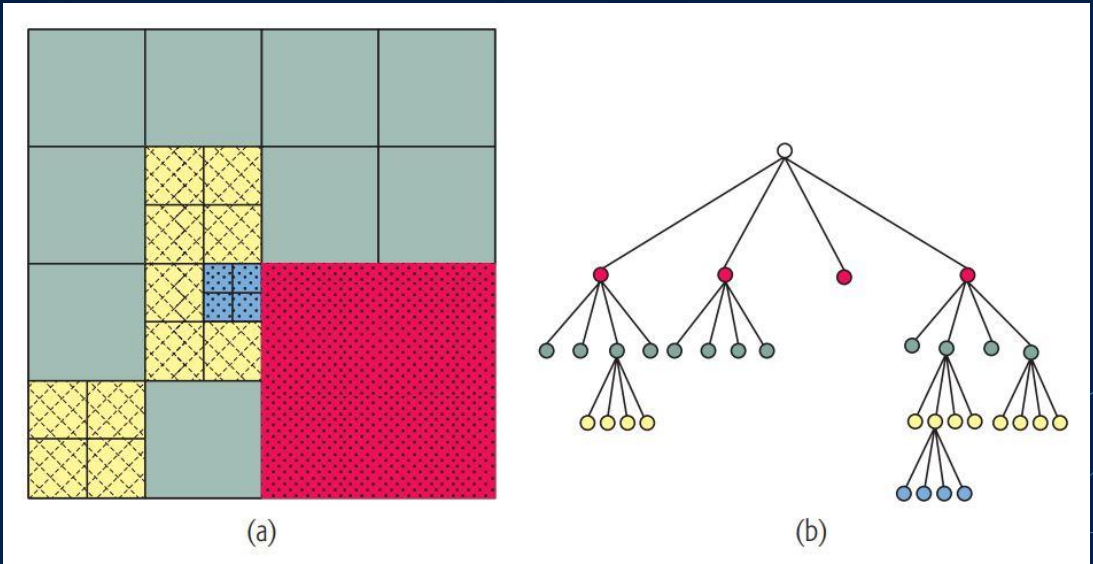
- Cada nó pode armazenar um número limitado de dados
- Caso esse limite seja extrapolado ocorrerá uma divisão gerando mais 4 sub quadrantes
- Os dados ou pontos de interesse são armazenados nos nós folhas



Region QuadTree

- Útil para representar uma parte do espaço bidimensional
- Cada nó obrigatoriamente tem que ter 4 nós filhos ou nenhum
- A altura depende da distribuição espacial dos pontos

Region QuadTree



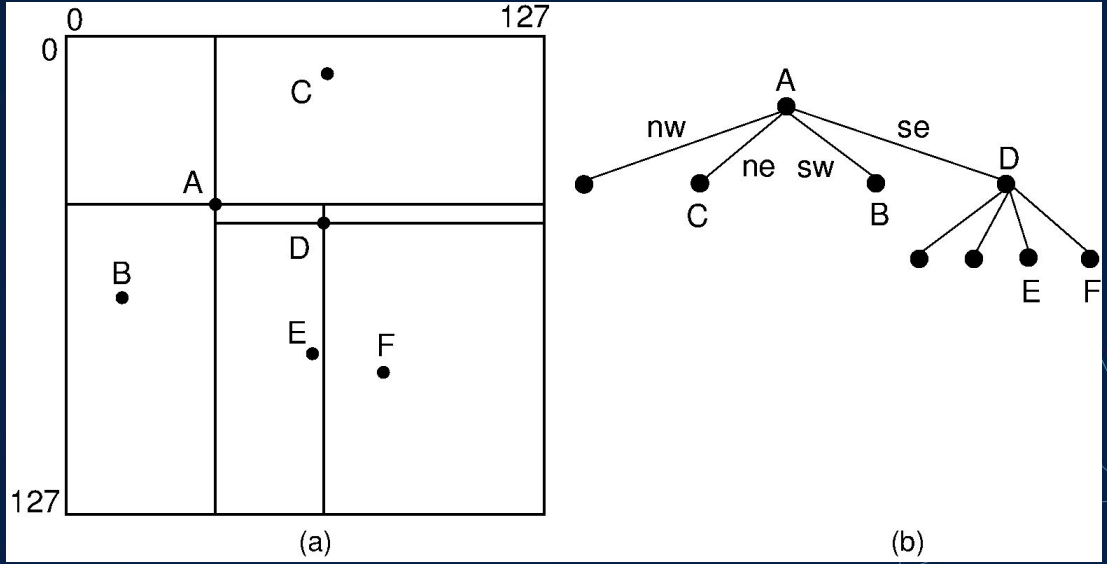
Em ‘a’ temos a representação gráfica de uma QuadTree de região.

