UNIVERSIDADE DE SÃO PAULO - ICMC
DEPARTAMENTO DE CIÊNCIAS DE COMPUTAÇÃO
SCC0561 Multimídia - 1º Sem/2017
Prof. Marcelo Manzato (mmanzato@icmc.usp.br).

PROJETO

Compressão de Imagem

Divulgação: 07/04/2017.
Datas de entrega: Fase 1: 02/05/2017; Fase 2: 27/06/2017.
Grupos de no máximo 2 alunos.

Objetivos:

O objetivo principal do projeto é colocar em prática conceitos teóricos abordados em sala de aula relacionados a técnicas de codificação de mídias. Para isso, os grupos deverão implementar algumas técnicas de compressão e descompressão – a saber – aplicadas a imagens do tipo BMP (*Bitmap Image Format*).

Fases:

O projeto será dividido em duas fases incrementais, cada uma possuindo datas de entrega específicas.

Projeto - Fase 1 – entrega: 02/05/2016: cada grupo deverá desenvolver um compressor (e respectivo descompressor) de imagem usando o algoritmo **Lempel-Ziv-Welsh (LZW)** (visto em sala de aula) com tamanho de dicionário fixo.

A entrada dos dados será uma imagem sem compressão no formato BMP. Um *parser* simplificado e modificado obtido desta página[1] é fornecido juntamente com essa especificação, sendo que o mesmo será capaz de ler todas as imagens que serão usadas pelo professor para correção do trabalho. Repare que ao executar o *parser* com uma imagem BMP sem compressão (algumas também estão sendo oferecidas como exemplo), o programa imprime na tela informações sobre o cabeçalho do arquivo, e em seguida, os pixels da imagem no formato **"(r,g,b)"**. Sua função será codificar esses valores usando o algoritmo LZW.

Neste trabalho, o algoritmo LZW deverá ser um pouco diferente ao que foi apresentado em sala. Na aula, foi apresentada uma versão simplificada que usava caracteres especiais para separar as palavras de um texto. No contexto deste trabalho, você deverá implementar uma versão que não precisa de separadores, mas que ainda assim maximize a frequência de ocorrência de símbolos adjacentes (sub-strings). Alguns materiais podem auxiliar:

1) https://www.youtube.com/watch?v=rZ-JRCPv_O8
2) https://www.youtube.com/watch?v=MQM_DsX-LBI
3) https://www.cs.duke.edu/csed/curious/compression/lzw.html

[1] http://paulbourke.net/dataformats/bmp/

Como saída, o codificador deverá gerar um arquivo **binário** contendo os dados comprimidos. O decodificador, por sua vez, deverá ler o arquivo comprimido, descomprimir os dados, e gerar outro arquivo BMP, o qual possa ser lido por qualquer visualizador de imagem.

Nesta fase, o tamanho do dicionário será determinado pela quantidade de bits para os índices, e deverá ser fixo durante toda execução (LZW estático). Essa quantidade deverá ser informada como parâmetro de entrada de seu programa. Caso o valor informado não seja suficiente (ou seja, todo dicionário foi preenchido e ainda há mais dados para serem codificados), o programa deverá finalizar a execução com uma mensagem de erro.

O número de bits informado no codificador deverá ser armazenado no cabeçalho do arquivo binário. Desse modo, o decodificador, após ler o cabeçalho, saberá quantos bits deverão ser usados para cada índice, podendo então alocar o dicionário com o tamanho informado.

