ICC 2 – Trabalho 05
Prof. Rodrigo Mello

Estagiários PAE

Martha Dais Ferreira
Fausto Guzzo da Costa

Data Máxima para Perguntas: 10/10/2014

Data de Entrega do Trabalho: 15/10/2014

Fórum para Envio de Perguntas: https://groups.google.com/d/forum/icc2-2014

Descrição

Considere um arquivo de áudio gravado em formato binário. Esse arquivo contém amplitudes do áudio gravadas em blocos de bytes. As amplitudes são resultantes de variações no diafragma do microfone. Ao gravar o áudio, conforme falamos no microfone, há variações no diafragma, que são codificadas em números a cada instante, e gravados em um arquivo binário.

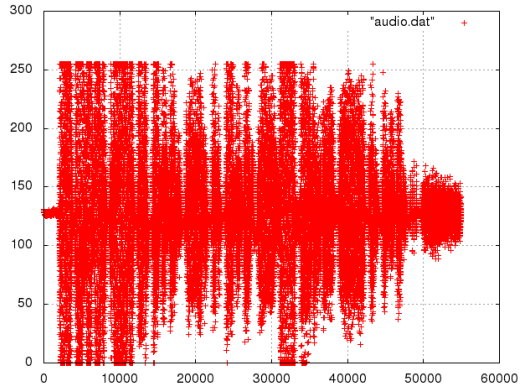
O valor da pressão de nossa voz sobre o diafragma do microfone é medido por um *unsigned byte*. Dessa maneira há 8 bits para representar a situação do diafragma em dado instante de tempo. Cada valor de amplitude varia de 0 até 255, pois ele é representado por um unsigned byte (ou seja, 1 byte sem sinal), em que há 8 bits que podem ser usados: logo $2^8 = 256$ possíveis valores. Como começa em 0 vai até 255.

O valor 127 (valor central) significa que o diafragma está em repouso, ou seja, ninguém está falando. Valores abaixo de 127 e acima de 127 significam variações do diafragma impostas por nossa voz.

Ao ler esse arquivo binário, **um byte por vez**, você teria valores como:

127 128 180 182 187 190 183 172 171 169 127 128

Ao construir um gráfico (plotar) esse arquivo com inteiros no formato texto (no Linux, pode-se usar o comando gnuplot para plotar) temos algo na forma:



Em que temos o tempo no eixo x e a amplitude do diafragma (1 byte) no eixo y. Como o arquivo produzido foi gravado com 8000 Hertz, há 8000 observações do diafragma por segundo de gravação. Esse número define a amostragem no tempo. Na Figura acima há cerca de 55000 observações no total, cada uma corresponde a 1 byte. Dividindo as 55000 observações por 8000 (amostragem no tempo) é possível saber que esse arquivo representa 6,8 segundos de gravação, ou seja, foram coletadas 8000 variações do diafragma do microfone para cada segundo.

A sua tarefa nesse trabalho consiste em **ler arquivos de áudio binários** e **processar utilizando o algoritmo K-Médias** que será descrito logo abaixo. Esse algoritmo irá substituir os valores de amplitude originais do áudio por novos valores, os quais permitirão maior compactação do áudio final produzido. Por exemplo, considere que o arquivo contém os seguintes valores de amplitude:

127 128 180 182 187 190 127 128 75 77 68 72 127 127

Agora considere que o algoritmo K-Médias será executado para K=3 grupos. Cada um desses 3 grupos irá conter um valor médio de amplitude capaz de representar uma faixa de valores. Para compreender melhor o funcionamento do algoritmo K-Médias, considere que iremos inicializar os 3 grupos com os seguintes valores de amplitude:

Grupo 1 = 130
Grupo 2 = 184
Grupo 3 = 66

Como primeiro passo o algoritmo K-Médias procura associar cada um dos valores de amplitude do áudio original a um dos grupos. Essa associação se dá por proximidade. Por exemplo, o primeiro valor de amplitude é 127 o qual é mais próximo do Grupo 1, ou seja:

127 – Grupo 1 = 127 – 130 = -3
127 – Grupo 2 = 127 – 184 = -57
127 – Grupo 3 = 127 – 66 = 61

Por meio dessas diferenças, observa-se que o valor de amplitude 127 está mais próximo do Grupo 1. Sendo assim, para os valores originais de amplitude:

127 128 180 182 187 190 127 128 75 77 68 72 127 127

Constrói-se um vetor que indicar a qual grupo cada amplitude está associada, como segue:

1 1 2 2 2 2 1 1 3 3 3 3 1 1

Como próximo passo, o algoritmo K-Médias recalcula o valor de cada Grupo de acordo com uma média das amplitudes a ele associadas. Por exemplo, as amplitudes