Em ‘b’, temos uma QuadTree propriamente dita

Point QuadTree

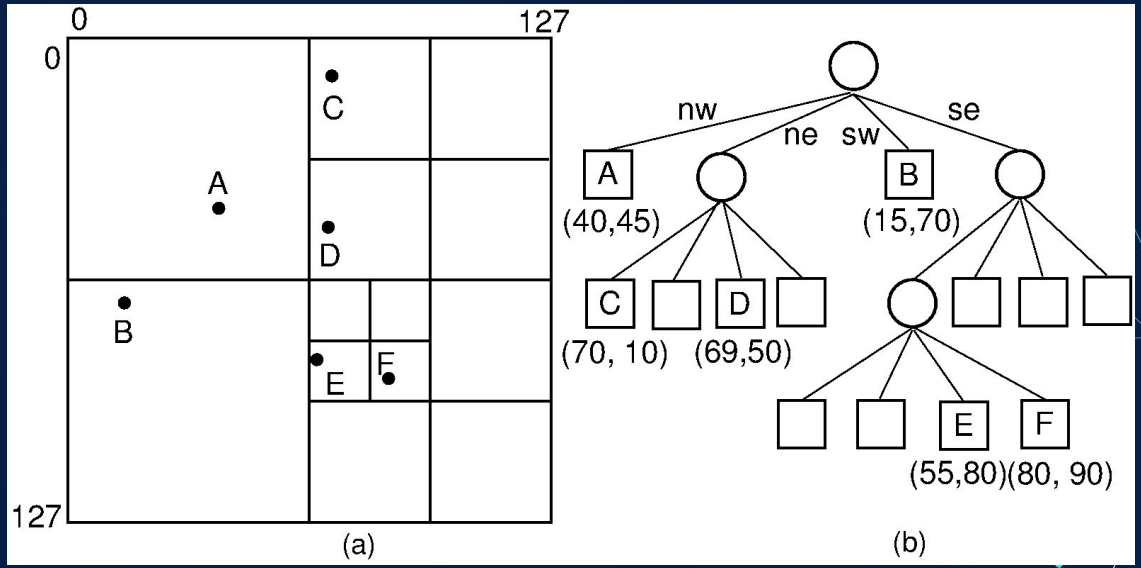
- É uma generalização da multidimensional de uma Árvore Binária
- Cada ponto de dados é representado por um nó da Árvore Quadrante
- Cada nó é um tipo de dado com, no mínimo, seis campos

Point QuadTree



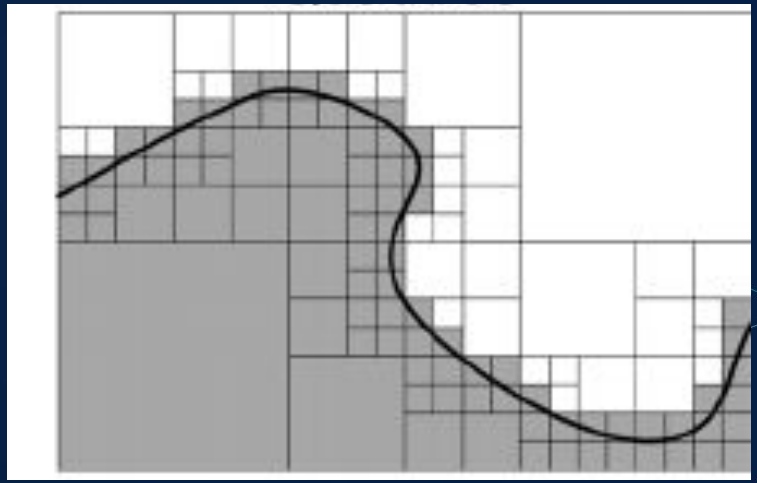
Point Region QuadTree

- Se assemelha muito a uma Region QuadTree
- Armazena uma lista de pontos que existem dentro da célula de um nó folha



