FASE 1 – AULA 3

1. Algoritmos de ordenação

Algoritmo de ordenação em ciência da computação é um algoritmo que coloca os elementos de uma dada sequência (vetor) em uma certa ordem -- em outras palavras, efetua sua ordenação completa ou parcial. As ordens mais usadas são: a numérica e a lexicográfica. Principais algoritmos de ordenação:

- Insertion sort
- Selection sort
- Bubble sort
- Comb sort
- Merge sort
- Heapsort
- Shell sort
- Quick sort

2. Método Bubble Sort

O Bubble Sort, ou ordenação "por bolha", é um algoritmo de ordenação dos mais simples. A ideia é percorrer o vetor diversas vezes, de forma que a cada passagem, o maior ou menor número é posicionado no início do vetor. Essa movimentação lembra a forma como as bolhas em um tanque de água procuram seu próprio nível, e disso vem o nome do algoritmo.

No melhor caso, o algoritmo executa n operações relevantes, onde n representa o número de elementos do vetor. No pior caso, são feitas n^2 operações. A complexidade desse algoritmo é de ordem quadrática – $O(n^2)$, não sendo recomendado para programas que precisem de velocidade e operem com quantidade elevada de dados.

3. Código padrão

```
public static void bubbleSort(int[] chaves) {
        int temp;
        // inicia o Bubble Sort
        for (int i = 0; i < chaves.length; i++) {
            for (int j = 1; j < chaves.length - i; j++) {
                if (chaves[j - 1] > chaves[j]) {
                    //troca os elementos
                    temp = chaves[j - 1];
                    chaves[j - 1] = chaves[j];
                    chaves[j] = temp;
                }
            }
        }
}
```

4. Testes

Faremos repetidos testes com esse código. A ideia é contar quantas vezes a comparação sinalizada em amarelo, será executada em diferentes cenários. Para isso, crie um contador dentro do IF sinalizado e exiba o resultado, anotando-o na tabela abaixo. Lembre-se de preencher o vetor com números aleatórios em uma faixa de valores compatível com a quantidade de elementos do vetor (se a faixa for muito pequena, teremos muitos valores repetidos).

Elementos (n)	**500**	**1.000**	**2.000**	**4.000**	**8.000**	**16.000**	**32.000**	**64.000**
Qtde. de comparações								

5. Atividade para entregar

Você deverá entregar um arquivo .DOC com a tabela comparativa e um gráfico que mostre a curva de crescimento da função que representa esse código.

Ainda, procure na Internet um gráfico resumido que mostre a curva de crescimento de todas as funções assintóticas estudadas (gráfico genérico). Esse gráfico deve estar inserido no arquivo .DOC que você irá entregar.