AED II - 2023 - ATIVIDADE DE PROGRAMAÇÃO 04 - QUICKSORT

Instruções:

1. E/S: tanto a entrada quanto a saída de dados devem ser "secas", ou seja, não devem apresentar frases explicativas. Siga o modelo fornecido e apenas complete as partes informadas (veja o exemplo abaixo).
2. Identificadores de variáveis: escolha nomes apropriados
3. Documentação: inclua cabeçalho, comentários e indentação no programa.
4. Submeta o programa no sistema judge: `http://judge.unifesp.br/aed2S01A2022/`

Descrição:

A eficiência do algoritmo *QuickSort* está ligada diretamente à escolha do elemento que servirá como pivô. Quando o último elemento do vetor é escolhido, vetores previamente ordenados (em ordem crescente ou decrescente) levam a uma complexidade quadrática ($O(n^2)$), aumentando em muito o tempo de execução do código.

Uma forma de diminuir a probabilidade de pior caso é escolher o pivô pelo método da mediana de três. Nesse método, são separados três elementos do vetor - por exemplo, o primeiro, o último, e o do meio - e o pivô será o elemento com valor intermediário entre os três. Por exemplo, se os elementos separados são a, b, e c tais que:

$$a < b < c$$

então o pivô deverá ser o elemento b.

Na atividade desta semana, você deverá implementar a versão recursiva do *QuickSort* de forma que o pivô possa ser tanto o último elemento do vetor quanto um elemento escolhido pelo método descrito acima. Seu objetivo é comparar a altura máxima e mínima entre os nós folhas. Lembre-se que o algoritmo consiste em separar o vetor em pedaços menores. Considere a menor altura o lado que terminar primeiro a ordenação, e a maior altura aquele que terminar por último.

Seu programa deverá retornar a diferença entre a altura máxima e a mínima nos dois casos (último elemento do vetor, e usando mediana de três), ou seja, o vetor de entrada deverá ser ordenado duas vezes.

Considere as seguintes condições:

1. Sua solução deve implementar a versão **recursiva** do *QuickSort*;
2. O vetor de entrada deverá ser ordenado **duas vezes**: na primeira o pivô inicial deve ser igual ao último elemento, enquanto na segunda o pivô deverá ser determinado pela mediana de três;
3. O código-fonte **deve** ser escrito em C/C++ ou Java;

4. **Toda** memória alocada dinamicamente (C/C++) deve ser desalocada;

5. **Nenhuma** variável global deve ser utilizada;

Entrada:

A primeira linha contém o valor de um inteiro n que representa o tamanho do vetor a ser ordenado.

A segunda linha contém os elementos do vetor de entrada, separados por espaço.

Saída:

A primeira linha contém um único número inteiro, que representa a diferença entre a altura máxima e mínima dos nós folhas do QuickSort. Neste caso, o pivô é o último elemento do vetor.

A segunda linha também contém a diferença entre as alturas máxima e mínima dos nós folha, mas para o caso em que o pivô é escolhido pela mediana de três.

Exemplos de entrada e saída:

Note que a profundidade inicial é 0, e cada chamada recursiva aumenta a profundidade em 1 nos dois lados em que ainda não se encontrou um nó folha.

Exemplos de entrada	**Exemplos de saída**
5 19 42 20 57 53	1 0
10 10 9 8 7 6 5 4 3 2 1	8 1

Tabela 1: Exemplos de entrada e saída