Edge QuadTree

- É usada para armazenar linhas em vez de pontos
- Perto de cada vértice, vai se dividindo até atingir seu nível máximo de decomposição



QuadTree

QuadTrees podem ser usadas para diversas aplicações, como por exemplo:

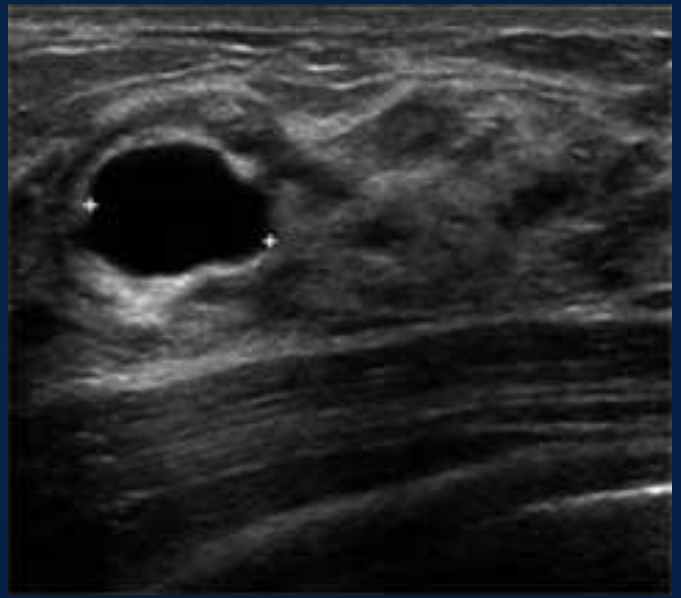
- Compressão de Imagens;
- Na medicina, com ecografias, ajudando a identificar tumores pela cor na imagem;
- Em jogos, com detecção de colisão em 2D.

QuadTree

Uso para compressão de imagens.

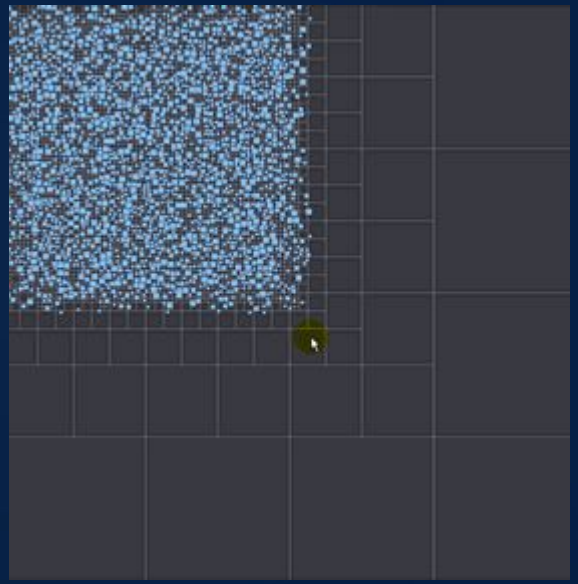
QuadTree

Uso na ecografia para detecção de tumores.



QuadTree

Sistema de detecção de colisão.



Vantagens e Desvantagens em Usar uma QuadTree

Vantagens:

- O gráfico pode ser armazenado de uma forma compacta
- Facilidade para rotacionar imagens
- Fornece uma estrutura de árvore num formato muito mais compacto e robusto

Desvantagens:

- A árvore resultante será muito maior se tivermos áreas com várias cores diferentes
- Um considerável consumo de processamento quando se trabalha com imagens complexas
- Não há balanceamento

