UNIVERSIDADE FEDERAL DE MINAS GERAIS
ALGORITMOS E ESTRUTURAS DE DADOS III

TP 3
Programação Dinamica

Aluno: Rafael Ramon de Oliveira Egídio
Email: nomar22@gmail.com
Matricula: 2009052360

1-Introdução

Este documento irá detalhar a resolução de um algoritmo , por meio de um programa de computador, do problema de abertura de um cadeado. Este cadeado possui um sistema com sequencia de rodas , cada uma contendo 26 letras do alfabeto , cada mudança de letra para a letra imediatamente posterior ou anterior a letra atual é contado um movimento. É considerado um movimento também quando move-se qualquer subsequencia contígua para a mesma direção.

Este problema resume-se em descobrir o menor número de movimentos que são necessários para a partir de um cadeado com n letras em que todas iniciam-se com a letra a , chegar a uma senha previamente informada.

2-Modelagem e solução proposta

Dado que este é um problema com uma alta complexidade de entendimento, foi definido que neste trabalho prático implementaremos um lógica de combinação de valor dos subtrechos e em seguida tentaremos combinar os valores destes subtrechos afim de encontrar a melhor solução para uma dada entrada.

Características intrísecas do problema

Subestrutra ótima- Este problema possui como subestrutura ótima caminhando juntamente com o tamanho da senha de forma que se temos um menor número de movimentos para n letras , todos os sub-trechos de letras anteriores a esta senha possui o menor número de movimentos para este subtrecho.

Superposição de problemas - Existe uma sobreposição de problemas neste problema de forma que existem subtrechos que podem ser calculados separadamente e podem ser reutilizados.

Dadas as caracterísicas deste problema é possível notar uma oportunidade para a utilização de técnicas de programação dinâmica .

Estruturas de dados

Bloco

Esta estrutura de dados tem função protagonista na execução algoritmo, pois ela armazena o número de movimentos do bloco, a quantidade de movimentos no sentido positivo e a quantidade de movimentos no sentido negativo.

Vetor de armazenamento

Será utilizado como estrutura auxiliar um vetor contendo registros da estrutura Bloco de tamanho 2 ^ n , que servirá para armazenar o número de movimentos necessários para se calcular uma subsequencia. Este vetor utilizará uma abstração do valor binário do indíce para encontrar as letras a quais pertencem uma dada a seguinte sequencia como exemplo:

dce

A partir da entrada dce, será gerado um vetor de 2^n = 8 , onde n é o número de letras contidas na senha, onde a presença de 1 no código binário representa a existencia da letra na subsequencia forma que a indexação das subsequencias que acontece da seguinte forma:

Indíce do Vetor	Ecd Código Binário do número	Comentário
[0]	000	-------
[1]	001	representa a subsequencia d
[2]	010	representa a subsequencia c
[3]	011	representa a subsequencia dc
[4]	100	representa a subsequencia e
[5]	101	-------
[6]	110	representa a subsequencia ce
[7]	111	representa a subsequencia dce

Lista Encadeada Auxiliar

Esta lista encadeada, tem a única função de armazenar a senha corrente.

3 Ambiente de Teste

Este trabalho foi testado em um computador Intel I3 2.4 GHz 4GB com sistema operacional Linux Ubuntu 12.04 32bits.

4 Requisitos

A solução do problema deverá ser implementada na linguagem de programação C. O programa receberá um arquivo de entrada,passado como parametro na chamada do programa e retornará uma saída conforme informada na entrada do programa.

Estrutura do arquivo de entrada:

1 //Número de Instancias

abcxyz //senha1

zzzzbzzzz //senha2

Estrutura do arquivo de saída:

O programa deverá imprimir no arquivo de saída seguindo o modelo:

5 //Número de movimentos senha 1
3 // Número movimentos senha 2

5 Algoritmo da Solução

Para a solução deste problema, usaremos técnicas de programação dinâmica , anteriormente a palavra da senha será armazenada em uma lista encadeada para consulta e auxilio ao vetor com a solução final. Para isto a cada letra da senha lida será inserido um registro na lista encadeada na forma de *Bloco,* que armazena o número de movimentos nescessários para que a partir da letra A se chegue a letra desejada.

