Universidade Federal de Minas Gerais
Sistemas de Informação
2019/2

Daniel Pires Quirino

Trabalho Prático de Matemática Discreta

Belo Horizonte - MG

2019

Sumário:

Introdução:

Esse trabalho prático consiste, primeiramente, na análise de um vetor com n números inteiros, sendo n um número inteiro positivo, determinando a soma máxima encontrada em um sub-vetor contínuo desse vetor.

Posteriormente será tratado sobre o problema dos quadrados mágicos para tamanho de lados iguais a 4,5 e 6. Esse problema consiste em criar um código que faça com que de acordo com um valor de entrada entre 3 - 6 seja criada um quadrado mágico que possui como característica o fato de que a soma de todos os lados e diagonais possuem como resultado um mesmo valor.

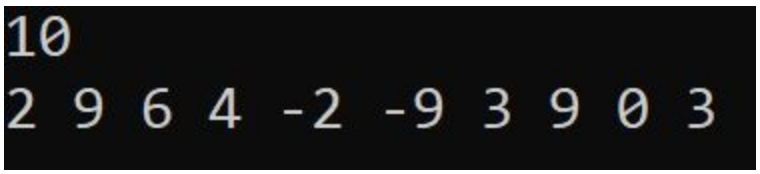
A partir dessa premissa, foi desenvolvido um software na linguagem C.

Soma Máxima:

Descrição do formato de entrada dos dados:

O formato de entrada deve ser:

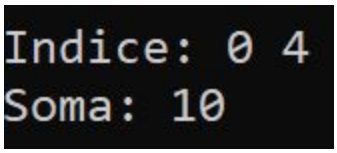
- Primeiramente o usuário deve digitar um número positivo e inteiro, que será o tamanho do vetor a ser analisado
- O segundo parâmetro deve ser os valores que o vetor assumo da posição 0 até a posição (n-1), sendo n o tamanho do vetor.
- Obs: Deve possui uma quebra de linha entre os dois parâmetros:



Exemplo de parâmetro de entrada.

Descrição do formato de saída dos dados:

O formato de saída consiste no índice, que mostra o intervalo das posições em que se têm a maior soma possível, e soma, que determina a maior soma possível do vetor



Exemplo de saída do programa

Descrição geral do programa:

O funcionamento geral deste software funciona da seguinte maneira.

Primeiramente, é obtido o número n (tamanho do vetor digitado pelo usuário), gerando um vetor de tamanho n, a partir de alocação dinâmica de memória. Esse vetor é percorrido, usando um algoritmo guloso, para que se possa determinar a maior soma e o maior índice. Posteriormente, esse algoritmo guloso é usado outra vez, com algumas modificações para que se possa encontrar o índice de menor valor que compõe a maior soma.

A ideia geral é a de guardar em uma variável a melhor soma de elementos a frente que termina exatamente no elemento em que estamos. Depois, percorrer a sequência. Assim, quando chego no elemento maior que guarda o melhor resultado para a casa anterior, posso usá-lo para calcular a maior soma que termina no elemento em que estamos, sendo este, portanto, o novo maior valor. Se isso superar o maior valor dentre todos os que já tivemos, ele será a maior soma.

Funções utilizadas:

Com intenção de facilitar o entendimento do código e seguindo as boas práticas de programação, além da função principal main(), foram criadas algumas outras funções auxiliares que serão listadas abaixo:

- max(): retorna o maior valor entre dois números

- getNumbersWithSpace(): obtêm os índices do vetor digitado como parâmetro de entrada pelo usuário

- getLowerIndex(): Obtêm o menor índice da maior soma do vetor

- maxSum(): retorna o valor da maior soma do vetor

- Foram utilizados também algumas funções já prontas como o ‘atoi()’, ‘malloc()’ e ‘strtok()’

Bibliotecas utilizadas:

Foram utilizadas as seguintes bibliotecas nesse software:

- stdio.h
- stdlib.h
- string.h
- conio.h

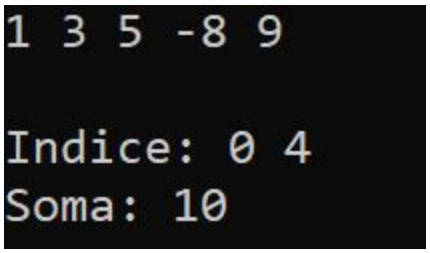
Análise Assintótica da solução realizada:

A complexidade desse algoritmo é dado por O(n).