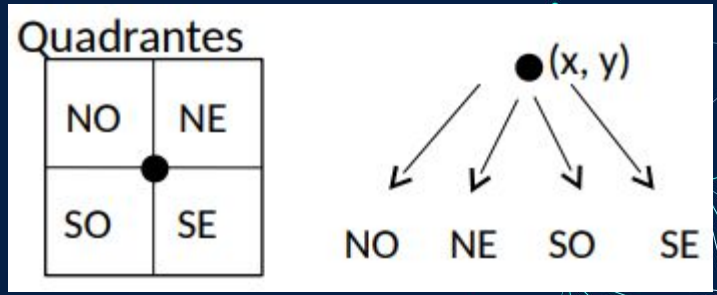
Estruturas em uma Point QuadTree

```
typedef struct TipoRegistro {
   //Componentes de um Registro
}TipoRegistro;

typedef enum {Noroeste, Nordeste, Sudoeste, Sudeste} Quadrante;

typedef struct TipoNo* TipoApontador;

typedef struct TipoNo{
   int x;
   int y;
   TipoRegistro r;
   TipoApontador quadrantes[4];
}TipoNo;
```



Localizar quadrante em uma Point QuadTree

- Complexidade: O(Altura do ponto) = O(1)

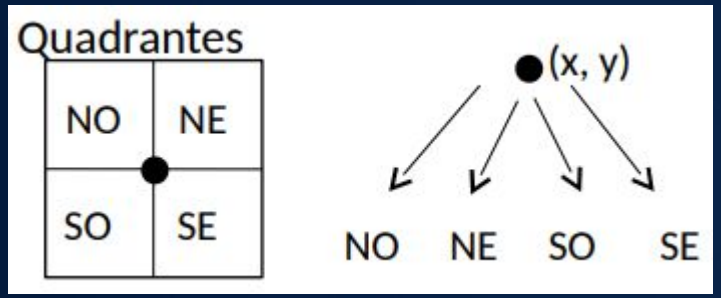
```
Quadrante RetornaQuadrante(TipoNo Ponto, TipoApontador PontoAtual){
   if(Ponto.x <= PontoAtual->x && Ponto.y >= PontoAtual->y)
      return Noroeste;
   else if(Ponto.x > PontoAtual->x && Ponto.y >= PontoAtual->y)
      return Nordeste;
   else if(Ponto.x <= PontoAtual->x && Ponto.y < PontoAtual->y)
      return Sudoeste;
   else if(Ponto.x > PontoAtual->x && Ponto.y < PontoAtual->y)
      return Sudeste;
}
```

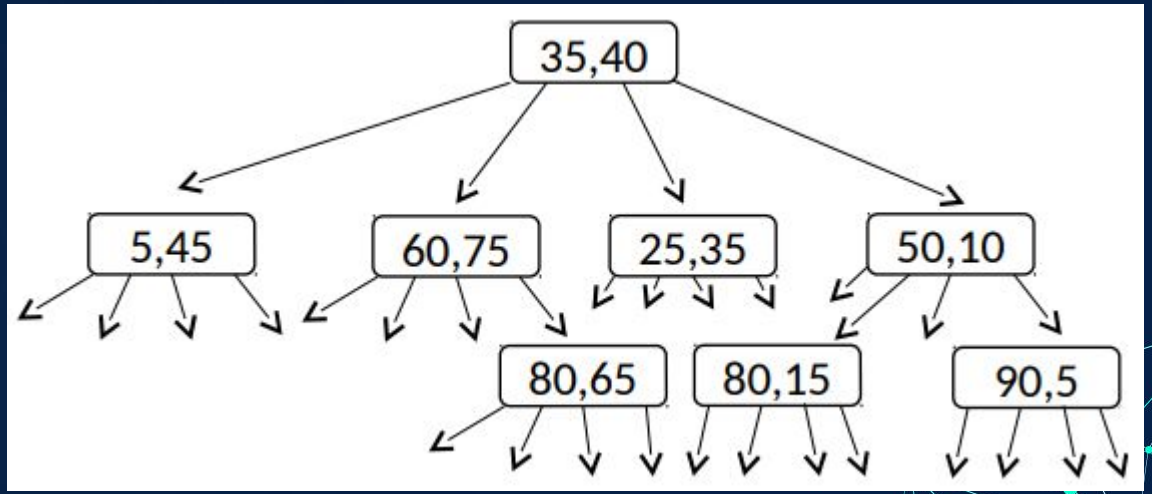
Localizar um ponto em uma Point QuadTree