Montada a lista encadeada com o bloco correspondente a cada letra, o algoritmo passa por percorrer todo o vetor de soluções e ir calculando o bloco para o indice atual. É importante salientar que todos os registros que possuirem o indíce como potencia de 2 (001,100,010), seus valores serão buscados diretamente da lista auxiliar, pois estes não representam blocos e sim uma única letra .

Para o cálculo daqueles registros que não são potencia de 2, será necessário utilizar uma função de mesclagem das blocos que dado dois blocos calcula o melhorCaso para os dois e retorna um novo bloco com estes dados.

Ainda para o cálculo dos registros não potencia de 2 será necessário calcular todos os sub-blocos que formam o registro atual, para isto basta mesclarmos os blcocos cujas as somas parciais formam o id do registro em questão e as duas parcelas já tenham sido calculadas.

Vamos descrever o passo a passo do comportamento do algoritmo:

DCE

D	Frente → 3 Traz → -23 Valor → 3
C	Frente → 2 Traz → -24 Valor → 2
E	Frente → 4 Traz → -22 Valor → 4

Indíce do Vetor	Ecd Código Binário do número	Algoritmo
[0]	000	AFD - ERRO
[1]	001	Pega diretamente da lista
[2]	010	Pega diretamente da lista
[3]	011	Mescla os blocos (1,2)
[4]	100	Pega diretamente da lista
[5]	101	AFD - FALSE
[6]	110	Mescla os blocos (2 , 4)
[7]	111	Pega o melhor caso entre as mesclagens já calculadas de (3,4) e (6,1)

Existe também um AFD que define se um número deve ser calculado, este AFD serve para identificar se o id representa um número inteiro no qual represente uma sequencia contígua do tipo 0110, 011 e negar sequencias do tipo 0101 1011 01101.

Principais Estruturas:

```
typedef struct bloco{
	int tamPalavra;
	int valor;
	int pos;
	int neg;
	int id;
	char letra;

}Bloco;

typedef struct {
  LinkCelula primeiro;
  LinkCelula ultimo;
  int len;
}Lista;
```

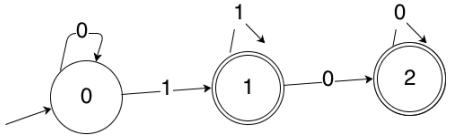
Lista encadeada comum, na estrutura célula existirá um bloco.

7 Pseudo-códigos e descrições de funções

LinkBloco* **criaVetor**(int i): Cria um vetor de blocos de tamanho i, no algoritmo principal este i será de 2^n. O(1)

Boolean **letrasSeparadas**(int inteiro): Trata-se da implementação do seguinte AFD

O(logn)



Boolean **estaNalista**(int inteiro): Verifica se o número é uma potencia de 2 , ou

seja estará na lista os números cujo binário sejam 010, 001, 100; O(1)

`LinkBloco` **`melhorCaso`**`(int id, LinkBloco *vetor)`: Tem a função de calcular o melhor caso para um id e para isto utiliza principalmente a função mesclaBlocos comparando cada soma parcial que forma i e escolhe aquela que leva ao melhor caso. O(n)

`LinkBloco` **`mesclaBlocos`**`(LinkBloco esq, LinkBloco dir)`: Função crítica deste algoritmo , tem a função de dados 2 blocos utilizar uma métrica para mesclar 2 blocos e retornar o produto desta mesclagem. Nesta abordagem utilizamos a seguinte estratégia, soma-se o valor de cada um dos blocos e retorna o menor valor entre esta soma, o maior dos sentidos positivos entre os blocos ou o maior dos sentidos negativos em módulo. O(1)

8 Análise de Espaço e Complexidade

A solução deste problema possui o loop no vetor de tamanho 2^n , em cada laço deste vetor é feito todas as somas parciais o que gera uma iteração de n/2 , com isto a complexidade deste algoritmo é exponencial com 2^n , com n sendo o tamanho da palavra.