Isso se deve ao fato de que a maior iteração que existe nesse algoritmo é um for que percorre as posições do vetor.

Testes realizados:

Geração de alguns testes realizados neste programa para ratificar a sua eficácia e correto funcionamento.



Exemplo (1)

```
10
1 4 -6 7 -3 8 9 -1 2 4

Indice: 3 9
Soma: 26
```

Exemplo(2)

```
6
1 2 3 4 5 6

Indice: 0 5
Soma: 21
```

Exemplo(3)

Quadrado Mágico:

Descrição do formato de entrada dos dados:

O formato de entrada deve ser:

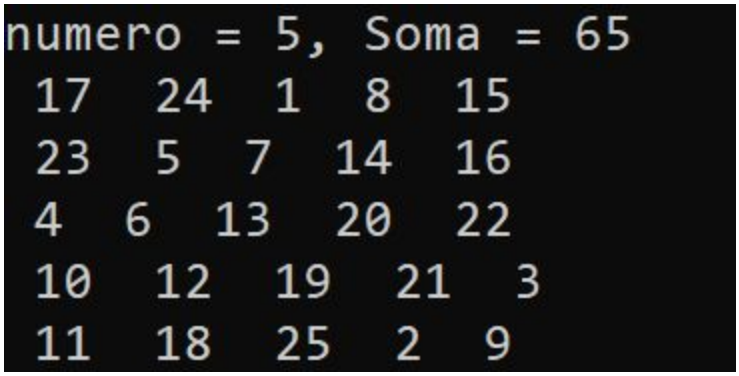
- O usuário deve digitar o tamanho do lado do quadrado mágico, podendo, portanto, variar entre 4 - 6
-

```
5
```

Exemplo de parâmetro de entrada.

Descrição do formato de saída dos dados:

O formato de saída consiste na impressão do quadro mágico com os valores preenchidos em cada posição:



Exemplo de saída do programa

Descrição geral do programa:

O funcionamento geral deste software funciona da seguinte maneira.

Primeiramente, é obtido o número n (lados do quadrado mágico), gerando, a partir de alocação de memória, uma matriz de tamanho n x n.

Para as matrizes com tamanho de número ímpar, é utilizado a lógica de primeiro preencher a linha de índice 0 e a coluna de índice (n-1)/2 com o valor 1. Posteriormente era segundo o seguinte algoritmo, caminhava para cima e para a direita do quadro, se o campo estivesse vazio, preenchia com o valor anterior mais 1. Porém, existem algumas regras para quando, por exemplo, o campo já possuir valor, nesse caso deve-se preencher o valor a posição anterior porém com a linha decrescido de uma unidade. Além disso caso os índices ultrapassem a região delimitada pelo quadrado, a posição deve ser a oposta.

Para as matrizes de tamanho 4 o procedimento a ser seguido deve ser o seguinte:
Em cada um dos cantos do quadrado mágico, marcar um mini-quadrado com lados de comprimento n/4. Posteriormente, marcar as casas no centro do quadrado em uma área quadrada de n/2. Assim, basta agora preencher os números do quadrado mágico da esquerda para a direita, mas enumere apenas se estiver na situação dos cantos preenchidos. Finalmente, preencher o resto do quadrado mágico, porém, de forma regressiva

Para as matrizes de tamanho 6 deve-se dividir o quadrado mágico em quatro quadrantes iguais. Atribua para cada quadrante um limite de números. Resolva cada quadrante utilizando o método dos quadrantes mágicos ímpares.
Fazer algumas substituições.