127, 128, 127, 128, 127, 127 foram associadas ao Grupo 1
180, 182, 187, 190 foram associadas ao Grupo 2
75, 77, 68, 72 foram associadas ao Grupo 3

Logo, computamos a média desses valores para cada Grupo, assim obteremos:

Para Grupo 1 temos a média(127, 128, 127, 128, 127, 127) = 127.33
Para o Grupo 2 temos a média(180, 182, 187, 190) = 184.75
Para o Grupo 3 temos a média (75, 77, 68, 72) = 73

Assim, os valores anteriores dos Grupos que eram:

Grupo 1 = 130
Grupo 2 = 184
Grupo 3 = 66

Irão se tornar:

Grupo 1 = 127.33
Grupo 2 = 184.75
Grupo 3 = 73

Então o algoritmo volta ao passo de associar, novamente, cada uma das amplitudes do áudio original aos novos grupos, calcula novamente as médias e continua atualizando esses grupos até que uma condição seja satisfeita. Em nosso caso, a condição será até que a média de mudança dos grupos tenha sofrido uma alteração menor que T. Por exemplo:

O Grupo 1 era 130 e foi para 127.33 sofrendo uma alteração de |2.67|
O Grupo 2 era 184 e foi para 184.75 sofrendo uma alteração de |-0.75|
O Grupo 3 era 66 e foi para 73 sofrendo uma alteração de |-7|
A média para esses grupos seria de 3.47.

Suponha que T = 5. Neste caso, o algoritmo para de executar. Caso contrário, suponha T=1, então o algoritmo deveria continuar executando.

Após terminar a execução do algoritmo K-Médias, você deve produzir um novo vetor a partir das amplitudes do áudio original. Seja as amplitudes do áudio original dadas por:

127 128 180 182 187 190 127 128 75 77 68 72 127 127

E seja cada uma dessas amplitudes, no passo do algoritmo (i.e., após execução e nenhum Grupo superar uma alteração maior ou igual a T), associada a um Grupo na forma:

1 1 2 2 2 2 1 1 3 3 3 3 1 1

Logo, você irá criar um novo vetor de amplitudes em que cada amplitude será subtituída pela valor de seu Grupo. Neste caso seria:

127.33 127.33 184.75 184.75 184.75 184.75 127.33 127.33 73 73 73 73 127.33 127.33

Em seguida você deve truncar os números, ou seja, aplicar o operador floor sobre esse novo vetor e obter:

127 127 184 184 184 184 127 127 73 73 73 73 127 127

Agora você deverá gerar um arquivo binário de saída contendo esse novo vetor e escutá-lo. Cada valor de amplitude alterado será, também, armazenado como 1 byte no arquivo final. Sendo assim, como há 14 valores, o arquivo final terá 14 bytes neste caso.

Dicas

1) Para gerar um arquivo de áudio binário no Linux utilize:

arecord -t raw -c 1 -r 8000 test.raw

2) Para ouvir o arquivo de áudio binário no Linux utilize:

aplay -t raw -r 8000 -c 1 -f u8 test.raw

Há outros aplicativos como o Audacity e o VLC Media Player que funcionam para outros sistemas operacionais além do Linux e que podem ajudar a gravar áudios e a tocá-los

3) Leia mais sobre o K-Médias em http://en.wikipedia.org/wiki/K-means_clustering

Motivação

Este uso do algoritmo K-Médias permite a compactação de sinais de áudio. Busque compactar com Bzip2, com Zip ou com Rar o arquivo de áudio original e, após terminar o processamento, o arquivo final produzido por seu programa. Você verá que conforme o número de grupos K for menor, mais você poderá compactar seu áudio. Você também observará que irá perder informações, mas mesmo assim, com um certo valor de K, você poderá ainda ouvir o áudio após transformação.

Formato para os Casos de Teste

Formato da entrada:
<nome do arquivo binário de áudio>
<número K de grupos>
<Grupo 1>
<Grupo 2>
...
<Grupo K>
<valor para T>

Formato da saída:
<conteúdo do arquivo binário de saída impressa no stdout, i.e., na saída padrão>

Exemplo de Caso de Teste

Entrada fornecida:
arquivo.raw

```
3
130
184
66
5
```

Saída esperada:
```
<os bytes produzidos pelo arquivo de saída e impressos na stdout (saída padrão)>
```

Informações Importantes

1) Um dos objetivos da disciplina de ICC2 é o aprendizado individual dos conceitos de programação. A principal evidência desse aprendizado está nos trabalhos, que são individuais neste curso. Você deverá desenvolver seu trabalho sem copiar trechos de código de outros alunos ou da Internet, nem codificar em conjunto. Portanto, compartilhem ideias, soluções, modos de resolver o problema, mas não o código.

