MAP-2121 - Segundo Exercício Programa - 2013

Instruções gerais -

Os exercícios computacionais pedidos na disciplina Cálculo Numérico têm por objetivo fundamental familiarizar o aluno com problemas práticos que requeiram técnicas numéricas em sua solução. Faça o seu programa antes da data da P3. Ele pode ser desenvolvido na linguagem de programação de sua preferência.

1 Introdução

O objetivo deste exercício programa é implementer um método de compactação de imagens, usando idéias similares às encontradas na análise harmônica e no método dos mínimos quadrados. O método que deverá ser implementado é parecido com o utilizado na conversão de uma imagem para o padrão JPEG, mas possui algumas diferenças, tanto para facilitar a implementação do EP quanto para que uma explicação do mesmo seja mais clara. Em particular, trataremos apenas de imagens em níveis de cinza, e não usaremos imagens coloridas. Os alunos interessados em descobrir como o padrão JPEG realmente funciona podem ler uma explicação resumida e acessível na Wikipedia, por exemplo.

2 Representação de Imagens em Arquivos Texto

Esta seção descreve a representação de imagens em arquivos do tipo texto. Tais formatos, apesar de serem muito menos populares do que os formatos binários que ocupam menos espaço, são utilizados em vários programas de conversão.

Estaremos interessados nos formatos PNM (portable anyformat): PBM (portable bitmap file format), PGM (portable greymap file format) e PPM (portable pixmap file format) que correspondem, respectivamente, a imagens em preto e branco, imagens em escala de cinza e imagens coloridas. Apesar de, neste exercício, utilizarmos apenas o padrão PGM, incluimos uma explicação dos outros formatos PNM por completude.

Caso você deseje observar o resultado prático da tarefa desenvolvida, você precisará de um programa capaz de visualizar figuras no formato PGM. Se você usa Linux, pode utilizar o xv ou o gimp. Caso você use Windows, existem vários progamas diferentes que podem ser baixados sem custo, por exemplo o Irfanview.

2.1 PBM

A definição de um arquivo PBM é a seguinte:

- Um "número mágico"para identificar o tipo de arquivo. Um arquivo PBM usa os caracteres "P1".
- Espaço (brancos ou quebras de linha)
- Uma largura, em caracteres decimais.
- Espaço.
- Uma altura, em caracteres decimais.
- Espaço.
- (Largura × Altura) caracteres "0"ou "1", começando do canto superior esquerdo da imagem. O caracter "1"representa preto e "0"representa branco.
- Espaços na secção acima serão ignorados.
- Linhas que começam com # são comentários, e portanto são ignoradas.
- Nenhuma linha pode ser maior do que 70 caracteres.

Por exemplo:

```
P1
#feep.pbm
24 7
000000000000000000000000
011110011110011110011110
010000010000010000010010
011100011100011100011110
```

01000001000001000001000
01000001111001111001000
00000000000000000000000

corresponde à imagem

2.2 PGM

A definição de um arquivo PGM é a seguinte:

- Um "número mágico"para identificar o tipo de arquivo. Um arquivo PGM usa os caracteres "P2".
- Espaço (brancos ou quebras de linha)
- Uma largura, em caracteres decimais.
- Espaço.
- Uma altura, em caracteres decimais.
- Espaço.
- O nível máximo de cinza, em caracteres texto decimais.
- (Largura × Altura) caracteres entre "0"e o valor máximo especificado, separados por espaço, começando do canto superior esquerdo da imagem. O caracter "0"representa preto e o nível máximo representa branco.
- Linhas que começam com # são comentários, e portanto são ignoradas.

- Nenhuma linha pode ser maior do que 70 caracteres.

