Universidade Federal da Bahia - UFBA
Instituto de Matemática - IM
Departamento de Ciência da Computação - DCC
Curso de Bacharelado em Ciência da Computação

MATA40 - Estrutura de Dados Período: 2011-1 Data: 15/06/2011
Prof. Antonio L. Apolinário Junior

Segundo Trabalho

Estruturas de dados baseadas em árvores são bastante aplicadas em área que trabalham com informações organizadas espacialmente e sobre as quais pretende-se aplicar algum tipo de estruturação hierárquica. Um exemplo é o uso de árvores aplicadas a algoritmos de compressão de dados. Algoritmos com a codificação de Huffman constroem uma estrutura de árvore binária para definir uma codificação de tamanho variável, baseada na freqüência dos símbolos em um texto. Dessa forma, códigos de menor tamanho são associados aos símbolos de maior freqüência, reduzindo assim o espaço de armazenamento do texto.

Um principio semelhante pode ser utilizado na compressão de imagens. Nesse caso, como uma imagem é uma estrutura bi-dimensional, o espaço deve ser dividido em 2 dimensões, horizontal e vertical, dando origem a uma árvore quaternária ou comumente chamada de ***quadtree***. A figura 1 mostra uma *quadtree* gerada a partir de uma imagem de 8x8 *pixels*.

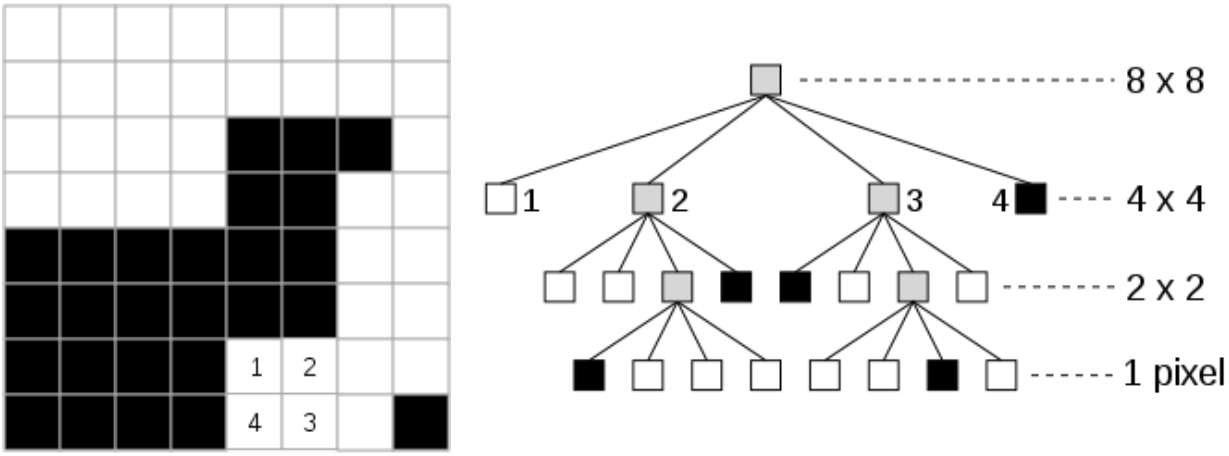

Figura 1 - Imagem e *quadtree*: à esquerda uma imagem de 8x8 *pixels* e à direita sua representação em uma estrutura de *quadtree*.

Como podemos ver na Figura 1, a estrutura da *quadtree* representa uma subdivisão recursiva da imagem em subimagens, de dimensões reduzidas a 1/4 da imagem original a cada novo nível da árvore. Portanto, a raiz da árvore representa a imagem original 8x8. No nível seguinte temos 4 subimagens, cada uma com dimensão 4x4, representando a subdivisão da imagem original em 4 quadrantes. A cada novo nível o processo de repete, até que a dimensão da imagem se reduza a um único *pixel* (folha da árvore).

O processo descrito acima dá origem a uma árvore completa, onde todas as folhas conterão a informação de um *pixel*. No entanto, analisando ainda a Figura 1, vemos que tão logo uma subimagem seja detectada como contendo uma única informação (todos os *pixels* pretos ou brancos) o processo de subdivisão pode ser abortado. Podemos ver claramente que é o caso dos filhos mais a esquerda e mais a direita do nó raiz, que correspondem, respectivamente, aos

quadrantes superior esquerdo e inferior esquerdo. É ai que a compressão ocorre: toda uma região que possui a mesma informação é “resumida” a um único nó da estrutura, evitando a repetição da informação dos *pixels* dessa região.

