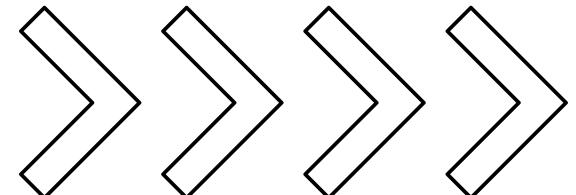


QUADTREE

ESTRUTURA DE DADOS

ESPACIAL



Aluno: Pedro Hugo Passos da Silva

Carlos

Professor: George Lima

Universidade Federal da Bahia

Sumário

03 O que é Quadtree?

04 Estrutura e Operações

05 Vantagens e Desvantagens

06 Aplicações

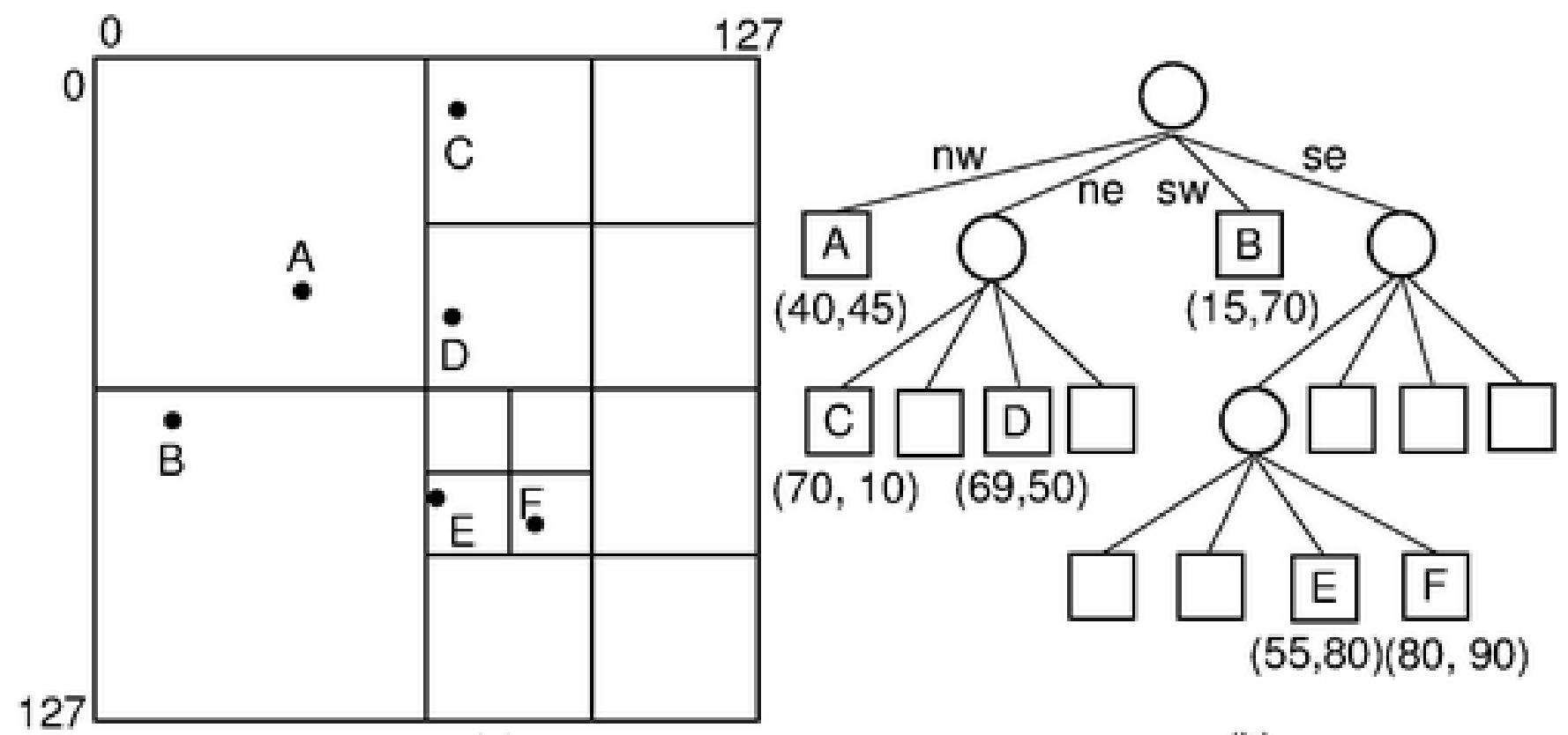
07 Implementação do Projeto

11 Conclusão

12 Referências Bibliográficas

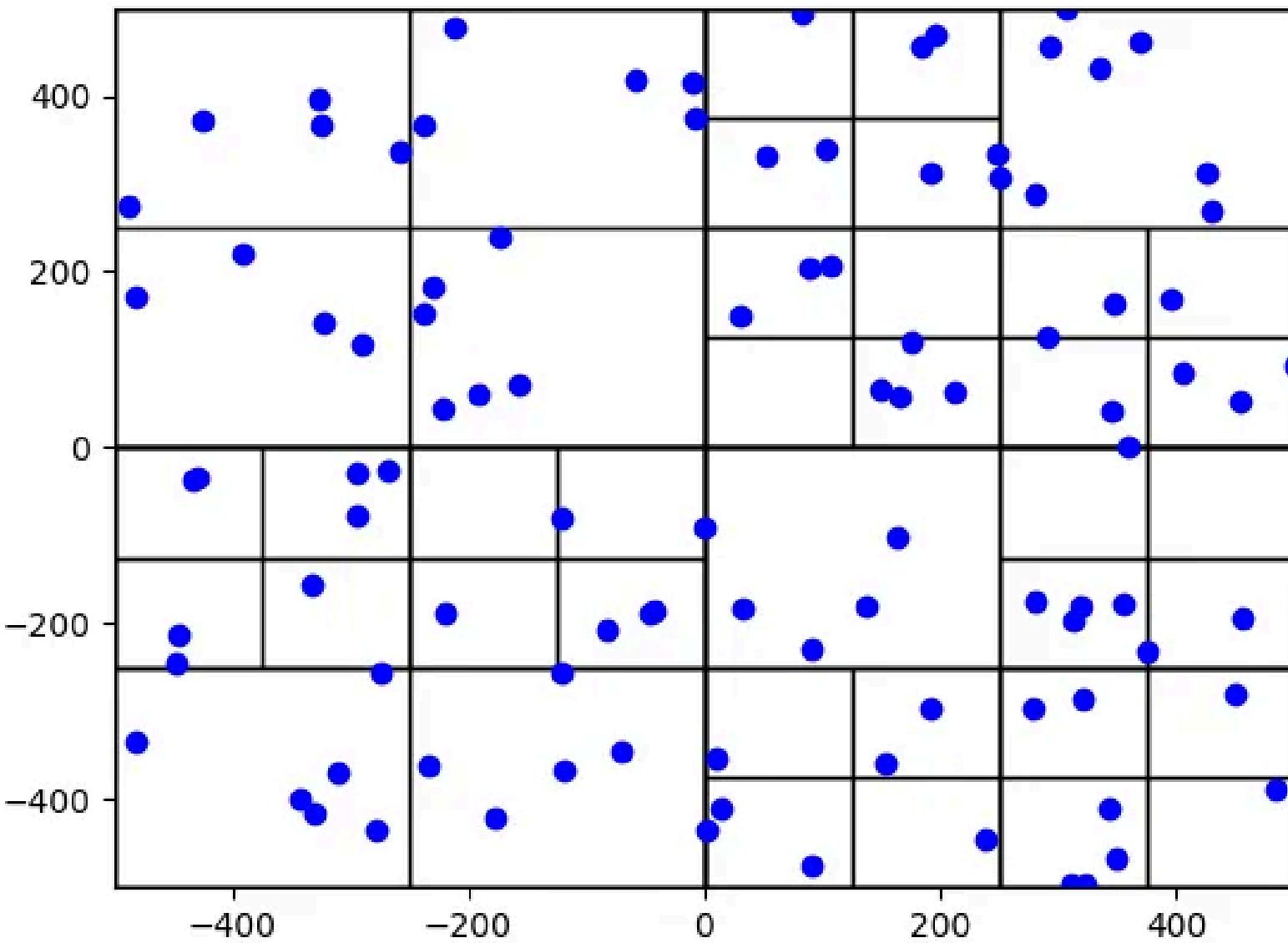
O que é Quadtree?

