Compressao de imagens com uma QuadTree

→ LEMBREM-SE DE USAR PAPEL E CANETA COMO RASCUNHO ANTES DE IMPLEMENTAR <<--

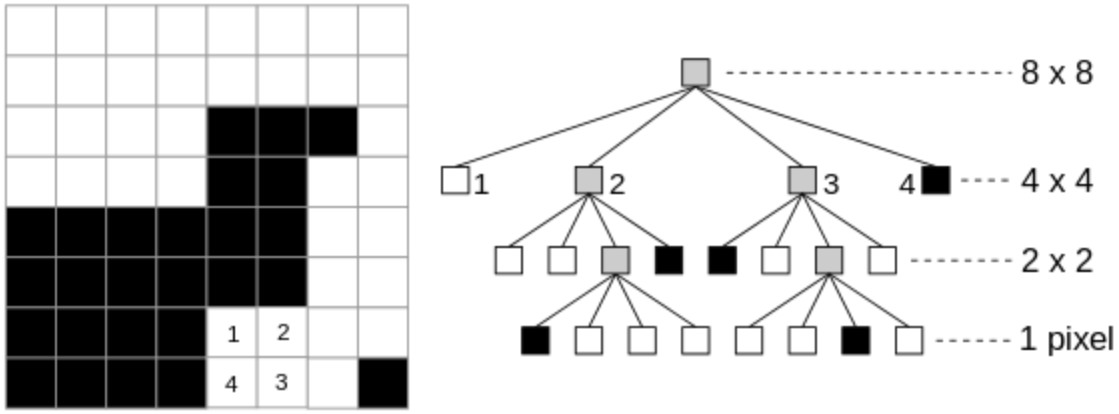
Trabalho 3: data limite para submissão: **05/07/2018 as 17:59**

Material de apoio a ser utilizado no trabalho:
https://drive.google.com/open?id=128o5M-B9QX33wiM1Z_Qt5r-oUnU-URfu

Introducao
Existem diversos tipos de árvores na computação. Um importante tipo e' a QuadTree, que e' muito utilizada para o processamento de dados 2D (ha tambem a OcTree, utilizada para dados 3D) na area de computacao grafica.

Uma QuadTree divide uma area em quatro quadrantes e repete esse processo de forma recursiva em cada um dos quadrantes ate que um determinado criterio de parada seja atingido. Isso e' utilizado para representar dados, indexar, etc.

A figura abaixo exibe uma imagem 8x8 (esquerda) preto e branca representada por uma QuadTree (direita):



(fonte: wikipedia)

Essa imagem foi dividida em 4 quadrantes (os numeros 1,2,3,4 no lado esquerdo indicam a ordem com que os quadrantes sao numerados) e cada nivel da arvore representa um refinamento (com o dobro da resolucao) dos quadrantes. Observe que no quadrante 1 do primeiro nivel todos os pixels sao brancos e, assim, não e' preciso refinar tal quadrante. O mesmo vale para o quadrante 4.

O quadrante 2, por outro lado, foi refinado. Considere o quadrante 2: seus “subquadrantes” 1 e 2 possuem apenas pixels brancos (e, portanto, não precisam ser refinados). O subquadrante 3, por outro lado, precisa ser refinado criando mais um nivel.

Intuitivamente quadtrees sao boas para comprimir imagens contendo varias regioes uniformes (que não precisam ser subdivididas). Veja alguns exemplos de figuras comprimidas aqui: http://demonstrations.wolfram.com/QuadtreeSubdivision/ e no Google Images: https://www.google.com.br/search?safe=off&biw=1067&bih=490&tbm=isch&sa=1&ei=VmMeW4vQH8KiwgSGp6zICA&q=quadtree&oq=quadtree&gs_l=img.3...0.0.0.6021.0.0.0.0.0.0.0.0..0.0....0...1c..64.img..0.0.0....0._y30VMAIYgI

Compressao com perdas
O tipo de compressao apresentado anteriormente e’ *lossless* (não ha qualquer perda de informacao). E’ possivel tambem realizar uma compressao com perdas (mas com melhor taxa de compressao): por exemplo, podemos deixar de expandir um quadrante se 90% dos pixels nesse quadrante forem da mesma cor (ou seja, se o quadrante for razoavelmente uniforme).

Imagens coloridas
Imagens coloridas podem ser comprimidas de forma semelhante a imagens em preto e branco. A ideia e’ que cada pixel armazena uma cor (neste trabalho representaremos cada cor utilizando o formato RGB, ou seja, cada cor sera representada por 3 numeros) e um quadrante e’ refinado se houver cores diferentes nele.