Vale notar que como temos associados a cada nó 4 filhos, há que se estabelecer um padrão para alocação dos filhos de um nó. Na Figura 1 podemos observar o padrão estabelecido pela numeração (1,2,3,4) na imagem 8x8. Ou seja, da esquerda para a direita, os filhos de um nó vão corresponder sempre aos quadrantes superior esquerdo (1), superior direito (2), inferior direito (3) e inferior esquerdo (4). Essa ordem deve ser mantida durante todo o processo para garantir a consistência da estrutura.

Note que no exemplo da Figura 1 a imagem é um ***bitmap***, ou seja, um mapa de *bits*, onde cada *pixel* possui apenas 2 tons de cores: preto ou branco. Em uma imagem mais real (ainda sem cor), temos uma variação de tons de cinza, tipicamente representada por um *byte* por pixel. Ou seja, a cada *pixel* da imagem temos a possibilidade de representar um valor de tom de cinza entre 0 (preto) e 255 (branco). Em uma situação como essa, a obtenção de um quadrante em que todos os *pixels* sejam exatamente iguais (como nos filhos (1) e (4) da raiz) é mínima. Nesse caso trabalhamos com a idéia de similaridade entre os *pixels* da região analisada. Caso em um quadrante todos os *pixels* sejam “parecidos”, ou seja, próximos de uma mesmo valor, podemos considerar que aquela quadrante pode ser “comprimido” por esse “valor próximo”.

Você provavelmente já deve estar imaginando que a forma mais fácil de saber se os *pixels* são “parecidos” é calcular o valor de tom de cinza médio da região (média aritmética dos *pixels*) e comparar o erro percentual na substituição de cada pixel pela media do quadrante. Se esse erro estiver abaixo de um certo limiar (5% por exemplo) podemos substituir toda a região pela sua cor média, caso contrário, significa que precisamos prosseguir na subdivisão em quadrantes, para avaliar as subimagens seguintes.

Considerando o exposto acima, seu trabalho será, baseado no código fonte base fornecido pelo professor, implementar um programa que construa uma estrutura de *quadtree* associada a uma imagem em tons de cinza, com o propósito de compacta-la. Esse código fonte está divido em alguns módulos, a saber:

- **EstruturasDeDados.h**
  Descrição das estruturas de dados a serem utilizadas nesse trabalho;
- **winGL.***
  Rotinas responsáveis pelo controle das janelas e dos desenhos;
  Rotinas responsáveis pela leitura de uma imagem em formato ppm colorida (3 bytes por pixel - RGB), e sua conversão para uma imagem em tons de cinza (1 byte por pixel - tons de cinza);
- **quadtree.***
  Programa principal e rotinas de tratamento de eventos de teclado, mouse e desenho.
- **PPM/*.ppm**
  Banco de imagens que você deve utilizar para rodar seu programa. Para simplificar seu trabalho, as imagens fornecidas são todas imagens quadradas de dimensões equivalentes a potências de 2.

A imagem é armazenada na aplicação pela variável ***imageGray*** como uma matriz de *bytes* (*unsigned char*) alocada dinamicamente pela rotina de conversão para tons de cinza. Sua dimensão é armazenada por duas variáveis inteiras iHeight e iWidth. Todas essas variáveis são globais.

A estrutura das informações a serem armazenadas em cada nó da *quadtree* já está definida, assim como o formato de armazenamento de cada nó da árvore. Não devem ser adicionados campos novos nessas estruturas. Caso sinta essa necessidade, consulte o professor para verificar se é realmente necessário.

O programa deve iniciar mostrando a imagem original. Através do teclado o usuário deve ser capaz de visualizar a imagem de duas formas distintas:

- A representação da imagem a partir de um certo nível da *quadtree* completa. O nível pode ser ampliado ou reduzido de forma iterativa (Figura 2);
- A representação da imagem comprimida para um determinado valor de erro percentual. O valor do erro deve ser incrementado ou decrementado em intervalos de 5%.

Figura 2 - Imagem de níveis distintos da *quadtree* completa.

Os trabalhos deverão ser desenvolvidos individualmente ou em duplas. O código fonte gerado deve ser comentado e legível. Acompanhando o código fonte um relatório técnico deve ser entregue, descrevendo as estruturas de dados utilizadas, uma justificativa e quais testes foram realizados para testar o funcionamento do programa (descrição, objetivo e resultado).

O trabalho deverá ser entregue no dia da terceira avaliação, impreterivelmente. O código fonte e a documentação deverão ser entregues gravados em um CD. Em caso de qualquer defeito de gravação, arquivos corrompidos ou qualquer outro problema similar o trabalho será considerado não entregue. Portanto, verifique bem o que for entregar!!

A cooperação entre alunos e grupos é considerada salutar. No entanto, trabalhos com alto grau de similaridade serão tratados como “plágio”, o que resultará em avaliação ***zero*** para todos os envolvidos.

Esse trabalho será contabilizado como 40% da nota da segunda avaliação.

Qualquer duvida adicional procurem o professor.