- Estrutura em árvore que partitiona recursivamente o espaço 2D em quatro quadrantes
- Cada nó pode ter até quatro filhos (noroeste, nordeste, sudoeste, sudeste)
- A subdivisão é adaptativa:
 - Áreas densas recebem subdivisões mais refinadas
 - Áreas esparsas permanecem com blocos maiores



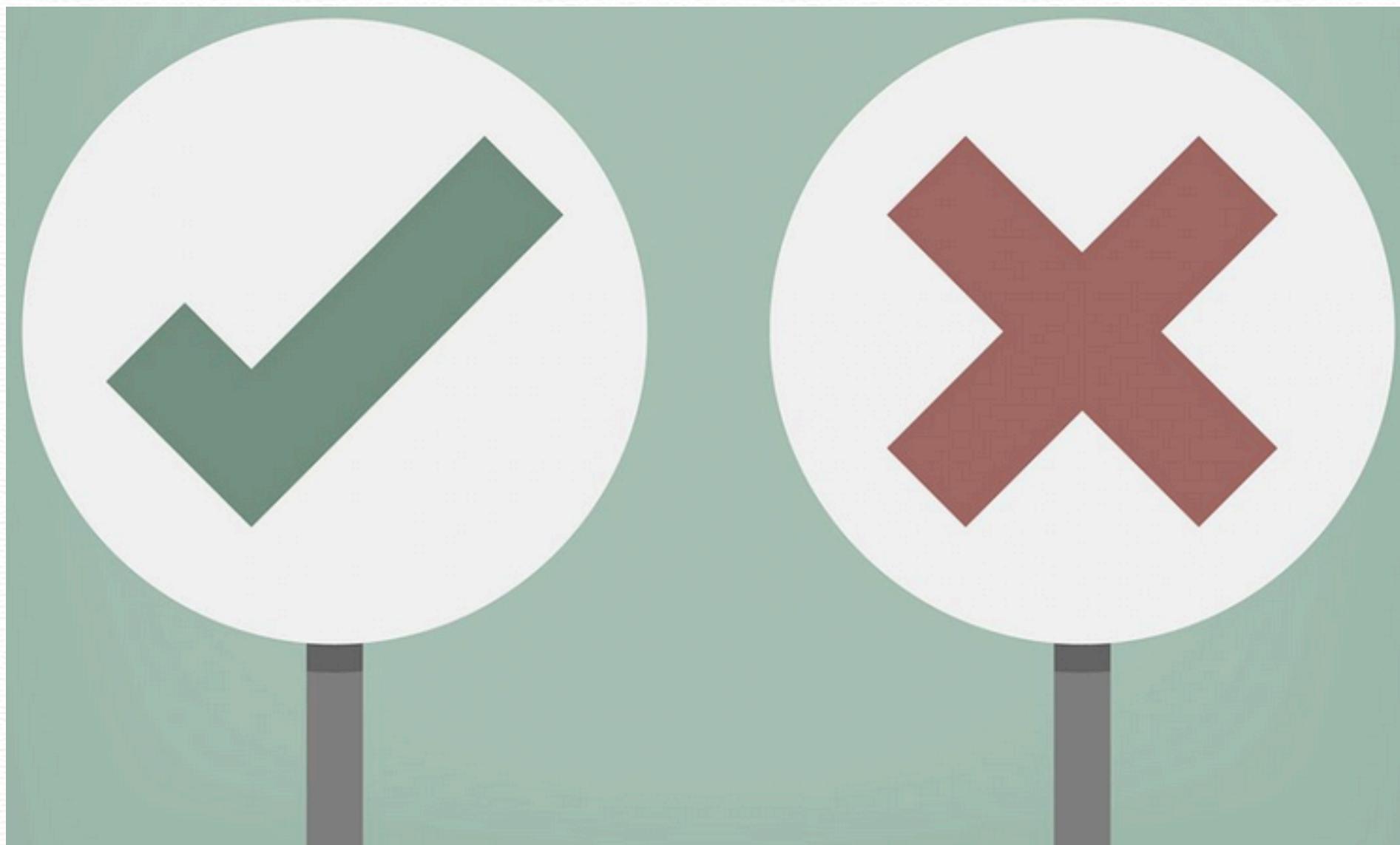
Estrutura e Operações

- Divisão Espacial:
 - Nó raiz cobre toda a região
 - Subdivisão ocorre quando um nó excede a capacidade (ex.: mais de 4 pontos)
- Operações Principais:
 - Inserção: Percorre a árvore para encontrar o quadrante adequado ($O(\log N)$)
 - Busca: Localiza um ponto comparando coordenadas ($O(\log N)$)
 - Remoção: Exclui o ponto e, se necessário, colapsa nós vazios



