

EXERCÍCIOS PROPOSTOS – AULA 05

Resolução: Individual;

Prazo: Até o início da próxima aula;

Entrega: Relatório, em PDF, contendo, obrigatoriamente: os códigos em Java experimentados; os cálculos elaborados; e os resultados obtidos;

Instruções: Efetue a **análise assintótica de funções** abaixo, utilizando os conceitos apresentados na respectiva aula.

Exercícios: De 1 à 12, a seguir:

1. Implementar, em Java, o algoritmo ***Insertion-Sort***;
2. Qual é a **Ordem de Complexidade** desse **Algoritmo**, considerando o pior caso? Não é necessário desenvolver a **Função de Complexidade**, deve-se, apenas, apresentar a **Ordem de Complexidade do Algoritmo**;

Dica: Estudar a implementação no capítulo 2 do livro do *Cormen*

3. No algoritmo a seguir, informe a quantidade de vezes que a **Linha 1** será executada, em tempo de execução e em função de n .

```
import java.util.Scanner;

public class TarefaT3_01 {
    public static void main(String[] args) {
        Scanner in = new Scanner (System.in);
        int n = in.nextInt();
        System.out.println(Func(n));
        in.close();
    }
    public static int Func(int n) {
        int m = 0;
        for (int i=1; i <= n; i++)
            for (int j = 1; j <= n; j++ ) {
                m = m + 1; // Linha 1
            }
        return m;
    }
}
```

4. No algoritmo a seguir, informe a quantidade de vezes que a **Linha 1** será executada, em tempo de execução e em função de n .

```
import java.util.Scanner;

public class TarefaT3_02 {
    public static void main(String[] args) {
        Scanner in = new Scanner (System.in);
        int n = in.nextInt();
        System.out.println(Func(n));
        in.close();
    }
    public static int Func(int n) {
        int m = 0;
        for (int i=2; i < n; i++)
            for (int j = 2; j < n; j++ ) {
                m = m + 1;           // Linha 1
            }
        return m;
    }
}
```

5. No algoritmo a seguir, informe a quantidade de vezes que a **Linha 1** será executada, em tempo de execução e em função de n .

```
import java.util.Scanner;

public class TarefaT3_03 {
    public static void main(String[] args) {
        Scanner in = new Scanner (System.in);
        int n = in.nextInt();
        System.out.println(Func(n));
        in.close();
    }
    public static int Func(int n) {
        int i = 4;
        int m = 0;
        while (i <= n) {
            m = m + 1;           // Linha 1
            i = i + 2;
        }
        return m;
    }
}
```

6. No algoritmo a seguir, informe a quantidade de vezes que a **Linha 1** será executada, em tempo de execução e em função de n .

```
import java.util.Scanner;

public class TarefaT3_04 {
    public static void main(String[] args) {
        Scanner in = new Scanner (System.in);
        int n = in.nextInt();
        System.out.println(Func(n));
        in.close();
    }
    public static int Func(int n) {
        int i = 1;
        int m = 0;
        while (i <= n) {
            m = m + 1; // Linha 1
            i = i * 2;
        }
        return m;
    }
}
```

7. No algoritmo a seguir, informe a quantidade de vezes que a **Linha 1** será executada, em tempo de execução e em função de n .

```
import java.util.Scanner;

public class TarefaT3_05 {
    public static void main(String[] args) {
        Scanner in = new Scanner (System.in);
        int n = in.nextInt();
        System.out.println(Func(n));
        in.close();
    }
    public static int Func(int n) {
        int m = 0;
        for (int i=1; i <= n; i++)
            for (int j = i; j <= n; j++ ) {
                m = m + 1; // Linha 1
            }
        return m;
    }
}
```

8. Supondo-se que se está comparando implementações de ordenação por inserção e ordenação por intercalação na mesma máquina. Para entradas de tamanho n , a ordenação por inserção é executada $8n^2$ etapas, enquanto a ordenação por intercalação é executada em $64n \ln n$ etapas. Para que valores de n a ordenação por inserção supera a ordenação por intercalação?
9. Qual é o menor valor de n tal que um algoritmo cujo tempo de execução é $100n^2$ funciona mais rápido que um algoritmo cujo tempo de execução é 2^n na mesma máquina?

10. Considere dois algoritmos **A** e **B** com complexidades respectivamente iguais a $128n^2$ e $4n^3$. Qual o maior valor de n , para o qual o algoritmo **B** é mais eficiente que o algoritmo **A**?
11. Considere dois computadores C1 e C2 que executam 10^8 e 10^{10} operações por segundo e dois algoritmos de ordenação **A** e **B** que necessitam $5n^2$ e $40n \log_{10} n$ operações com entrada de tamanho n , respectivamente. Qual o tempo de execução de cada algoritmo em cada um dos computadores C1 e C2 para ordenar 10^8 elementos?
12. Um algoritmo tem complexidade 2^n . Num certo computador, num tempo t , o algoritmo resolve um problema de tamanho **25**. Imagine, agora, que se tenha disponível um computador **100** vezes mais rápido. Qual o tamanho máximo de problema que o mesmo algoritmo resolve no mesmo tempo t no computador mais rápido?