1.1) O plágio vai contra o código de ética da USP;

1.2) Quando autores e copiadores combinam, estão ludibriando o sistema de avaliação, por isso serão utilizadas ferramentas de análise plágio tais como o MOSS (http://moss.stanford.edu/);

1.3) O trabalho em grupo e a cooperação entre colegas é em geral benéfico e útil ao aprendizado. Para ajudar um colega você pode lhe explicar estratégias e ideias. Por exemplo, pode explicar que é preciso usar dois loops para processar os dados, ou que para poupar memória basta usar uma certa estrutura de dados, etc. O que você não deve fazer é mostrar o seu código. Mostrar/compartilhar o código pode prejudicar o aprendizado de seu colega:

1.3.1) Depois seu colega ver seu código, será muito mais difícil para ele imaginar uma solução original e própria;

1.3.2) O seu colega não entenderá realmente o problema: a compreensão passa pela prática da codificação e não pela imitação/cópia.

1.4) Um colega que tenha visto a sua solução pode eventualmente divulgá-la a outros colegas, deixando você numa situação muito complicada, por tabela;

1.5) O texto acima foi baseado e adaptado da página http://www.ime.usp.br/~mac2166/plagio/, da qual recomendo a leitura completa.

2) Todos os códigos fontes serão comparados por um (ou mais) sistema(s) de detecção de plágio, e os trabalhos com alta similaridade detectada terão suas notas zeradas, tanto aqueles relativos ao código de origem quanto do código copiado. A detecção de plágio será reportada à Seção de Graduação para providências administrativas;

3) A avaliação incluirá a porcentagem de acertos verificada pelo Sistema de Submissão de Trabalhos e também a análise do seu código, incluindo identação, comentários, bom uso da memória e práticas de programação. Portanto faça seu código com cuidado, da melhor forma possível.

Sobre o sistema de submissão:

1. Seu código deverá incluir arquivo fonte .c .h e Makefile – todos os arquivos deverão obrigatoriamente conter no início um comentário com seu nome, número USP, turma e data da entrega do trabalho;

2. A data/hora de entrega do trabalho é aquela estipulada no sistema. Trabalhos entregues por email NÃO serão aceitos, mesmo que dentro da data/hora estipulada. Faça seu trabalho com antecedência

para evitar entregar em cima da hora e ter problemas de submissão;

2.1. A submissão é de responsabilidade do aluno, e os problemas comuns à entrega próxima ao fechamento do sistema também. Portanto: problemas de acesso à rede não serão aceitos como desculpa para entrega por email ou fora do prazo;

3. A compilação e execução do código é feita no sistema pelos comandos:

make all
make run

4. A saída do seu programa deve ser exatamente igual à saída esperada, incluindo: espaços em branco, quebras de linha e precisão decimal;

5. Há um limite em segundos para a execução dos casos de teste e um limite de memória total para ser utilizado. Você deverá gerenciar bem o tempo de execução e o espaço ocupado para que seu programa fique dentro desses limites, para evitar uso excessivo, pois o sistema irá invalidar o caso em que o limite foi excedido;

6. Ao enviar, aguarde sem recarregar a página nem pressionar ESC, para obter a resposta. Caso demore mais do que 2 minutos para dar uma resposta, feche e abra novamente a página do servidor. Verifique se seu programa tem algum problema de entrada de dados (ou se tem algum loop infinito ou parada para pressionamento de tecla). Caso não tenha, aguarde alguns minutos e tente novamente;

7. O erro de “Violação de Memória” significa acesso indevido a arranjo ou arquivo. Use compilação com g e o programa valgrind para tentar detectar a linha de código com erro.

7.1. Exemplo 1 de violação de memória:

```
int **mat = (int **)malloc(sizeof(int *) * 3);
mat[0] = (int *)malloc(sizeof(int)*10);
for (i = 1; i < 3; i++) {
   free(mat[i]);
}
// apenas a posicao 0 de mat foi alocada as outras nao
// portanto esta liberando regiao nao alocada
// gerando violacao de memoria
```

7.2) Exemplo 2 de violação de memória:

```
int B[5] = {5, 6, 7, 8, 9};
int N = 5;
int *A = malloc(N*sizeof(int));
int j = 0, i = 1; // 'j' inicializado em 0, 'i' inicializado em 1
while (j < N){
A[i] = B[j]; // 'j' e 'i' sao indices diferentes
j++;         // quando j = (N1) i = (N1)+1 e
i++;
     // havera escrita indevida em A
}
```