O compressor deverá operar por linha de comando, aceitando como parâmetro o nome e caminho do arquivo BMP original, e também o nome e caminho do arquivo binário a ser gerado. O descompressor, por sua vez, também será operado por linha de comando, aceitando ambos os parâmetros também (arquivo binário de entrada e arquivo BMP de saída). O seguinte formato de execução deverá ser obedecido:

Para compressão:

```
./encode -i arquivo_original.bmp -o arquivo_binario.bin -n 10
```

Para descompressão:

```
./decode -i arquivo_binario.bin -o arquivo_descomprimido.bmp
```

No exemplo acima, o codificador irá comprimir o arquivo "`arquivo_original.bmp`", usando 10 bits para índice (o tamanho do dicionário, portanto, será de 2^{10}=1024 posições), e irá gerar o arquivo "`arquivo_binario.bin`". O decodificador, por sua vez, fará a leitura do arquivo comprimido "`arquivo_binario.bin`", e gerará a imagem "`arquivo_descomprimido.bmp`". Repare que o número de bits não deve ser informado para o decodificador, sendo que este obterá esse valor no cabeçalho do arquivo comprimido. Os comandos `encode` e `decode` são arquivos executáveis, ou seja, deverão ser criados ao compilar seu projeto.

É importante salientar que parte da correção será feita automaticamente, portanto, é fundamental obedecer o formato acima especificado, caso contrário, a nota atribuída ao programa pelo sistema de correção será zero.

Projeto - Fase 2 – entrega: 27/06/2017: cada grupo deverá modificar o compressor da Fase 1 para permitir tamanho variável de tabela/dicionário. O compressor deverá operar inicialmente com 10 bits, oferecendo 1024 entradas para a tabela. Após o preenchimento de todas as entradas, tanto o codificador quanto o decodificador deverão incrementar automaticamente a quantidade de bits dos índices, aumentando também o tamanho do dicionário. Esse processo deverá ser executado recursivamente sempre que a capacidade do dicionário atingir o valor máximo.

Deste modo, a fase 2 não irá mais finalizar a execução caso o dicionário não seja suficiente para codificação, tampouco informar o decodificador sobre o tamanho do dicionário utilizado. Pelo contrário, ambos deverão inicializar o dicionário com 10 bits, e iniciar o processamento até que o dicionário tenha

sido totalmente preenchido. A partir daí, ambos deverão agora trabalhar com 11 bits para os índices, continuando o processamento até que o dicionário estendido tenha sido totalmente preenchido. Isso será realizado quantas vezes forem necessárias para que todo o arquivo de entrada tenha sido processado.

O seguinte formato de execução deverá ser obedecido:

Para compressão:

```
encode -i arquivo_original.bmp -o arquivo_binario.bin -d
```

sendo que a opção '`-d`' é para diferenciar a fase 2 (dicionário dinâmico) da fase 1 (dicionário estático). Nesta versão, a opção '`-n`' também deverá estar disponível, a fim de que o programa possa ser executado conforme especificado na fase anterior.

Para descompressão:

```
decode -i arquivo_binario.bin -o arquivo_descomprimido.bmp -d
```

sendo que a opção '`-d`' é para diferenciar a fase 2 (dicionário dinâmico) da fase 1 (dicionário estático). Caso esta opção não seja informada, o decodificador deverá ser executado conforme especificado na fase 1.

Novamente, atente que parte da correção dessa segunda fase também será feita automaticamente, portanto, é fundamental obedecer ao formato especificado acima.

Considerações sobre o projeto (fase 1 e fase 2):

1. Cada grupo deverá desenvolver um projeto original. Cópias, mesmo parciais, não serão toleradas.
2. A implementação deverá ser feita em C ou C++, usando os compiladores gcc ou g++ (ambiente Unix). Não poderá ser utilizada nenhuma biblioteca externa, exceto o *parser* BMP fornecido com essa especificação.
3. O programa deverá operar por linha de comando (sem interface gráfica).
4. Deverão ser entregues: os arquivos contendo o código-fonte e um arquivo TXT (relatório) contendo as informações do grupo e explicação de como compilar o programa. Todos os arquivos deverão ser agrupados em um único arquivo ZIP.
5. Modo de entrega: um dos componentes do grupo deverá fazer *upload* dos arquivos no seu respectivo escaninho no Tidia até a data da entrega.
6. Será descontado 1.0 ponto da nota final por dia de atraso.

Bom trabalho!