Por exemplo:

```
P2
#feep.pgm
24 7
15
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 3 3 3 3 0 0 7 7 7 7 0 0 11 11 11 11 0 0 15 15 15 15 0
0 3 0 0 0 0 0 7 0 0 0 0 0 11 0 0 0 0 0 15 0 0 15 0
0 3 3 3 0 0 0 7 7 7 0 0 0 11 11 11 0 0 0 15 15 15 15 0
0 3 0 0 0 0 0 7 0 0 0 0 0 11 0 0 0 0 0 15 0 0 0 0
0 3 0 0 0 0 0 7 7 7 7 0 0 11 11 11 11 0 0 15 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

corresponde à imagem

2.3 PPM

A definição de um arquivo PPM é a seguinte:

- Um "número mágico"para identificar o tipo de arquivo. Um arquivo PPM usa os caracteres "P3".
- Espaço (brancos ou quebras de linha)
- Uma largura, em caracteres decimais.
- Espaço.
- Uma altura, em caracteres decimais.

- Espaço.
- O nível máximo de componentes de cor, em caracteres texto decimais.
- (Largura × Altura) triplas de níveis de cor, separados por espaço, começando do canto superior esquerdo da imagem. Cada valor das triplas corresponde, respectivamente, ao nível de vermelho, verde e azul. O valor "0"significa que a cor não aparece, enquanto que o valor máximo significa que a cor está no nível máximo.
- Linhas que começam com # são comentários, e portanto são ignoradas.
- Nenhuma linha pode ser maior do que 70 caracteres.

Por exemplo:

