1. Descreva de forma genérica, ou seja, de forma lógica/matemática, o referido problema da ordenação.

R: O problema de ordenação traz como solução colocar *n* elementos de um conjunto em uma certa ordem. Dependendo da amostra de dados, a ordenação se dá de forma numérica, alfabética, alfanumérica ou qualquer métrica que seja logicamente ou matematicamente comparáveis.

Exemplo de ordenação com elementos numéricos:

[5, 7, 3, 6, 8, 2] -> [2, 3, 5, 6, 7, 8]

Exemplo de ordenação com elementos alfabéticos:

[‘f’, ‘r’, ‘i’, ‘p’, ‘g’] -> [‘f’, ‘g’, ‘i’, ‘p’, ‘r’]

Exemplo de ordenação com elementos alfanuméricos (compara-se com o código da tabela ascii):

[5, 7,‘ i’, 3, ‘f’, ‘r’, ‘p’, ‘g’, 6, 8, 2] -> [2, 3, 5, 6, 7, 8, ‘f’, ‘g’, ‘i’, ‘p’, ‘r’]

1. O referido problema é classificado como fácil ou difícil? Explique.

R: O problema de ordenação é classificado como um problema fácil, pois é possível possível resolver o conjunto com um algoritmo de polinomial (P).

1. Qual o cenário de melhor e também o de pior caso para algoritmo Insertion Sort? Explique cada um deles.

R: O melhor caso para o insertion sort é quando todos os elementos já estão ordenados. Para esse caso, o algoritmo vai precisar passar somente uma vez por todos os elementos do conjunto para verificar se estão ordenados. O(n) onde n é o número de elementos. Já o seu pior caso é um conjunto de elementos já ordenados, porém no sentido contrário. Com isso, caso um conjunto esteja ordenado de forma decrescente e deseja-se ordenar de forma crescente, o algoritmo terá que percorrer todos os elementos do conjunto 2 vezes por interação (uma para selecionar o elemento e outra para descolar o elemento). O(n^2) onde n é o número de elementos.

1. Qual o cenário de melhor e também o de pior caso para algoritmo Merge Sort? Explique cada um deles.

R: O melhor caso seria um conjunto de dados com um único elemento, Pois não seria necessário passar pelo conjunto, deixa a complexidade em O(1). Já para conjuntos com mais de 1 elemento, a complexidade seria O(n log na base 2 n), ou seja, o algoritmo consome o mesmo tempo baseado no número de elementos do conjunto. Isso se dá pois sempre haverá a parte de dividir e de juntar os elementos.

1. Explique as diferenças essenciais entre as duas técnicas de solução de problemas que estudamos denominadas de gulosa e divisão/conquista. Forneça pelo menos um exemplo de cada uma delas para o problema da ordenação.

R: A técnica de dividir e conquistar cuida do problema dividindo-os em problemas menores e resolvendo de forma de forma independente. A parte de dividir lida com a diminuição do problema em um subproblema mínimo. Já a parte de conquistar, liga com da ordenação dos subproblemas. Exemplo de algoritmos: Merge Sort e Quick Sort

Já os algoritmos gulosos tentam solucionar o problema achando a melhor solução no ótimo global. Exemplo de algoritmos: Busca em árvore e algoritmo de Dijkstra