**Pesquisar sobre complexidade de algoritmo e calcule a complexidade dos algoritmos anteriores**

* **Merge Sort:** Ele é o algoritmo que usa os paradigmas de “dividir e conquistar”. Divide em problema em partes menores, ordena recursivamente e combina os resultados. É um método eficiente de ordenação que utiliza o conceito de divisão e conquista.

**- Complexidade:** A lista é dividida repetidamente e fundida. É o **O(n log n).**

* **Insertion Sort:** Ela é uma técnica simples e eficiente para ordenar elementos em uma lista. Ele funciona comparando cada elemento com os elementos anteriores, e inserindo-o na posição correta na lista ordenada.

**- Complexidade:** A complexidade dele é uma lista ser ocorrida após quando ela está invertida. É o **O(n2).**

* **Bubble Sort:** O bubble sort realiza múltiplas passagem por uma lista. Ele compara itens adjacentes e troca aqueles que estão fora de ordem. Cada passagem pela lista coloca o próximo maior valor na sua posição correta. Em essência, cada item se desloca como uma “bolha” para a posição à qual pertence.

**- Complexidade:** O Bubble Sort faz a comparação entre o elemento da esquerda com o da direita e troca as suas posições caso eles estejam fora de ordem. O algoritmo irá realizar esse processo até que todos os elementos estejam ordenados.

* **Busca Linear:** É um algoritmo que procura um valor específico em uma lista, verificando cada elemento um a um. É um algoritmo simples e intuitivo, mas não é tão eficiente quanto outros algoritmos.

**- Complexidade:** A complexidade se refere em que n é o número de elementos da lista, isso significa que, na pior das hipóteses, o algoritmo pode precisar verificar cada elemento antes de encontrar o alvo (ou determinar que o alvo não está na lista). **O(n)**

* **Busca Binária:** É um algoritmo de busca eficiente utilizado para encontrar um determinado elemento em uma lista ordenada. Ela é baseada no princípio de dividir pela metade, ou seja, a cada iteração, o algoritmo divide a lista em duas partes e verifica em qual metade o elemento procurado está presente. Dessa forma, elimina-se metade da lista a cada iteração, reduzindo drasticamente o tempo de busca em comparação com outros algoritmos de busca.

**- Complexidade:** A complexidade de tempo da busca binária em Python é **O(logn),** onde **n**, é o tamanho da lista. Isso significa que o tempo de execução do algoritmo cresce de forma proporcional ao logaritmo do tamanho da lista.