Funções utilizadas:

Com intenção de facilitar o entendimento do código e seguindo as boas práticas de programação, além da função principal main(), foram criadas algumas outras funções auxiliares que serão listadas abaixo:

- alocarMatriz(): faz a alocação dinâmica da matriz.
- fillMatrix(): função generica para preencher todas as matrizes de tamanho entre 4-6.
- fillOddSizeMagicBoard(): preencher funções ímpares.
- fillEvenMagicSquareSimple(): preencher função de lado igual a 6.
- fillEvenMagicSquare(): preencher função de lado 4.
- soma(): função para calcular o valor da soma de qualquer lado(diagonal), do quadrado mágico

Bibliotecas utilizadas:

Foram utilizadas as seguintes bibliotecas nesse software:

- stdio.h
- Stdlib.h

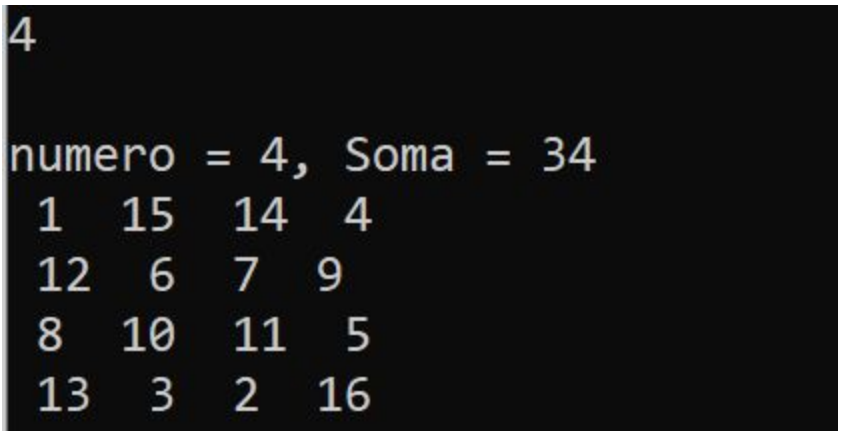
Análise Assintótica da solução realizada:

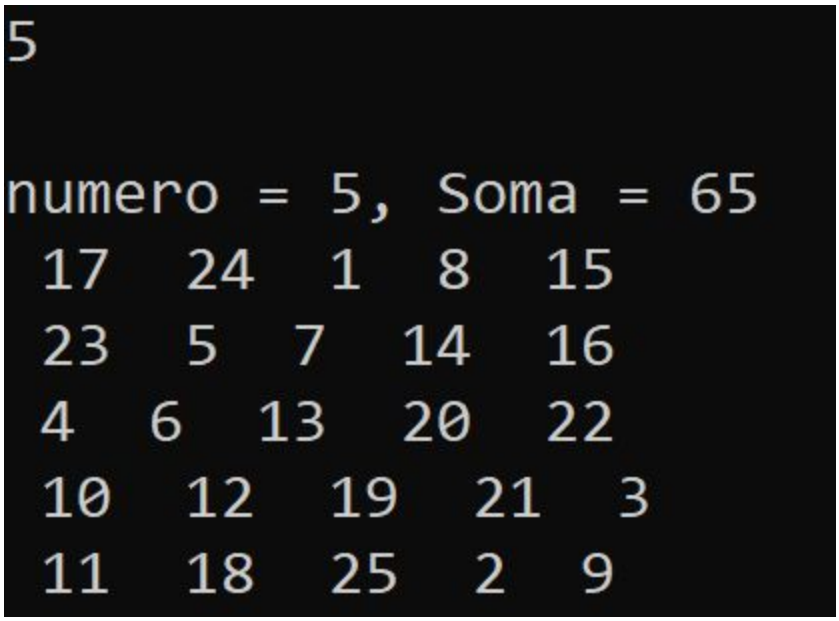
A complexidade desse algoritmo é dado por O(n^2).

Isso se deve ao fato de que a maior iteração que existe nesse algoritmo for aninhado com outro for.

Testes realizados:

Geração de alguns testes realizados neste programa para ratificar a sua eficácia e correto funcionamento.





```
6

numero = 6, Soma = 111
 35  1  6  26  19  24
 3  32  7  21  23  25
 31  9  2  22  27  20
 8  28  33  17  10  15
 30  5  34  12  14  16
 4  36  29  13  18  11
```