Universidade de São Paulo
Instituto de Ciências Matemáticas e de Computação
Departamento de Ciências da Computação
SCC0201 — Introdução a Ciências da Computação II

Trabalho 02:

Professores: Fernando Pereira dos Santos e Moacir Antonelli Ponti

PAEs: Edresson Casanova (C) e Leo Sampaio Ferraz Ribeiro (D)

Monitores: Gabriel Alves Kuabara (B) e Guilherme Amaral Hiromoto (A)

Desenvolva o trabalho sem olhar o de colegas.
Se precisar de ajuda pergunte, a equipe de apoio está aqui por você.

1 Objetivo do Trabalho

Desenvolver a leitura e tratamento de arquivos de áudio, manipular dados de arquivos, implementação de fórmulas matemáticas e ordenação.

2 Arquivo WAV/WAVE

WAV ou WAVE (Waveform Audio File Format), é um subconjunto da especificação da Microsoft Resource Interchange File Format (RIFF) para armazenamento de arquivos de áudio digital. O formato não aplica qualquer compressão a sequência de bytes e armazena as gravações de áudio com diferentes taxas de amostragem e taxas de bits. Tem sido e é um dos formatos padrão para CDs de áudio. Os arquivos WAVE são maiores em tamanho do que os novos formatos, tais como MP3, que utilizam compressão com perdas para reduzir o tamanho do arquivo, mantendo a mesma qualidade de áudio.

Como ondas sonoras são representadas em formato digital? Basicamente, imaginando ondas sonoras, a amplitude de uma onda num dado momento é convertido em dados binários, representados por valores numéricos, que podemos chamar de "sample". Logo, o conjunto destes valores, são análogos a o formato da onda. Confira a imagem abaixo:

Para quem tiver curiosidade sobre como exatamente funciona um arquivo de áudio do tipo WAV, basta o clique: WaveFormat.

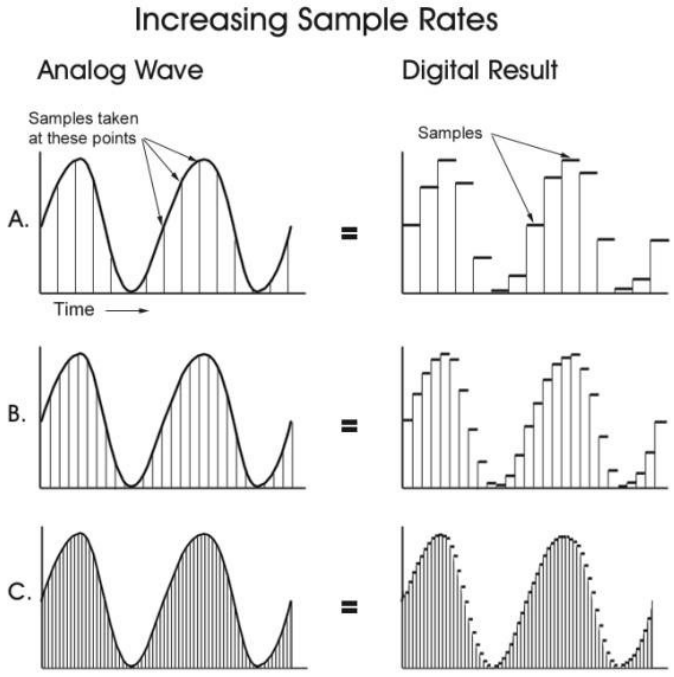
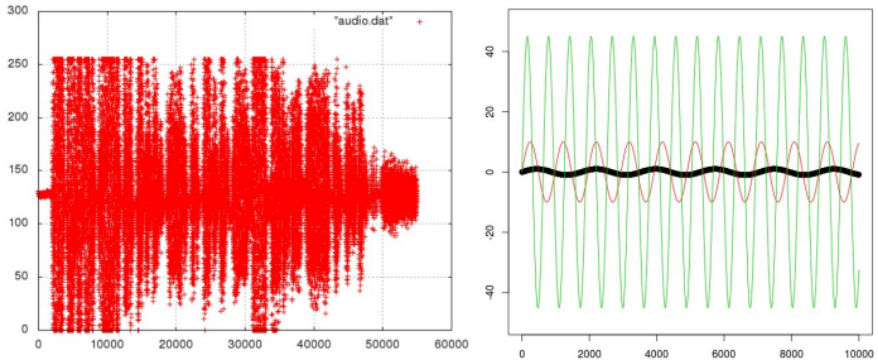


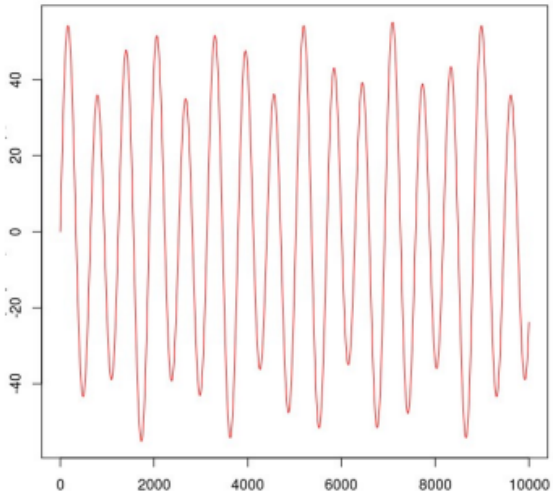
Figure 1: Exemplo de áudio em formato digital

