Algoritmos de Ordenação

Questão 1. Implemente em uma linguagem de programação a sua escolha o algoritmo de ordenação QuickSort com o pivô sendo o elemento do meio. Em seguida, trabalhe os itens abaixo:

a. Use uma sequência aleatória de números inteiros entre 1 e 100 como entrada do algoritmo implementado e apresente a sequência ordenada como saída.

b. Compare o tempo de execução do algoritmo quando o mesmo recebe como en- trada sequências ordenadas (de forma crescente e decrescente) e desordenadas. Caso mais números fossem inseridos nas sequências, como o desempenho do algoritmo seria afetado? Usando dados reais, ilustre graficamente suas conclusões.

c. Modifique o algoritmo implementado para que a escolha do pivô seja feita de forma aleatória. Em seguida, refaça os itens (a) e (b) usando o algoritmo modificado. Por fim, faça um estudo comparativo dos dois algoritmos nas situações apresentadas.

Questão 2. Implemente em uma linguagem de programação a sua escolha o algoritmo de ordenação HeapSort. Em seguida, trabalhe os itens abaixo:

a. Use uma sequência aleatória de números inteiros entre 1 e 100 como entrada do algoritmo implementado e apresente a sequência ordenada como saída.

b. Compare o tempo de execução do algoritmo quando o mesmo recebe como en- trada sequências ordenadas (em ordem crescente e decrescente) e desordenadas. Caso mais números fossem inseridos nas sequências, como o desempenho do algoritmo seria afetado? Usando dados reais, ilustre graficamente suas conclusões.

c. Faça um estudo comparativo do algoritmo HeapSort implementado com os algorit- mos QuickSort, usados na Questão 1, nas situações apresentadas.

Questão 3. Implemente em uma linguagem de programação a sua escolha os algoritmos de ordenação linear CountingSort e RadixSort. Em seguida, trabalhe os itens abaixo:

a. Analise o comportamento do tempo de execução do algoritmo CountingSort quando o número de elementos da sequência de entrada é aumentado gradativamente. Durante a análise, comente a relação que existe entre o tamanho da sequência de entrada e o elemento de maior valor nessa sequência. Usando dados reais, ilustre graficamente suas conclusões.

b. Faça um estudo comparativo do tempo de execução do algoritmo CountingSort com as implementações do QuickSort, feitas na Questão 1, em situações reais (práticas) de uso. Explique qual algoritmo é mais eficiente e por que.

c. Usando o algoritmo RadixSort, ordene os CEPS do estado do Amapá. Especifique o algoritmo de ordenação estável utilizado na implementação. Durante a tarefa, analise o comportamento do tempo de execução a medida que a quantidade de CEPS é aumentada nasequência de entrada. Usando dados reais, ilustre graficamente suas conclusões.