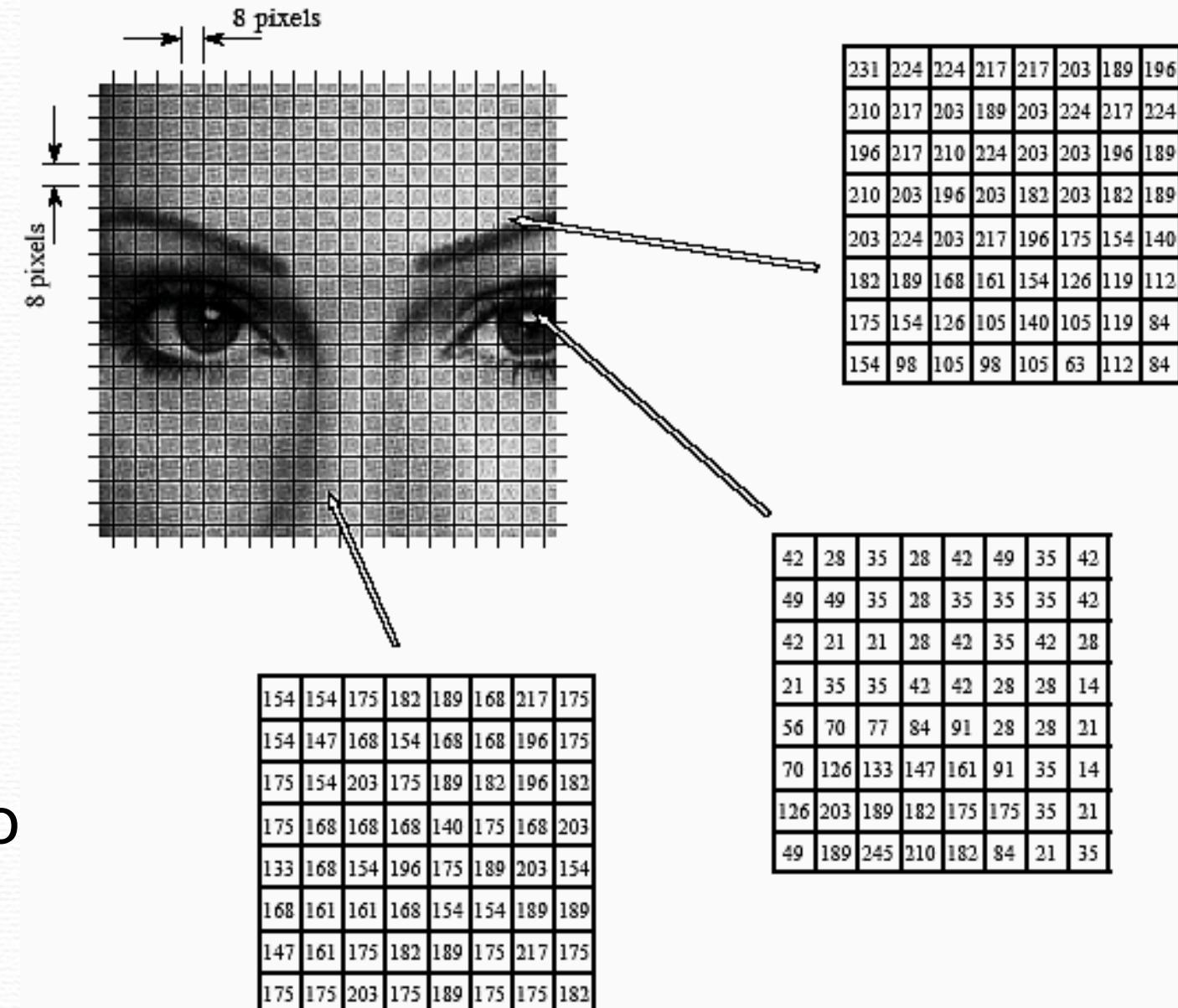
Vantagens e Desvantagens

- Vantagens:
 - Adaptação à distribuição dos dados
 - Redução do espaço de busca em consultas e detecção de colisões
 - Possibilidade de compressão e visualização hierárquica
- Desvantagens:
 - Em dados irregulares, pode ocorrer excesso de subdivisões
 - Complexidade na implementação para dados dinâmicos
 - Risco de artefatos em compressão se a granularidade for inadequada