A complexi

9 Algoritmo principal

Para número de instancias

Le arquivo de entrada

Monta Lista auxiliar

cria Vetor tamanho 2^n

itera i em Vetor

se Registro atual for bloco contiguo

se o número for uma potencia de 2

pega da lista auxiliar

senão

mescla blocos que formam a soma do valor de i

senão

letras separadas não devem ser calculadas , fazer nada

Escrever no arquivo de saída

10 Execução

Para a compilar este trabalho deve-se executar o comando make de dentro do diretório raiz do projeto em seguida os comandos para heurístico e ótimo respectivamente:

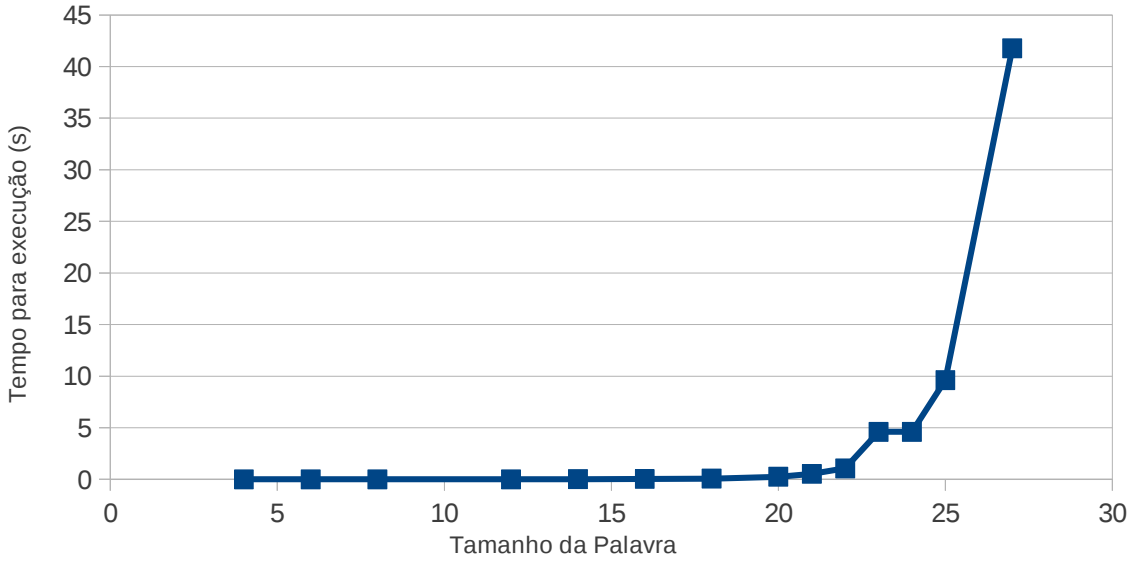
./tp3 [arquivodeentrada.txt} [arquivodesaida.txt]

O arquivo de entrada deverá seguir a especificação dada em tópico anterior e a saída do

programa será impressa no arquivo de saida desde que o mesmo possua permissão de escrita.

11 Avaliação Experimental

Para a realização dos testes foram utilizadas entradas para diversas instancias e tamanho da palavra limitado por 30 letras, que é o tamanho máximo aceito pelo tipo de dados long int.



12 Conclusão

Foram encontrados diversos problemas no desenvolvimento deste trabalho prático , dos quais podemos pricipalmente a dificuldade no entendimento do problema por completo, acredito que os resultados que não conferirem com o gabarito neste trabalho prático se devem especialmente a função mesclar blocos, que devido ao calendário apertado e a demora no entendimento do trabalho prático, acabou comprometida. Contudo com este trabalho prático foi especialmente importante no meu modo de ver, por ter apresentado um problema realmente desafiador que por apesar de não ter conseguido um total exito para a atividade proposta, foi de grande engrandecimento.

Referências

Ziviani, Nivio (2011) “Projeto de Algoritmos com implementações em Pascal e C”, Cengage Learning, 3ª Edição.

http://www.geeksforgeeks.org/