EXERCÍCIOS PROPOSTOS – AULA 03

Felipe Fazio da Costa; RA: 23.00055-4

1. Algoritmo: Cálculo do Fatorial de n

Código Java:

```
import java.util.Scanner;

public class ex_1 {
   public static void main(String[] args) {
      Scanner input = new Scanner(System.in);

      // Inicialização de variáveis
      int resultado = 1;
      int a = 0;

      System.out.println("Enter the value of a factorial: ");
      a = input.nextInt();

      // Loop para calcular o fatorial
      for(int i = a; i > 0; i--){
         resultado = resultado * i;
      }

      System.out.println("The result is equal to: " + resultado);
      input.close();
   }
}
```

Complexidade Computacional:

O algoritmo executa um loop que itera de `n` até `1`, realizando uma multiplicação em cada iteração. Portanto, a função de complexidade é:

int resultado = 1: 2 operações
int a = 0: 2 operações
T(n) = O(n)

int i = a: 2 operações

i > 0: 3 * (n + 1)

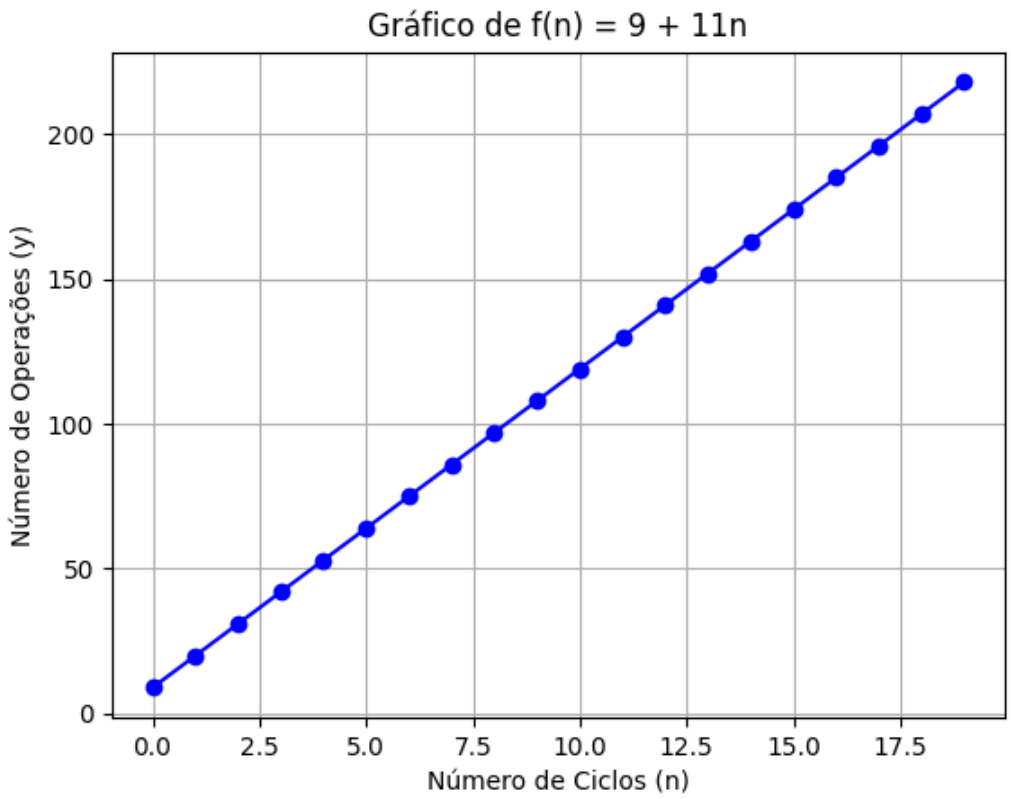
i--: 4 * n

resultado = resultado * i: 4 *n

Resultado: f(n) = 9 + 11*n.