No caso de compressao com perdas, podemos verificar o quao uniforme um quadrante e’ e, entao, dividi-lo se ele não estiver uniforme.

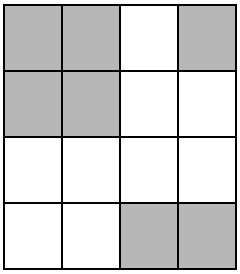
Quadrantes
Atencao: neste trabalho usaremos a mesma numeracao de quadrantes tipicamente utilizada em matematica:

1	0
2	3

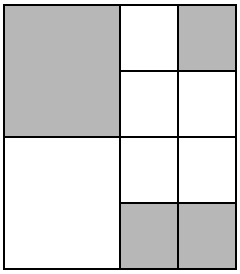
Armazenando uma QuadTree em um arquivo de texto
Podemos armazenar uma QuadTree em um arquivo de texto utilizando delimitadores para representar os filhos de cada nodo.

Neste trabalho, havera parenteses delimilitando cada nodo. Se o nodo for expandido, seus quatro filhos (cada um delimilitado por parentesis) estarao dentro dos parenteses do nodo. Caso contrario, a cor do nodo sera armazenada dentro dele (o valor dos componentes r, g e b serao armazenados separados por virgula).

Considere, por exemplo, a imagem 4x4 abaixo. Suponha, por simplicidade, que em vez de armazenarmos os componentes RGB vamos armazenar apenas 1 (para pixels pretos) ou 0 (pixels brancos):

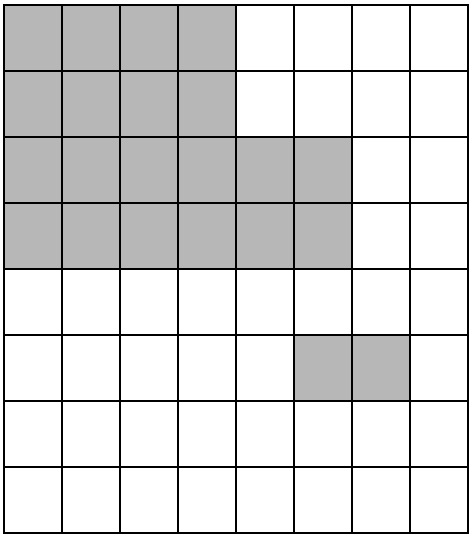


Na representacao exata, os quadrantes 0 e 3 (do primeiro nivel) sao expandidos criando um segundo nivel. Veja o resultado abaixo:

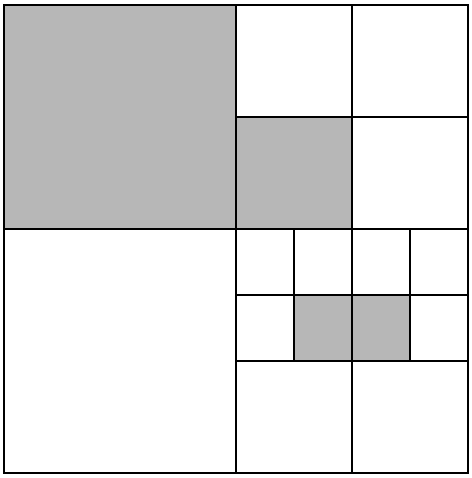


Podemos representar a figura acima em modo texto da seguinte forma (as cores usadas na fonte do texto sao apenas para facilitar a visualizacao): o vermelho indica o quadrante 0 do primeiro nivel, o verde o quadrante 1, o amarelo o quadrante 2 e o azul o quadrante 3):
((1)(0)(0)(0))(1)(0)((0)(0)(1)(1))

Considere tambem a figura abaixo.



No caso de compressao sem perdas, ela seria comprimida para:



E poderia ser representada textualmente como:
(((0)(0)(1)(0))(1)(0)(((0)(0)(1)(0))((0)(0)(0)(1))(0)(0)))

Nesse trabalho, como trabalhamos com imagens coloridas, em vez de numeros (como o “0”) teriamos triplas de numeros em cada nodo não expandido (exemplo: “127,0,5”)

O formato de imagens PPM
Neste trabalho utilizaremos o formato de arquivo PPM (em modo texto). Basicamente esse formato representa os pixels individualmente e armazena 3 valores de cores para cada pixel (os valores R, G e B).