```
P3
#feep.ppm
4 4
15
0 0 0 0 0 0 0 0 0 15 0 15
0 0 0 15 7 0 0 0 0 0 0 0
0 0 0 0 0 0 15 7 11 0 0 0
0 15 15 0 0 0 0 0 0 0 0 0
```

corresponde à imagem

3 Mudança de base

O espaço das matrizes $n \times m$ é um espaço vetorial de dimensão nm. **Por simplicidade nas fórmulas abaixo, contaremos as linhas e as colunas**

de uma matriz A qualquer iniciando com 0. Assim, a matriz A tem entradas $(a_{i,j})$ onde $0 \leq i \leq n-1, 0 \leq j \leq m-1$. Podemos introduzir no espaço das matrizes o produto interno:

$$< A, B >= \sum_{i=0}^{n-1} \sum_{j=0}^{m-1} a_{i,j} b_{i,j}.$$

Como todo espaço vetorial de dimensão finita dotado de produto interno, podemos encontrar bases ortogonais para este espaço. A base canônica para este espaço é dada pelas matrizes com uma única entrada igual a 1 e o resto das entradas iguais a zero, isto é, $E_{i,j}, 0 \leq i \leq n-1, 0 \leq j \leq m-1$, onde a entrada (k,l) da matriz $E_{i,j}$ é 1 se $i = k$ e $j = l$ e é zero caso contrário. Claramente, $A = \sum_{i=0}^{n-1} \sum_{j=0}^{m-1} a_{i,j} E_{i,j}$, ou seja, as entradas $a_{i,j}$ nada mais são do que as coordenadas de A na base canônica.

Porém, também podemos escolher outras bases ortogonais e, para compactação de imagens, uma base particularmente útil é chamada de base dos cossenos discretos. As matrizes $X_{k,l}, 0 \leq k \leq n-1, 0 \leq l \leq m-1$ (**note que, de novo, os índices foram numerados a partir de** 0) cuja entrada (i,j) é dada por

$$X_{k,l}(i,j) = \alpha(k)\beta(l) cos\left[\frac{\pi}{n} k(i+\frac{1}{2})\right] cos\left[\frac{\pi}{m} l(j+\frac{1}{2})\right],$$

onde $\alpha(0) = \sqrt{\frac{1}{n}}$ e $\alpha(k) = \sqrt{\frac{2}{n}}$ caso $k > 0$, e onde $\beta(0) = \sqrt{\frac{1}{m}}$ e $\beta(l) = \sqrt{\frac{2}{m}}$ se $l > 0$, formam uma outra base ortonormal para este espaço (Exercício). Portanto, também podemos escrever $A = \sum_{k=0}^{n-1} \sum_{l=0}^{m-1} x_{k,l} X_{k,l}$, onde, neste caso, $x_{k,l}$ são as coordenadas de A na base dos cossenos discretos.

Conhecendo os $a_{i,j}$, podemos encontrar as coordenadas $x_{k,l}$ pela fórmula (Projeção numa base ortogonal):

$$x_{k,l} = \frac{< A, X_{k,l} >}{< X_{k,l}, X_{k,l} >} = \alpha(k)\beta(l) \sum_{i=0}^{n-1} \sum_{j=0}^{m-1} a_{i,j} cos\left[\frac{\pi}{n} k(i+\frac{1}{2})\right] cos\left[\frac{\pi}{m} l(j+\frac{1}{2})\right],$$

chamada de DCT(normalizada), ou discrete cosine transformation, onde, como antes, $\alpha(0) = \sqrt{\frac{1}{n}}$ e $\alpha(k) = \sqrt{\frac{2}{n}}$ caso $k > 0$, e onde $\beta(0) = \sqrt{\frac{1}{m}}$ e $\beta(l) = \sqrt{\frac{2}{m}}$ se $l > 0$.

A inversa desta transformação, chamada de IDCT(normalizada), permite

encontar os valores de $a_{i,j}$ quando conhecemos as coordenadas de A na base dos cossenos discretos. Esta fórmula nada mais é do que

$$a_{i,j} = \sum_{k=0}^{n-1} \sum_{l=0}^{n-1} x_{k,l} X_{k,l}(i,j).$$

4 O Algoritmo de Compactação

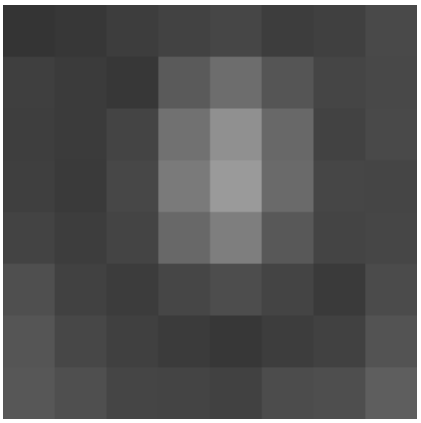
O algoritmo que vocês implementarão passa pelas seguintes etapas:

Dada uma figura com 256 níveis de intensidade de cinza, representada por uma matriz $N \times M$ com entradas inteiras entre 0 e 255, você deve, primeiramente, particionar esta matriz em blocos de 8×8 pixels.

Cada bloco destes deve ser processado da seguinte forma: Se $A_{8\times 8}$ é a matriz

$$A = \begin{pmatrix} 52 & 55 & 61 & 66 & 70 & 61 & 64 & 73 \\ 63 & 59 & 55 & 90 & 109 & 85 & 69 & 72 \\ 62 & 59 & 68 & 113 & 144 & 104 & 66 & 73 \\ 63 & 58 & 71 & 122 & 154 & 106 & 70 & 69 \\ 67 & 61 & 68 & 104 & 126 & 88 & 68 & 70 \\ 79 & 65 & 60 & 70 & 77 & 68 & 58 & 75 \\ 85 & 71 & 64 & 59 & 55 & 61 & 65 & 83 \\ 87 & 79 & 69 & 68 & 65 & 76 & 78 & 94 \end{pmatrix}$$

que representa o bloco abaixo, primeiro subtraímos 128 de cada entrada de



A, encontrando a matriz B. Depois disso você deve encontrar a matriz X das