- Complexidade: O($\log_4 n$)

```
TipoApontador ProcuraPonto(TipoNo Ponto, TipoApontador A){
   Quadrante q;
   if(Ponto.x == A->x && Ponto.y == A->y)
      return A; //O ponto foi encontrado na QuadTree A
   else if(A == NULL)
      return NULL; //O ponto não existe na QuadTree A
   else{
         q = RetornaQuadrante(Ponto, A); //Localizando o quadrante que possa conter o ponto
         return ProcuraPonto(Ponto, A->quadrantes[q]);
    }
}
```

Localizar um ponto em uma Point QuadTree



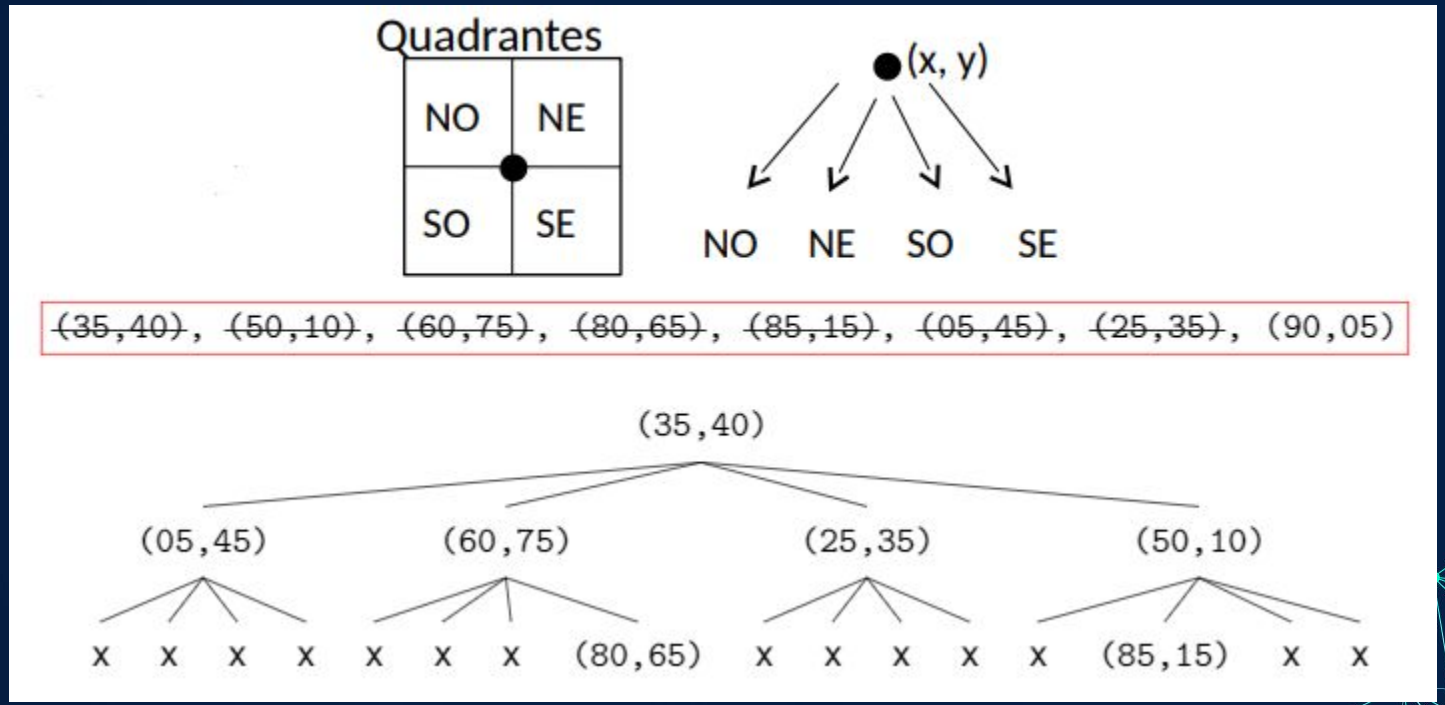