3 Transformada Discreta de Fourier

Uma das ferramentas mais úteis na análise e processamento de sinais como áudio é a Transformada Discreta de Fourier. Essa transformada permite obter um coeficiente (que é um par de valores, ou seja, um valor complexo) para cada sample (amplitude do diafragma do microfone) do sinal de áudio original. Como uma onda sonora pode ser representada por uma combinação de várias ondas senoidais, de forma grosseira, esses coeficientes representam a quantidade de uma determinada onda senoidal presente na onda original, desde aquelas de menor frequência (sons mais graves), até as de maior frequência (sons mais agudos). Para exemplificar, considere o sinal de áudio a seguir.



Observe que ao obter várias senóides podemos, por meio de sua soma, reconstruir o sinal original. Somando as senóides da figura à direita:



Para obter a Transformada Discreta de Fourier de um sinal, usa-se a equação abaixo que será dada em código:

$$C_k = \sum_{n=0}^{N-1} x_n \cdot e^{-i2\pi \frac{k}{N} n}$$

Em que:

- um conjunto de valores, um vetor x_0, x_1, ..., x_{n-1} é aplicado como entrada;
- k é o índice do coeficiente produzido pela transformada, ou seja, se k = 1, então produzimos o primeiro coeficiente C_k e assim sucessivamente. Em geral k = 1, 2, 3, ... N-1;
- N é o número máximo de coeficientes que serão produzidos pela transformada;
- O índice n é utilizado no somatório a fim de produzir um coeficiente C_k;
- O termo e representa a aplicação da função cexp(valor) na linguagem C, ou seja, é o exponencial de certo valor;
- O termo i indica imaginário, ou seja, calculamos a função cexp(valor) para um valor que é dado por um número complexo;

Por consequência do uso do número complexo na exponencial, cada coeficiente C_k produzido por essa transformada é composto por uma parte real e outra imaginária, ou seja, um número complexo (atente-se ao uso das bibliotecas <math.h> e <complex.h>).

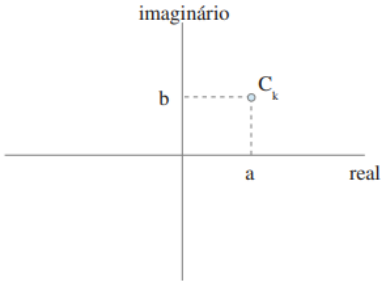


Figure 2: Complex number

A maneira mais simples de compreender um número complexo é considerando um espaço de coordenadas polares na forma:

Após aplicar a transformada sobre os bytes que compõem um áudio, iremos obter um vetor de coeficientes com vários pontos nesse espaço de coordenadas polares. Com isso temos um vetor que cada elemento indica o "quanto de uma certa onda" compõe a onda de áudio original.

4 Magnitude e Ordenação

Cada ponto desses no plano coordenado complexo representa um componente do áudio. Com isso podemos medir a magnitude de cada componente. A magnitude indica a importância desse componente no áudio, ou seja, quanto maior a magnitude, mais relevante esse componente é para o áudio em questão. Então, queremos ordenar de maneira **decrescente** os coeficientes desse vetor usando como método de comparação, a magnitude de cada componente. Assim, saberemos quais os coeficientes mais importantes para ao áudio trabalhado. Para medir a magnitude computamos através da distância Euclidiana:

$$M_k = sqrt(a^2 + b^2)$$

- a é a parte real do número complexo, acessada pela função: creal(complexNumber);
- b é a parte imaginária do número complexo, acessada pela função: cimag(complexNumber);

Após a ordenação, vamos utilizar os T (valor passado na entrada) primeiros valores do vetor no nosso novo áudio. Portanto, iremos zerar os coeficientes nas posições maiores e igual a T. Atente-se que zerar é atribuir zeros, o vetor deve continuar com o mesmo tamanho.

5 Inversa da Transformada Discreta de Fourier

Uma das propriedades importantes de uma TDF é a possibilidade de invertê-la. A teoria da TDF diz que é possível reconstruir o áudio original a partir da representação desse

sinal em somas de funções senoidais, ou seja, conseguimos reconstruir o sinal de áudio a partir da soma de senos e cossenos (representados na equação pela exponencial complexa).

