Interface gráfica do usuário, Aplicativo

Descrição gerada automaticamente

Ciência da Computação

Projeto e Análise de Algoritmos III

Documentação Trabalho Prático tp-2

Vitor Ferreira França

2024

1-Introdução

O problema a ser solucionado é encontrar a máxima pontuação possível no jogo onde. Dado uma sequência de números inteiros, O jogador pode fazer várias jogadas. Em cada jogada ele pode escolher um elemento da sequência (vamos denotá-lo ak ) e excluí-lo, sendo que os elementos ak+1 e ak−1 também devem ser excluídos da sequência. Essa jogada traz ak pontos para o jogador.

**Exemplo:**

Sequancia:“3 ,6 ,7 ,5 ,3”

-É escolhido o primeiro item “3” a1

Sequência após “7,5,3”

Pontuação =3

-É escolhido o último item “3” ak

Sequência após “7”

Pontuação =6

-É escolhido o item “7” ak

Sequencia após “ ”

Pontuação =13

Esse é a melhor pontuação possível

13

2- Construção e funcionamento do Algoritmo

## **2.1- Chegada de dados pelas linhas de comandos**

É passado primeiramente a estratégia que será usada (“D” ou “A”), logo após, qual arquivo de input deverá ser utilizado, exemplo: ./tp2 d input0.txt .

## **2.2- Variáveis de entrada**

O arquivo “.txt” informara o tamanho do vetor e seus itens , desta forma

“A primeira linha contém o inteiro N (0 ≤ N ≤ 105 ) que mostra quantos números existem na sequência de Joâo. A segunda linha contém N inteiros a1, a2, ..., an.

Exemplo de entrada:

9

1 2 1 3 2 2 2 2 3 ”

**2.3- Funções de abrir o arquivo para leitura e escrita no output**

**-A função para abertura de um arquivo** , abre o mesmo utilizando o nome inserido pelo terminal ,por meio da construção de um caminho de onde estará localizado o input , após isso o caminho é aberto como uma “FILE” e retornada como um ponteiro para ser utilizada.

-**A função de escrita no output** recebe a resposta para que seja mandado para o “output.txt” ,após isto é construído o caminho para o arquivo para escrita e

Se caso ocorra um erro na abertura é printado o erro no terminal, após isso é fechado o arquivo.

## **2.3- Construindo a TAD que vai armazenar a sequência e suas funções**

Foi utilizado um vetor dinâmico para tal, pela sua estrutura como vetor e sua dinamicidade para adicionar e retirar itens. Ela contem desta forma o array que armazena os itens ,seu tamanho e capacidade total do mesmo.

**-Primeiramente é usado a função para criar o array dinâmico** que aloca a memória inicial para o vetor com sua capacidade tamanho, e é retornado.

**-A Função de adicionar** primeiramente checa se é preciso redimensionar o vetor, logo após é adicionado o item na ultima posição do vetor .

**-A função de remover um elemento** primeiramente verifica se o índice está dentro dos limites válidos do array, após isto ela reposiciona todos os itens do vetor e e diminui seu tamanho.

-**A função de printar o array** feito para testes durante o desenvolvimento do software, recebe e percorre o vetor printando o que está nele “displayArray”.

-**A função para deletar** o item escolhido e os seus vizinhos do lado percorre faz uma pesquisa no vetor para achar o item desejado, e deletar ele os ao seu lado no vetor

## **2.4- Forma dinâmica da resolução do problema**

**-A função “biger”** é uma função auxiliar que retorna o valor máximo entre dois números. Ela recebe dois números inteiros como entrada e retorna o maior deles.

**-A função “Dinamica”** é a função principal que calcula a pontuação máxima. Ela recebe um array de inteiros representando a sequência de números e o tamanho dessa sequência “size”.

Se o tamanho da sequência for zero (size == 0), isso significa que não há números na sequência, então a função retorna 0, pois não há pontuação a ser obtida.

A função declara duas variáveis inteiras in e ex, que representam a pontuação incluindo e excluindo o número atualmente considerado, respectivamente. Inicialmente, ambas são definidas como 0.

Em seguida, a função entra em um loop que itera sobre cada número na sequência, de 0 até size - 1.

Para cada número na sequência, a função atualiza as variáveis in e ex. A ideia aqui é simular duas situações: incluir o número atual e excluir o número atual.

Dentro do loop, a função armazena temporariamente o valor atual de inc em uma variável temporária temp.

Em seguida, atualiza o valor de in para a soma do valor atual de ex e o número atual da sequência (vet[i]). Isso simula a situação de incluir o número atual na pontuação.Depois, atualiza o valor de ex para o máximo entre o valor anterior de in (armazenado em temp) e o valor anterior de ex. Isso simula a situação de excluir o número atual da pontuação.

Após o término do loop, a função retorna o máximo entre in e ex, representando a pontuação máxima que João pode obter.

Pseudocódigo:

-----------------------------------------------------------------------------------------------

Função Dinamica(vetor, tamanho):

Se tamanho for igual a 0:

Retorne 0 // Não há números para calcular a pontuação

pontuação\_incluindo = 0

pontuação\_excluindo = 0

Para cada elemento i no vetor:

temp = pontuação\_incluindo

pontuação\_incluindo = pontuação\_excluindo + vetor[i]

pontuação\_excluindo = Maior(temp, pontuação\_excluindo)

Retorne Maior(pontuação\_incluindo, pontuação\_excluindo)

## **2.5- Forma alternativa da resolução do problema**

3- Análise de tempo dos Algoritmo

## **3.1- Tempos forma dinâmica da resolução do problema**

## Resumo:

|  |  |  |  |
| --- | --- | --- | --- |
| Tamanho sequência: | Tempo total decorrido: | Tempo de CPU | Tempo de usuário: |
| 5 | 0.001715 | 0.000000 | 0.000000 |
| 9 | 0.000790 | 0.000000 | 0.0000000 |
| 50 | 0.000946 | 0.000000 | 0.000000 |
| 100 | 0.001102 | 0.000000 | 0.000000 |
| 500 | 0.001402 | 0.000000 | 0.000000 |
| 1000 | 0.000992 | 0.000000 | 0.000000 |
| 5000 | 0.001363 | 0.000000 | 0.000000 |
| 10000 | 0.001235 | 0.000000 | 0.000000 |
| 50000 | 0.002632 | 0.000000 | 0.000000 |
| 100000 | 0.001598 | 0.000000 | 0.000000 |

Gráfico, Gráfico de linhas

Descrição gerada automaticamente

4- Análise de Complexidade dos Algoritmo

## **4.1- Complexidade forma dinâmica da resolução do problema**

-A função biger(int a, int b) compara dois números e retorna o maior. Esta função tem uma complexidade de tempo constante, ou seja, O(1), porque realiza um número fixo de operações.

-A verificação inicial if (size == 0) também tem uma complexidade de tempo constante, O(1).A inicialização das variáveis in e ex é outra operação de tempo constante, O(1). O loop for (int i = 0; i < size; i++) itera sobre cada elemento do vetor. Dentro do loop, todas as operações (atribuição de variáveis e chamadas à função biger) são operações de tempo constante. Portanto, a complexidade de tempo do loop é proporcional ao tamanho do vetor, ou seja, O(n), onde n é o tamanho do vetor. Portanto, a complexidade total do tempo do algoritmo é dominada pelo loop for, resultando em uma complexidade de tempo linear, O(n).