Inserção em uma Point QuadTree

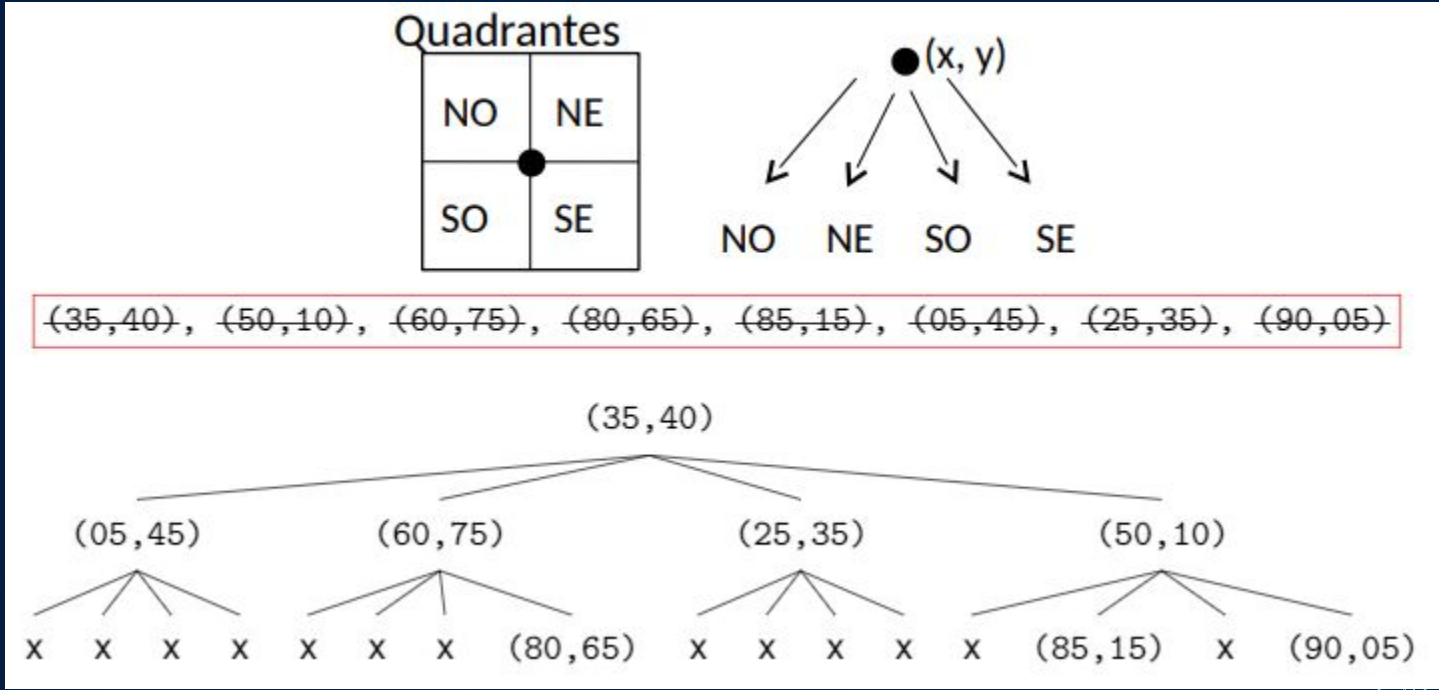
- Complexidade: O($\log_4 n$)

```
void InserePonto(TipoApontador Ponto, TipoApontador* A){
   if(*A == NULL) //A QuadTree está vazia
      *A = Ponto;
   TipoApontador Pai, Aux = *A;
   Quadrante q;
   while( (Aux != NULL) && (Ponto->x != Aux->x) && (Ponto->y != Aux->y) ){ //A QuadTree não terá dois pontos iguais
      Pai = Aux;
      q = RetornaQuadrante(Ponto, Aux);
      Aux = Aux->quadrantes[q];
   }
   if(Aux == NULL) //Inserindo o ponto
      Pai->quadrantes[q] = Ponto;
}
```

