Resolução – Lista 3 (Projeto e Análise de Algoritmos)

Janeiro - 2023 / Leticia Bossatto Marchezi – 791003

Questão 1

Explique como podemos utilizar a estratégia dividir para conquistar para desenvolver um algoritmo eficiente para o problema do par de pontos mais próximo no plano. Explique porque isso é possível, detalhando os dois resultados teóricos que fundamental esse algoritmo.

Resolução:

Algoritmo com divisão para conquistar

Usa alguns passos:

1) Encontrar o par mais próximo na metade direita

2) Encontrar o par mais próximo na metade esquerda

3) Combinar os pares para verificar se há um par formado pelo ponto P e outro de R

Criando uma lista de pontos(P_x) ordenados pela coordenada *x* em ordem crescente. A mediana dessa lista define uma reta que particiona o plano em região direita (right - R) e esquerda(left - Q)

Teorema 1: O par de pontos pode ser composto por um ponto do lado direito e outro do esquerdo. Se a distância entre 1 ponto à esquerda e um ponto à direita é d, então ambos pontos devem estar no máximo à uma distância d de uma faixa ao redor de L Teorema 2: Se dois pontos estão próximos à faixa entre os planos, eles também estão próximos na lista de coordenadas y.

□

Questão 2

Explique o que é a DFT, explicando para que ela serve e como podemos calculá-la a partir da forma matricial.

Resolução:

DFT é a versão discreta do algoritmo da transformada de Fourier e é utilizada para analisar a frequência de sinais discretos. Sua aplicação permite o processamento de sinais digitais como voz, imagem e áudio ao permitir a decomposição de sinais para outro domínio.

A DFT pode ser calculada a partir da matriz de Vandermonde utilizando as equações:

$$\overrightarrow{X} = W\overrightarrow{x} \tag{2.1}$$

Sendo W a matriz da DFT definida por:

$$W_{k,n} = e^{\frac{-j2\pi kn}{N}} \tag{2.2}$$

para k=0, 1, 2, . . . , N − 1 e para n=0, 1, 2, . . . , N − 1

Sendo k o índice das linhas e n o índice das colunas.

Assim, para um sinal com x amostras (N=x), a matriz será:

$$W = \frac{1}{\sqrt{x}}\begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & w^1 & w^2 & ... & w^{x-1} \\ 1 & w^2 & w^4 & ... & w^{2(x-1)} \\ 1 & w^3 & w^6 & ... & w^{3(x-1)} \\ ... & ... & ... & ... & ... \\ 1 & w^{x-1} & w^{2(x-1)} & ... & w^{(x-1)(x-1)} \end{bmatrix} \tag{2.3}$$

Como a DFT é uma exponencial complexa, ela tem relação direta com senos e cosenos. Dessa forma, é possível obter os coeficientes da matriz ao dividir uma circunferência em x partes, iniciando em w^0 até a seção w^x. E, finalmente, cada elemento é calculado a partir da equação:

$$w = e^{\frac{-j2\pi}{x}} = \frac{\sqrt{2}}{2} - j\frac{\sqrt{2}}{2} \tag{2.4}$$

Lembrando que pela peridiocidade da circunferência vale a regra:

$$w^{x+n} = w^n \tag{2.5}$$

□

Questão 3

Explique como o algoritmo FFT utiliza a estratégia dividir para conquistar para melhorar o desempenho da DFT. Mostre as derivações matemáticas.

Resolução:

A FFT aplica dividir para conquistar na DFT, consistinto em dividir as amostras em 2 partes: as amostras pares, e as amostras ímpares

- Assume-se que $N = 2^m$, caso não seja, pode-se realizar o preenchimento das amostras até a próxima potência com elementos zerados.

Ideia: separe X[n] em índices pares e ímpares Obtem-se duas partes, a primeira expressão da soma é a DFT do sinal formado apenas pelas amostras pares e a segunda expressão da soma é a DFT do sinal formado apenas pelas amostras ímpares, obtendo um problema de complexidade O(nln)

□

Questão 4

Calcule a complexidade do algoritmo FFT e compare com o complexidade da DFT.

Resolução:

$$X[k] = \frac{1}{\sqrt{N}} \sum_{n=0}^{\frac{N}{2}-1} X[2n] W_N^{2kn} + W_N^k \frac{1}{\sqrt{N}} \sum_{n=0}^{\frac{N}{2}-1} X[2n+1] W_N^{k(2n+1)}$$

Análise de complexidade

$$T(n) = 2T(\frac{n}{2}) + O(n)$$

Árvore de complexidade:

cn -> cn

cn/2 cn/2 -> cn

cn/4 cn/4 cn/4 cn/4 -> cn

Em todos os níveis o custo é o mesmo.

O nível máximo(folha da árvore, fim da recursão) é preciso ter $\frac{n}{2^k} = 1$ -> $k = log_2 n$

Assim:

$$T(n) = \sum_{i=0}^{log_2 n} cn = cn(log_2 n + 1) = O(nlogn)$$

Logo, FFT é $O(nlogn)$

Enquanto DFT é $O(n^2)$, ou seja, para aumento da quantidade de amostras FFT é enormemente mais eficiente/rápida.

$$T(n) = \sum_{i=0}^{log_2 n} cn = cn(log_2 n + 1) = O(nlogn)$$

□

Questão 5

Enuncie e demonstre matematicamente o Teorema Mestre (forma reduzida).

$$T(n) = \begin{cases} O(n^d) & a < b^d \\ O(n^d logn) & a = b^d \\ O(n^{log_b a}) & a > b^d \end{cases}$$

Sabemos que:

$$T(n) = a[T(\frac{n}{b^2}) + c\frac{n}{b}^d] + cn^d$$

$$T(n) = a^2[T(\frac{n}{b^2}) + ac\frac{n}{b}^d] + cn^d$$

Seguindo com esse processo até o k-ésimo termo, chega-se na forma geral, dada por:

$$T(n) = a^k T(n/b^k) + a^{k-1}c(n/b^{k-1})^d + a^{k-2}c(n/b^{k-2}) + ... + a^0 c(n/b^0)^d$$

Como n=b$^k, então$n/b$^k = 1$, o que leva a:

$$T(n) = a^k T(1) + \sum_{i=0}^{k-1} a^i c(\frac{n}{b^i})^d$$

Isolando as constantes do somatório:

$$T(n) = a^k T(1) + cn^d \sum_{i=0}^{k-1} (\frac{a}{b^d})^i$$

Estudo do caso 2. $a = b^d$

Nesse caso, o somatório se torna constante.

$$\sum_{i=0}^{k-1} (\frac{a}{b^d})^i = \sum_{i=0}^{k-1} (\frac{b^d}{b^d})^i = k$$

O que implica em:

$T(n) = a^k T(1) + cn^d k$

Como n=b$^k, então$k=log$_b n$

$T(n) = a^k T(1) + cn^d log_b n$

Como a = b$^d então:$

$T(n) = b^{dk} T(1) + cn^d log_b n$

E nesse caso b$^k = n:$

$T(n) = n^d T(1) + cn^d log_b n$

O segundo termo é dominante, ou seja, para a=b^d,O(n$^d log_b n$)

□

Resolução:

$T(n) = aT(\frac{n}{b}) + cn^d$

Então se n=b$^k (potenciadeb) com k > 1, a1, b > 1, c > 0 ed0, temos:$