Aplicações

- Compressão de Imagens:
 - Representa regiões homogêneas com menos informação
- Detecção de Colisões em Jogos:
 - Limita as verificações somente a entidades próximas
- Sistemas de Informação Geográfica (SIG):
 - Eficiência em consultas espaciais e armazenamento de dados
- Processamento Gráfico:
 - Organização e renderização em múltiplas resoluções



Implementação do Projeto

- Arquivos Principais:
 - main.c: Menu interativo para inserir, remover e buscar pontos; gera arquivo SVG
 - quadtree.c: Funções básicas da quadtree (criação, inserção, busca, remoção)
 - map_quadtree.c: Geração da visualização em SVG – eixos, subdivisões e pontos

The image shows a terminal window with three code editors open in separate tabs:

- main.c**: A C program that includes stdio.h, stdlib.h, stdbool.h, quadtree.h, and map_quadtree.h. It contains functions for inserting points into a quadtree and a main loop for interacting with the user.
- quadtree.c**: A C program that includes stdio.h, stdlib.h, and quadtree.h. It defines a Quad structure and functions for creating nodes, inserting points, and destroying trees.
- map_quadtree.c**: A C program that includes map_quadtree.h and quadtree.h. It handles the generation of an SVG visualization, including drawing axes and recursive subdivision.

Implementação do Projeto - Inserção

- Verificação de limites:
 - Se o nó estiver fora da área delimitada pelo Quadtree (q), a inserção é cancelada.
- Caso base – Nó folha:
 - Se q ainda é uma folha (não tem subdivisões):
 - Se q->n estiver vazio, o nó é armazenado diretamente.
 - Se já existir um nó na posição, ele é atualizado (evitando nós duplicados).
- Subdivisão da região:
 - Se já houver um nó armazenado e for necessário inserir outro:
 - O espaço é dividido em quatro quadrantes menores.
 - O nó existente e o novo nó são reinseridos na estrutura correta.
- Recursão para inserção no quadrante correto:
 - O código determina em qual dos quatro quadrantes o novo nó deve ser inserido.
 - Se o quadrante ainda não existir, ele é criado.
 - A inserção continua recursivamente no subquadrante adequado.

Implementação do Projeto - Busca

- Verificação de limites:
 - Se o ponto p estiver fora dos limites da Quadtree (q), retorna NULL (não encontrado).
- Caso base – Nό folha:
 - Se q for um nό folha, retorna o nό armazenado ($q->n$), pois a busca nό pode avançar mais.
- Determinação do quadrante:
 - O código calcula o ponto mόdio ($midX$, $midY$) da regiόn.
 - Compara $p.x$ e $p.y$ com os valores mόdios para determinar em qual dos quatro quadrantes continuar a busca.
- Busca recursiva no quadrante correto:
 - Se o quadrante correspondente existe, a funcōao chama `Quad_search` recursivamente nele.
 - Se o quadrante nό existir, retorna NULL (ponto nό encontrado).

Implementação do Projeto - Remoção

- Verificação inicial:
 - Se q for NULL ou o ponto p estiver fora dos limites, a função retorna imediatamente.
- Caso base – Nό folha:
 - Se q for um nό folha e contiver o ponto p, o nό é removido e liberado da memória.
- Determinação do quadrante:
 - O código calcula o ponto médio (midX, midY) da região.
 - Compara p.x e p.y com os valores médios para determinar em qual dos quatro quadrantes continuar a busca pela remoção.
- Remoção recursiva no quadrante correto:
 - Se o quadrante correspondente existe, a função chama Quad_remove recursivamente nele.
- Verificação de colapso da subárvore:
 - Se todos os quadrantes estiverem vazios, eles são desalocados, e a Quadtree volta a ser um nό folha.

Conclusão

- A quadtree é uma estrutura poderosa para dados espaciais
- Facilita operações como inserção, busca e detecção de colisões
- Adapta-se dinamicamente à densidade dos dados
- Possui aplicações diversas em compressão de imagens, jogos, SIG e gráficos
- Este projeto demonstra, de forma prática, a criação e visualização hierárquica de dados

2D

Referências

- Samet, H. (1984). The Quadtree and Related Hierarchical Data Structures. *Computing Surveys*, 16(2).
- Finkel, R. A., & Bentley, J. L. (1974). Quad trees: A data structure for retrieval on composite keys. *Acta Informatica*, 4(1), 1–9.
- Bentley, J. L. (1975). Multidimensional binary search trees used for associative searching. *Communications of the ACM*, 18(9), 509–517.