O interessante dessa técnica é que, em geral, o vetor de coeficiente (após a Transformada) são esparsos (possuem muitos valores nulos, ou próximos de zero). Dessa forma, perceba que, se precisarmos transferir o áudio para outro computador, podemos, apenas, transferir os T coeficientes mais importantes e suas posições originais no vetor (ou seja, antes de ordená-lo), além, é claro, do número total de coeficientes. No outro computador, poderíamos recuperar essas informações, criar um vetor para conter os coeficientes mais importantes e aqueles menos importantes que terão valor igual a zero. Em seguida, aplicaríamos a Inversa da Transformada Discreta de Fourier e obteríamos uma aproximação do sinal de áudio.

$$x_n = \frac{1}{N} \sum_{k=0}^{N-1} C_k \cdot e^{i2\pi \frac{k}{N} n}$$

Perceba que assim, transferiríamos uma quantidade muito menor de dados para o computador destino e poderíamos, ainda, reconstituir o áudio e ouvi-lo. Tratando-se assim, de uma **Compressão de Áudio**.

6 Proposta

Considerando um arquivo de áudio gravado em unsigned char, ou seja, cada valor de variação do diafragma do microfone é dado por um inteiro de 8 bits sem sinal, segue os passos:

1) Leia do teclado o nome do arquivo de áudio a ser aberto. Esse arquivo será aberto em modo leitura de arquivo binário.

2) Leia do teclado o número de coeficientes que devem ser utilizados na compressão.

3) Leia todo o conteúdo do arquivo e armazene em um vetor de unsigned char dinamicamente alocado (memória heap), ou seja, cada sample do arquivo de áudio é retratado por um inteiro de 8 bits sem sinal. Uma função será dada para fazer esta leitura dos dados.

4) Em seguida, considere esse vetor de dados/samples como entrada para a Transformada Discreta de Fourier, ou seja, a partir deles será produzido um vetor de coeficientes do tipo double complex dinamicamente alocado.

5) Em seguida ordene o vetor de coeficientes de maneira **decrescente** segundo suas magnitudes, ou seja, o coeficiente de maior magnitude será o primeiro do vetor e o de menor magnitude, o último. Será preciso, no entanto, guardar qual é a posição original de cada coeficiente, pois essa informação será usada nos próximos passos.

6) Atribua zero aos coeficientes das posições igual e maiores do que T. Devem ser mantidos no vetor de coeficientes somente os valores dos T primeiros coeficientes.

7) Em seguida, volte os coeficientes para suas posições originais, ou seja, para as posições que eles tinham antes da ordenação.

8) Utilizando como entrada os coeficientes processados (obtidos no passo anterior), aplique a Transformada Inversa de Fourier. Note que a Inversa poderá gerar valores de ponto flutuante, que deverão, na hora da escrita, serem convertidos para inteiros.

9) Salve o resultado dessa Transformada Inversa em um arquivo binário de áudio em que as amplitudes do diafragma são formadas por 8 bits sem sinal. Não esqueça de copiar para o novo arquivo o cabeçalho do arquivo original, esse corresponde aos primeiros 44 bytes do arquivo.

10) O programa deverá mostrar como saída:

- O número de observações lidas do arquivo de áudio (primeira linha);
- O número de valores menores ou iguais a zero tanto na parte real quanto na parte imaginária no vetor dos coeficientes original (antes da atribuição de 0 para os valores de posições maiores e iguais a T) (segunda linha);
- A sequência ordenada das magnitudes dos T primeiros coeficientes, convertidas para um inteiro (casting para int) (terceira linha em diante).

Para demonstrar uma das aplicações do procedimento realizado, toque o arquivo de entrada e de saída e perceba que, o áudio, mesmo com menos dados, pois cortamos várias frequências, continua com o áudio muito similar ao original.

7 Exemplo de Caso Teste

Entrada:
audio01.wav
6
Saída:
4000
1251
428300
12003
8020
6011
4003
3999

8 Submissão

1. **Crie um header com identificação**. Use um header com o nome, número USP, código do curso e o título do trabalho. Uma penalidade na nota será aplicada se seu código estiver faltando o header.

2. **Comente seu código**. O objetivo é que tenhamos um código claro e que facilite a correção. Exemplos: Se uma variável deixa sua função clara em nome, um comentário sobre ela não é necessário; Um loop triplo que acessa um vetor de matrizes e altera os valores a depender da posição provavelmente pede por um comentário explicando a lógica por trás.

3. **Algoritmo de Ordenação**. De preferência utilize alguma ordenação já aprendida em aula. Pense na complexidade de tempo e espaço de cada uma para que ordene os dados de maneira eficiente.

4. **Organize seu código em funções**. Use funções para deixar cada passo da execução mais clara, mesmo que a função seja chamada apenas uma vez. Comente sua função, descrevendo suas entradas e saídas, sempre que o nome da função e entradas não deixar isso óbvio. Caso quiser, pode fazer em vários arquivos usando um Makefile e um zip para envio.

5. **Utilize Alocação Dinâmica** para armazenar os elementos do vetor e não se esqueça de liberar a memória alocada ao fim da execução.

6. **Compilação** Atente-se a compilação, essa deve incluir a flag -lm para o uso das bibliotecas matemáticas. Por exemplo: gcc -o prog proc.c -lm.

7. **Tire Dúvidas com a Equipe de Apoio**. Se não conseguiu chegar em uma solução, dê um tempo para descansar a cabeça e converse com a equipe de apoio sobre a dificuldade encontrada se precisar.