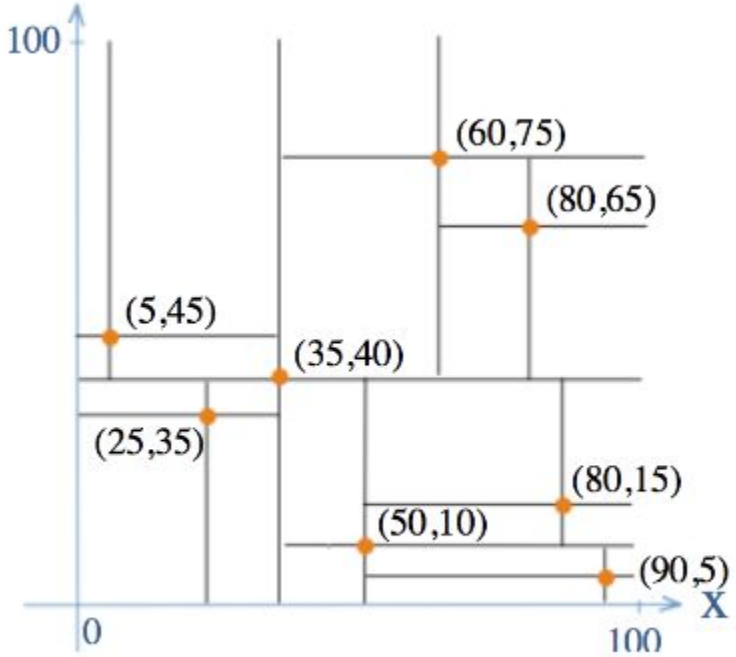
Inserção em uma Point QuadTree



Inserção em uma Point QuadTree



Possível representação de uma Point QuadTree



(35,40), (50,10), (60,75), (80,65), (85,15), (05,45), (25,35), (90,05)

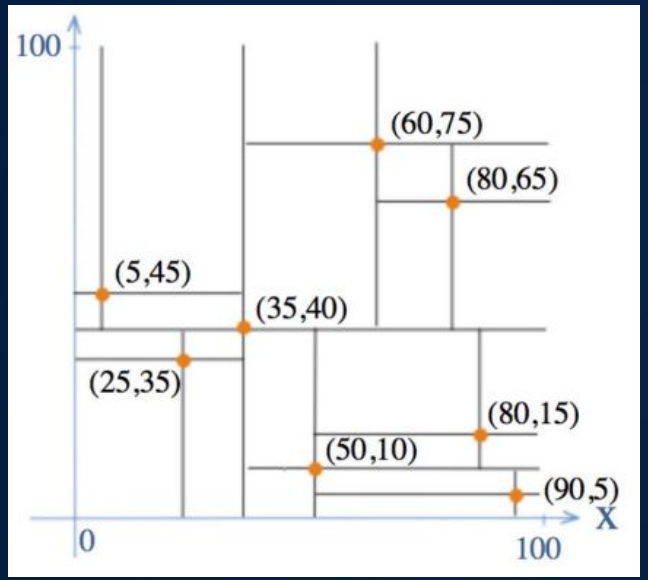
Pesquisa em um intervalo em uma Point QuadTree