Mais informacoes podem ser obtidas aqui (e em outros sites):
http://netpbm.sourceforge.net/doc/ppm.html

Por simplicidade, não suportaremos comentarios nos arquivos PPM. Alem disso, o codigo para ler/gravar imagens em formato PPM sera provido (veja a classe PPMImage).

Criterio para o refinamento de um quadrante (na compressao com perdas)
Definir um bom criterio que alia qualidade dos resultados a boa taxa de compressao e’ um desafio. Neste trabalho o criterio para decidir se um quadrante devera ser refinado ja foi implementado na classe PPMImage (veja a funcao “isUniform”) e e’ bastante simples (baseado na media e no desvio padrao das cores).
Alem de decidir se uma regiao (quadrante) devera ser refinada, a funcao isUniform tambem retorna uma cor que representa essa regiao (caso ela seja uniforme). Essa sera a cor que seu algoritmo de compressao devera gravar no quadrante correspondente da imagem comprimida (ou seja, se a maioria das cores de um quadrante forem “avermelhadas”, tal quadrante sera representado por uma cor “avermelhada” na saida comprimida (e essa cor devera ser utilizada para preencher todos os pixels do quadrante ao descomprimir a imagem))

Seu programa devera chamar tal método para decidir se um quadrante sera dividido ou não. A ideia de calcular o desvio padrao das cores funciona relativamente bem, mas não e' o ideal… Pesquisar tecnicas melhores e criar um método melhor sera deixado como exercicio extra (valendo pontos extras).

Trabalho
Neste trabalho, os alunos deverao utilizar uma QuadTree para comprimir imagens coloridas armazenadas no formato PPM. Embora possa ser possivel resolver este trabalho sem criar uma QuadTree explicitamente, você devera declarar uma QuadTree, implementa-la e utiliza-la.

Seu programa devera ler a imagem da entrada padrao (stdin) e gravar o resultado na saida padrao (stdout). Ele devera ter um argumento: se o argumento for "comprimir" (sem aspas), ele devera ler uma imagem em formato PPM e gravar na saida padrao um texto representando a versao comprimida da imagem utilizando a ideia descrita acima.

Se o argumento for "descomprimir", ela devera ler uma imagem comprimida da entrada padrao e gravar na saida a imagem descomprimida (em formato PPM).

A classe PPMImage (PPMImage.h e PPMImage.cpp) não devera ser alterada! Modifique apenas o arquivo main.cpp (se necessario, crie arquivos extras para suas classes).

Exemplos de entrada e saida:
Abaixo temos alguns exemplos de figuras comprimidas SEM PERDAS (seu programa devera realizar **compressao com perdas** -- o exemplo abaixo foi criado apenas por motivos didaticos). A entrada representa a entrada para o programa utilizando a opcao "comprimir". Se for utilizada a opcao "descomprimir", a saida abaixo seria a entrada do programa e a entrada seria a saida esperada.

A primeira e segunda figura abaixo representam, respectivamente, as duas figuras apresentadas na secao "Armazenando uma QuadTree em um arquivo de texto" (o codigo RGB 0 0 0 representa a cor preta e o codigo 255 255 255 representa a cor branca).

Entrada (saida no caso de descompressao)	**Saida (entrada para compressao)**
P3 4 4 255 0 0 0 0 0 0 255 255 255 0 0 0 0 0 0 0 0 0 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 0 0 0 0 0 0	**4 255** **(((0,0,0)(255,255,255)(255,255,255)(255,255,255))(0,0,0)(255,255,255)((255,255,255)(255,255,255)(0,0,0)(0,0,0)))**

P3 8 8 255 0 0 0 0 0 0 0 0 0 0 0 0 255 255 255 255 255 255 255 255 255 255 255 255 0 0 0 0 0 0 0 0 0 0 0 0 255 255 255 255 255 255 255 255 255 255 255 255 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 255 255 255 255 255 255 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 0 0 0 0 0 0 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255	**8 255** **(((255,255,255)(255,255,255)(0,0,0)(255,255,255))(0,0,0)(255,255,255)(((255,255,255)(255,255,255)(0,0,0)(255,255,255))((255,255,255)(255,255,255)(255,255,255)(0,0,0))(255,255,255)(255,255,255)))**

Note que a versao comprimida das imagens contem uma linha com dois numeros N e Max_intensity. N representa a resolucao da imagem (a imagem sempre sera quadrada e, portanto, tera resolucao N x N) e Max_intensity sera a intensidade maxima dos componentes dos pixels (por simplicidade, o Max_intensity da imagem comprimida sera sempre igual ao Max_intensity da imagem descomprimida).