Cálculo e Resultados:

Exibindo gráfico:



2. Algoritmo: Cálculo da Média Total dos elementos de um vetor de tamanho n

Código Java:

```
import java.util.Scanner;

public class ex_2 {
   public static void main(String[] args) {
      Scanner input = new Scanner(System.in);

      // Inicialização de variáveis
      int[] vetor = {1, 2, 3, 4, 5, 6, 7, 8, 9};
```

```
        float media = 0;
        float resultado = 0; // Inicializando a variável resultado

        // Calcula a soma dos elementos do vetor
        for(int i = 0; i < vetor.length; i++){
           resultado += vetor[i];
        }
        // Calcula a média
        media = resultado / vetor.length;

        System.out.println("The result is equal to: " + media);
        input.close();
    }
}
```

Complexidade Computacional:

O loop percorre todo o vetor de tamanho `n`, realizando uma soma em cada iteração. Assim, a função de complexidade é:

int[] vetor = {1, 2, 3, 4, 5, 6, 7, 8, 9}: 1 + n operações

float media = 0: 2 operações

float resultado = 0: 2 operações

int i = 0: 2 operações

i < vetor.length: 3*(n + 1) operações

i++: 4*n operações

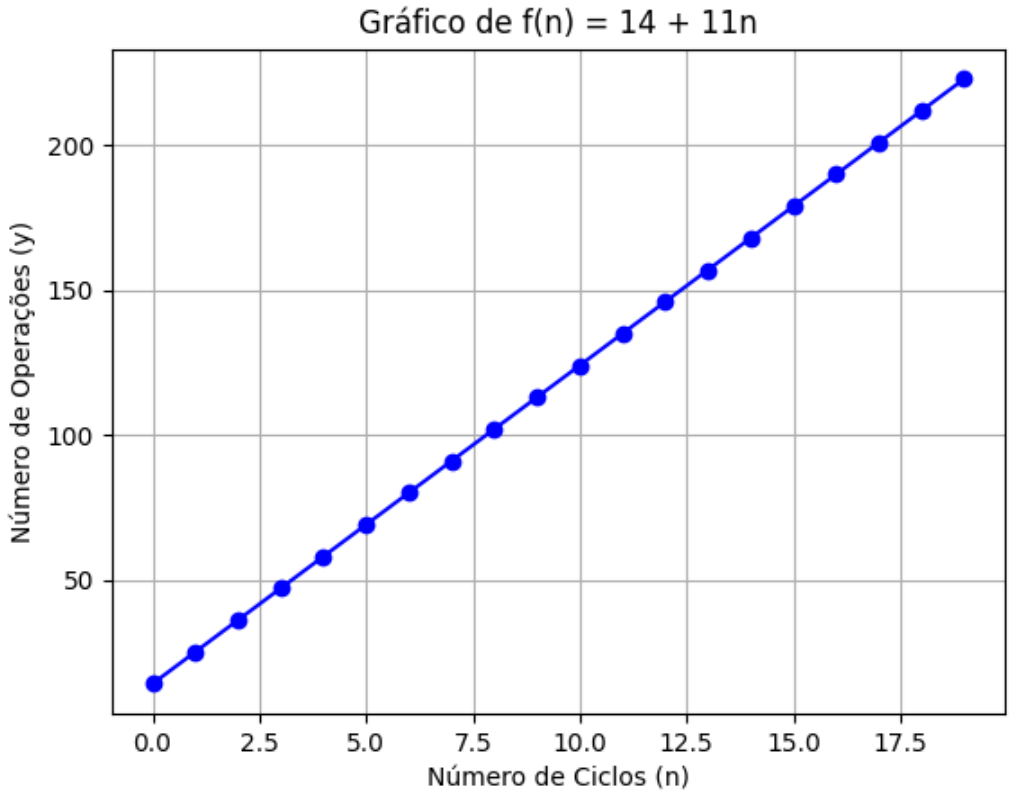
resultado += vetor[i]: 7*n operações

media = resultado / vetor.length: 4 operações

Resultado: h(n) = 14 + 11*n

Cálculo e Resultados:

Exibindo gráfico:



Conclusão:

- Ambos os algoritmos apresentam complexidade linear O(n)$O(n)$, o que é eficiente para problemas de pequeno a médio porte. No entanto, para entradas muito grandes, algoritmos com complexidade menor (como O(log n)$O(\log n)$ ou O(1)$O(1)$) seriam preferíveis.
- A análise de complexidade é essencial para entender o desempenho de um algoritmo e prever como ele se comportará com diferentes tamanhos de entrada.
- A visualização gráfica das funções de complexidade (como mostrado nos gráficos) ajuda a compreender o crescimento do número de operações em relação ao tamanho da entrada.

Em resumo, os exercícios reforçam a importância de analisar a complexidade computacional dos algoritmos, permitindo escolher a melhor solução para um problema com base no desempenho esperado.