

1. Crie um Tipo Abstrato de Dados para Matrizes. Implemente as funções de inicializar a matriz (toda zerada), computar a soma de duas matrizes, subtração de duas matrizes e o determinante de uma matriz quadrada. Seu programa pode indicar um erro caso a matriz passada para o determinante não seja quadrada.

a. Indique a complexidade das funções acima

2. Escreva um algoritmo que determina se um número inteiro é primo. Qual é a complexidade do seu algoritmo?
3. Escreva um algoritmo que compacta uma string. A compactação de uma string é uma operação que converte uma string como:

“aaaaabcdddeeeffffff”

Para

“a5b1c2d3e3f6”

Escreva uma função função que faz a transformação inversa.

Qual a complexidade das 2 funções?

4. Implemente uma função com a seguinte assinatura:

int existeSoma(int *values, int n, int target)

ou com vetores

int existeSoma(int values[], int n, int target)

Tal função deve indicar se no vetor de values existem dois elementos cuja soma

target. Por exemplo, no vetor [10, 20, 3, 45, 0], caso target=65, o retorno é 1 (45+20=65). Porém caso passe target=29 o retorno é 0.

Qual a complexidade da sua função?

5. Escreva uma função que inverte as palavras de uma string:
“Alice Likes Bob”
vira

“Bob Likes Alice”

Qual a complexidade da sua função?

6. Escreva uma função para encontrar o local de início de fim de uma string dentro de outra string. A assinatura da função é:

```
void subPosition(char *text, char *sub, int *start, int *end)
```

Por exemplo, para uma entrada:

```
char *sub = “muito”  
char *texto = “Eu gosto muito de AEDS2”
```

Seu código deve armazenar start=9 e end=13

Qual a complexidade da sua função?

7. Para cada uma das afirmativas abaixo, diga se a afirmativa é verdadeira (V) ou falsa (F). Em todas as afirmativas, justifique a sua resposta. Respostas sem justificativa não serão consideradas.

- a. () Considere um programa P que faz uma série de operações de custo constante, chama uma função F1 com complexidade dada por $f(n)$ e depois chama uma função F2 com complexidade dada por $g(n)$, onde $g(n) = 1000 \cdot f(n)$. Pode-se afirmar que o programa P é $O(f(n))$.
- b. () Considere um programa cuja função de complexidade é $f(n) = 3\log(n)$. É correto afirmar que esse programa é $O(\log n)$, mas não é $O(n * n)$.
- c. () Um programa P executa uma função F1 com complexidade $f(n)$ em 50% de suas n interações, e uma função F2 com complexidade $g(n)$ nas demais interações. Portanto, o programa P tem complexidade $O(\max(O(f(n)), O(g(n))))$.
- d. () Sejam duas funções de complexidade $g(n) = 5n^2 + 3n + 4$ e $f(n) = 95n^2 + n + 15$. É correto afirmar que um programa P1 cuja complexidade é $g(n)$ é mais rápido que um programa P2, com complexidade $f(n)$.

8. Considerando que $0 < \varepsilon < 1 < c$, indique para cada par de expressões (A, B) na tabela abaixo, se A é O , Ω , ou Θ de B. Justifique suas respostas.

	A	B	O	Ω	Θ
(i)	$n^4 + 100n^3$	$n^3 + 100n^4$			
(ii)	2^n	$3^{(n/2)}$			

(iii)	c^ε	$(c+1)^\varepsilon$			
(iv)	$\log n$	$\sqrt{\log n}$			