Fazer analise assintótica dos seguintes algoritmos

a. Vetor x vetor

```c
void somaVetor(int *va, int *vb, int *resultado, int n) {
    for (int i = 0; i < n; i++) {
        resultado[i] = va[i] + vb[i];  // O(n)
    }
}
```

Big O = O(n)

b. Vetor x matriz

```c
void vetorMatriz(int *vetor, int **matriz, int *resultado, int n, int m) {
    for (int i = 0; i < n; i++) {              // Percorre as n linhas da matriz
        resultado[i] = 0;
        for (int j = 0; j < m; j++) {          // Percorre as m colunas
            resultado[i] += vetor[j] * matriz[i][j];  // O(n * m)
        }
    }
}
```

Big O = O(n)^2

c. Matriz x matriz

```c
void matrizMatriz(int **A, int **B, int **resultado, int n, int m, int p) {
    for (int i = 0; i < n; i++) {  // Percorre as n linhas da matriz A
        for (int j = 0; j < p; j++) {  // Percorre as p colunas da matriz B
            resultado[i][j] = 0;
            for (int k = 0; k < m; k++) {  // Percorre as m colunas de A (e as m linhas de B)
                resultado[i][j] += A[i][k] * B[k][j];  // O(n * m * p)
            }
        }
    }
}
```

Big O = O(n)^3

Aplicação das Propriedades do Big-O

i) Constantes e Big-O

Se KKK é uma constante, temos a seguinte propriedade:

K·O(f(N))=O(f(N))K \cdot O(f(N)) = O(f(N))K·O(f(N))=O(f(N))

Isso significa que, mesmo que aumentemos as instruções de um algoritmo multiplicando-as por uma constante KKK, a ordem de complexidade assintótica não muda. Ou seja, a maior ordem de crescimento permanece a mesma.

Exemplos:

- **Vetor x Vetor**: Se a complexidade é O(n)O(n)O(n), então K·O(n)=O(n)K \cdot O(n) = O(n)K·O(n)=O(n). A constante KKK não altera o comportamento assintótico do algoritmo.
- **Vetor x Matriz**: Se a complexidade é O(n2)O(n^2)O(n2), a constante KKK também não altera a complexidade assintótica, mantendo O(n2)O(n^2)O(n2).
- **Matriz x Matriz**: Se a complexidade é O(n3)O(n^3)O(n3), a ordem de crescimento continua sendo O(n3)O(n^3)O(n3), independentemente de multiplicadores constantes.

ii) Soma de Complexidades

Quando somamos duas funções com a mesma ordem de crescimento assintótico, a maior ordem domina:

O(f(N))+O(f(N))=O(f(N))O(f(N)) + O(f(N)) = O(f(N))O(f(N))+O(f(N))=O(f(N))

Ou seja, se um algoritmo executa dois passos com a mesma complexidade assintótica O(f(N))O(f(N))O(f(N)), a soma dessas complexidades ainda terá a mesma ordem, sem mudança no comportamento assintótico.

Exemplo:

- **Matriz x Matriz**: Se o algoritmo para multiplicação de matrizes tem uma complexidade de O(n3)O(n^3)O(n3) e executa duas vezes, a soma das complexidades seria O(n3)+O(n3)O(n^3) + O(n^3)O(n3)+O(n3), mas o comportamento assintótico final permanece O(n3)O(n^3)O(n3), pois o termo dominante é o de maior ordem.