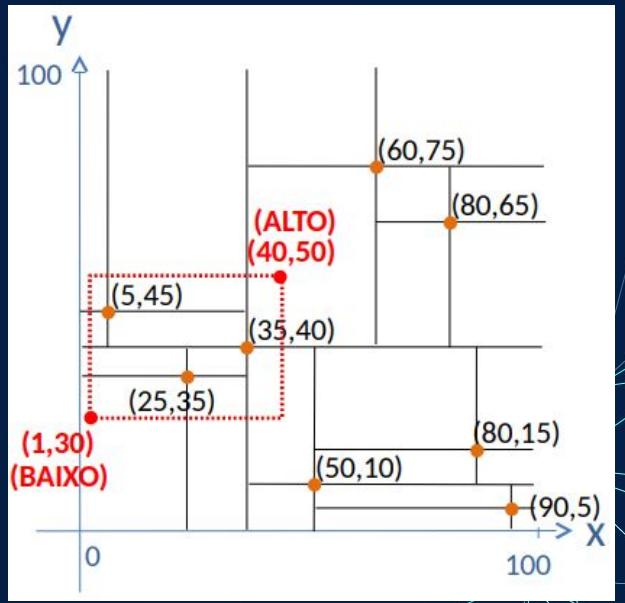
- Complexidade: O($\log_4 n$)

```
void PesquisaIntervalo(TipoNo Ponto, TipoNo PontoBaixo, TipoNo PontoAlto, TipoNo* VetorPontos, TipoApontador A){
   if(PontoBaixo.x <= x && x <= PontoAlto.x && PontoBaixo.y <=y && y <= PontoAlto.y)
      AdicionaPonto(Ponto, VetorPontos);
   if(PontoBaixo.x <= x && PontoBaixo.y <= y)
      return PesquisaIntervalo(Ponto, PontoBaixo, PontoAlto, VetorPontos, A->quadrantes[Sudoeste]);
   if(PontoBaixo.x <=x && y <= PontoAlto.y)
      return PesquisaIntervalo(Ponto, PontoBaixo, PontoAlto, VetorPontos, A->quadrantes[Noroeste]);
   if(x <= PontoAlto.x && PontoBaixo.y <= y)
      return PesquisaIntervalo(Ponto, PontoBaixo, PontoAlto, VetorPontos, A->quadrantes[Sudeste]);
   if(x <= PontoAlto.x && y <= PontoAlto.y)
      return PesquisaIntervalo(Ponto, PontoBaixo, PontoAlto, VetorPontos, A->quadrantes[Nordeste]);
  return;
}
```

Pesquisa em um intervalo em uma Point QuadTree

- Pesquisa de pontos no intervalo entre os pontos (1,30) e (40,50) na QuadTree mostrada anteriormente.





03

Execução

Um exemplo de aplicação da QuadTree

04

Bibliografia

Referências utilizadas

Leituras

http://www.cs.umd.edu/~hjs/pubs/kim.pdf
http://orion.lcg.ufrj.br/gc/download/ede.pdf
https://www.geeksforgeeks.org/quad-tree/
https://en.wikipedia.org/wiki/Quadtree
http://www.decom.ufop.br/guilherme/BCC203/geral/ed2_introducao-estruturas-dados-espaciais_ricardo.pdf
http://www.decom.ufop.br/guilherme/BCC203/geral/ed2_introducao-estruturas-dados-espaciais_josiane.pdf
https://docplayer.com.br/79048588-Algoritmos-e-estrutura-de-dados-ii-estrutura-de-dados-espaciais.html
https://dev.socrata.com/docs/datatypes/point.html
https://en.wikipedia.org/wiki/Quadtree#Some_common_uses_of_quadtrees
https://lume.ufrgs.br/handle/10183/23901
https://web.tecgraf.puc-rio.br/ftp_pub/lfm/MirandaGeomComp1999.pdf https://en.wikipedia.org/wiki/Spatial_database#Spatial_index
https://www.ufjf.br/jairo_souza/files/2009/12/5-Indexa%c3%a7%c3%a3o-Point-QuadTree.pdf
https://stackoverflow.com/questions/41946007/efficient-and-well-explained-implementation-of-a-quadtree-for-2d-collision-det
https://www.saudebemestar.pt/pt/exame/imagiologia/ecografia/
https://docs.qgis.org/2.8/en/docs/gentle_gis_introduction/raster_data.html
https://desktop.arcgis.com/en/arcmap/10.3/tools/data-management-toolbox/create-raster-dataset.htm
https://www.eosconsultores.com.br/o-que-e-um-banco-de-dados-geograficos/
https://slideplayer.com.br/slide/359212/
https://geojson.io/

Vídeos

https://www.youtube.com/watch?v=J6Q4mqj8PwY&list=PLRa695dKCXe0YJsKEyw2jyho_zW94p1U1 (Playlist)
What is Spatial Data - An Introduction to Spatial Data and its Applications
Coding Challenge #98.1: Quadtree - Part 1
Coding Challenge #98.2: Quadtree - Part 2
Coding Challenge #98.3: Quadtree Collisions - Part 3

Dúvidas


slidesgo