A seguir temos um exemplo de entrada e saida considerando a compressao com perdas (implementada utilizando a funcao “isUniform” para decidir quais quadrantes seriam refinados) -- note que seu programa devera funcionar igual ao exemplo abaixo.

Comando a ser executado:
./a.out comprimir < brasao.ppm > brasaoComprimido.txt
./a.out descomprimir < brasaoComprimido.txt > brasaoDescomprimido.pbm

As duas saidas acima (e as quatro abaixo) se encontram no material de apoio (pasta exemplos).

./a.out comprimir < pixels.ppm > pixelsComprimido.txt
./a.out descomprimir < pixelsComprimido.txt > pixelsDescomprimido.pbm

./a.out comprimir < capivara.ppm > capivaraComprimido.txt
./a.out descomprimir < capivaraComprimido.txt > capivaraDescomprimido.pbm

Pontos extras:
Conforme mencionado, a funcao provida para decidir quais quadrantes devem ser refinados pode ser melhorada para aumentar a taxa de compressao mantendo a qualidade dos resultados.

Como exercicio extra (valendo ate 20% a mais na nota do trabalho), adicione uma opcao de argumento para seu programa (o nome do argumento sera “extra”). Quando essa opcao for utilizada (ou seja, quando seu programa for chamado de forma similar a chamada

“./seuPrograma.out extra < entrada.ppm > comprimido.txt”) seu programa devera utilizar algum algoritmo melhor para decidir se cada quadrante devera ser refinado.

Você ganhara pontos extras se o resultado do seu programa for menor (ou seja, se a imagem for mais comprimida) do que o resultado obtido utilizando a funcao de refinamento provida (sem degradar a qualidade dos resultados de forma perceptivel). A pontuacao obtida dependera da melhoria na taxa de compressao e da qualidade dos resultados.

Para ganhar pontos extras você devera descrever sua ideia no arquivo README. Explique de forma sucinta seu algoritmo e o motivo dele funcionar melhor do que o algoritmo provido. Indique tambem as diferencas observadas (para diversas imagens) entre o tamanho das imagens comprimidas utilizando o criterio dado e o tamanho obtido utilizando o seu criterio.

Restricoes:
Por simplicidade, todas imagens serao quadradas e suas resolucoes serao potencias de 2.

Arquivo README

Seu trabalho deverá incluir um arquivo README.

Tal arquivo contera (pelo menos):

- Seu nome/matricula
- Informacoes sobre fontes de consulta utilizadas no trabalho

Submissao

Submeta seu trabalho utilizando o sistema Submitty ate a data limite. Seu programa sera avaliado de forma automatica (os resultados precisam estar corretos, o programa não pode ter erros de memoria, etc), passara por testes automaticos “escondidos” e a qualidade do seu codigo sera avaliada de forma manual.

Você devera enviar o arquivo README e todo o codigo fonte do seu trabalho (ele será compilado com o comando g++ *.cpp -O3 -std=c++11 -Wall)

Duvidas

Dúvidas sobre este trabalho deverão ser postadas no sistema Piazza. Se esforce para implementá-lo e não hesite em postar suas dúvidas!

Avaliacao manual

Principais itens que serao avaliados (alem dos avaliados nos testes automaticos):
- Comentarios
- Indentacao
- Nomes adequados para variaveis
- Separacao do codigo em funcoes logicas

- Uso correto de const/referencia
- Uso de variavies globais apenas quando absolutamente necessario e justificavel (uso de variaveis globais, em geral, e' uma ma pratica de programacao)
- Etc

Regras sobre plagio e trabalho em equipe

- Este trabalho devera ser feito de forma individual.
- Porém, os alunos podem ler o roteiro e discutir ideias/algoritmos em alto nivel de forma colaborativa.
- As implementações (nem mesmo pequenos trechos de codigo) não deverão ser compartilhadas entre alunos. Um estudante não deve olhar para o código de outra pessoa.
- Crie um arquivo README (submeta-o com o trabalho) e inclua todas as suas fontes de consulta.
- Não poste seu codigo (nem parte dele) no Piazza (ou outros sites) de forma publica (cada aluno e' responsavel por evitar que outros plagiem seu codigo).
- Trechos de código não devem ser copiados de livros/internet. Se você consultar algum livro ou material na internet essa fonte deverá ser citada no README.
- Se for detectado plagio em algum trecho de codigo do trabalho a nota de TODOS estudantes envolvidos sera 0. Alem disso, os estudantes poderao ser denunciados aos orgaos responsaveis por plagio da UFV.