coordenadas de B na base apropriada de cossenos discretos, i.e., a entrada (k, l) de X corresponde ao coeficiente $x_{k,l}$ dado pela DCT. Até este momento o processo realizado é completamente inversível, portanto o conhecimento de X nos permite reconstruir A perfeitamente. Porém, como as entradas de A são inteiras e as entradas de X não, a memória necessária para armazenar X é ainda maior do que a necessária para armazenar A.

Após encontrar os coeficientes de X, fazemos a primeira operação de compressão, arredondando os mesmos para o inteiro mais próximo, encontrando uma nova matriz Y. Veja que agora a memória necessária para armazenar Y é a mesma que a necessária para armazenar A.

Finalmente, como o olho humano é menos sensível a variações de cinza com frequência alta, e como estas variações estão representadas pelas coordenadas $y_{i,j}$ onde $i+j$ são grandes, iremos ignorar todas as entradas tais que $i+j \geq d$. O valor de d será escolhido de acordo com a taxa de compressão que pretendemos obter. Por exemplo, se escolhermos $d = 8$, faremos $z_{i,j} = y_{i,j}$ se $i + j < 8$ e $z_{i,j} = 0$ caso contrário. Assim, se a matriz X fosse:

$$X = \begin{pmatrix} 129.5 & 13.42 & 11.65 & 12.30 & -3.14 & 2.71 & -97.8 & 90.5 \\ 0.84 & -3.01 & 22.54 & -12.30 & 34.2 & -52.51 & 77.8 & 59.5 \\ -0.31 & 3.42 & 1.65 & 2.30 & -31.14 & 8.71 & -74.48 & -0.5 \\ 42.0 & -1.35 & 24.30 & -1.14 & 8.34 & -74.44 & -0.01 & -0.04 \\ -3.01 & 52.54 & -2.30 & 43.2 & -2.51 & 7.8 & 9.5 & 43.6 \\ 14.2 & 12.5 & -1.23 & -3.14 & 2.71 & -9.8 & 9.5 & 41.1 \\ 21.65 & -52.30 & -31.14 & 8.71 & -74.48 & -0.5 & 0.02 & 0.21 \\ 33.3 & -42.0 & -1.35 & 24.30 & -1.14 & 8.34 & -74.44 & -0.01 \end{pmatrix}$$

teremos que

$$Y = \begin{pmatrix} 130 & 13 & 12 & 12 & -3 & 3 & -98 & 91 \\ 1 & -3 & 23 & -12 & 34 & -53 & 78 & 60 \\ 0 & 3 & 2 & 2 & -31 & 9 & -74 & 0 \\ 42 & -1 & 24 & -1 & 8 & -74 & 0 & 0 \\ -3 & 52 & -2 & 43 & -3 & 8 & 10 & 43 \\ 14 & 13 & -1 & -3 & 3 & -10 & 10 & 41 \\ 22 & -52 & -31 & 9 & -74 & 0 & 0 & 0 \\ 33 & -42 & -1 & 24 & -1 & 8 & 74 & 0 \end{pmatrix}$$

e

$$Z = \begin{pmatrix} 130 & 13 & 12 & 12 & -3 & 3 & -98 & 91 \\ 1 & -3 & 23 & -12 & 34 & -53 & 78 & 0 \\ 0 & 3 & 2 & 2 & -31 & 9 & 0 & 0 \\ 42 & -1 & 24 & -1 & 8 & 0 & 0 & 0 \\ -3 & 52 & -2 & 43 & 0 & 0 & 0 & 0 \\ 14 & 13 & -1 & 0 & 0 & 0 & 0 & 0 \\ 22 & -52 & 0 & 0 & 0 & 0 & 0 & 0 \\ 33 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Veja que Z possui apenas $\frac{(9)(8)}{2} = 36$ entradas relevantes, pois as entradas abaixo da anti-diagonal principal são nulas, logo a memória necessária para representá-la é $\frac{36}{64} = \frac{9}{16}$ da necessária para representar o bloco A.

5 Reconstituição da Imagem

Por outro lado, caso você necessite reconstituir a sua imagem após a compressão, podemos começar com a matriz Z e aplicar a IDCT, chegando a uma matriz $\tilde{B}$. Após isso, somamos a cada entrada de $\tilde{B}$ 128, e chegamos a uma matriz $\tilde{A}$ que, esperamos, deve ser próxima da matriz original A. O último passo é arredondar as entradas de $\tilde{A}$ para o inteiro mais próximo, obtendo a matriz $\bar{A}$.

6 Tarefa

Você deverá escrever dois programas, na linguagem de sua preferência (por exemplo, FORTRAN). O primeiro programa deve receber, como entrada, um arquivo .pgm onde o número de pixels horizontais e verticais é um múltiplo de 8 e o valor de d. O seu programa deve particionar a imagem recebida em blocos 8 por 8, e compactar cada um destes blocos segundo o algoritmo descrito, escrevendo o resultado num outro arquivo de tipo texto, sem usar memória desnecessária.

O segundo programa terá como entrada o arquivo gerado no primeiro programa, e o valor de d, e deve reconstituir, primeiramente as matrizes Z e depois, da melhor forma possível, a imagem original, gerando como resposta um arquivo no formato PGM.

Teste o seu programa, utilizando os valores $d = 6, 8$ e 10, para as três imagens abaixo, disponíveis no site da matéria.

Divirta-se!