1 Quicksort

Em aula vimos o algoritmo de Quicksort. Nesta tarefa é solicitada a implementação de quatro versões deste algoritmo, usando duas estratégias para escolha de particionador, e duas estratégias de particionamento.

As estratégias para escolha do particionador que devem ser implementadas são:

1. Escolha 1 (Mediana de 3): Particionador deve ser a mediana entre o primeiro, último e elemento na posição média de cada sub-vetor (média entre o índice do primeiro e último elemento, arrendodada para baixo);
2. Escolha 2 (Aleatório): Particionador será um elemento aleatório do sub-vetor.

A estratégia de escolha deve primeiro definir qual será o particionador, e após feita esta escolha, colocar o particionador na primeira posição do sub-vetor, trocando com o elemento atualmente na primeira posição. Após esta troca, implemente o algoritmo de Quicksort como visto em aula usando o primeiro elemento como particionador.

Duas estratégias de particionamento devem ser implementadas, conforme vistas em aula:

1. Particionamento 1 (Lomuto)
2. Particionamento 2 (Hoare)

Para todas as quatro combinações (escolha x particionamento), teste cada uma das versões do algoritmo com o arquivo *entrada-quicksort.txt*, contendo vetores aleatórios de tamanho 100, 1000, 10000, 100000 e 1000000 em anexo. O programa deve ler os vetores de um arquivo da entrada padrão, e gravar resultados na saída padrão[1]. A entrada padrão deve ser redirecionada para o arquivo *entrada-quicksort.txt*, contendo vários vetores a serem ordenados. Cada vetor é descrito em 1 linha. O primeiro número da linha descreve o tamanho n do vetor, seguido por n números do vetor. Um exemplo que ilustra o formato do arquivo de entrada segue:

```
16 16 14 12 1 8 4 9 6 15 13 11 2 7 3 10 5
16 3 10 5 16 14 12 1 8 4 9 6 15 13 11 2 7
```

Para cada uma das 4 combinações de escolha e particionamento, execute o código com o arquivo de entrada, e imprima as seguintes estatísticas no exato formato listado abaixo:

```
TAMANHO ENTRADA 100
SWAPS #SWAPS
RECURSOES #RECURSOES
TEMPO #TEMPO EM SEGUNDOS
TAMANHO ENTRADA 1000
SWAPS #SWAPS
RECURSOES #RECURSOES
TEMPO #TEMPO EM SEGUNDOS
TAMANHO ENTRADA 10000
SWAPS #SWAPS
RECURSOES #RECURSOES
TEMPO #TEMPO EM SEGUNDOS
TAMANHO ENTRADA 100000
SWAPS #SWAPS
```

[1] https://www.ime.usp.br/~pf/algoritmos/aulas/io.html

```
RECURSOES #RECURSOES
TEMPO #TEMPO EM SEGUNDOS
TAMANHO ENTRADA 1000000
SWAPS #SWAPS
RECURSOES #RECURSOES
TEMPO #TEMPO EM SEGUNDOS
```

Para cada execução, salve essas estatísticas em arquivos com os seguintes nomes:

- *stats-mediana-hoare.txt*
- *stats-mediana-lomuto.txt*,
- *stats-aleatorio-hoare.txt*
- *stats-aleatorio-lomuto.txt*

2 Entrega

A solução deve ser enviada pelo Moodle dentro de um arquivo .zip, contendo os seguintes arquivos:

- **integrantes.txt**: coloque o nome dos integrantes do grupo (até 2 pessoas) , com um nome por linha
- **stats-mediana-hoare.txt**: no exato formato listado acima
- **stats-mediana-lomuto.txt**: no exato formato listado acima
- **stats-aleatorio-hoare.txt**: no exato formato listado acima
- **stats-aleatorio-lomuto.txt**: no exato formato listado acima
- código fonte correspondente a solução

3 Desafio Bonus - 25% extra

Resolva o problema "10810 - Ultra-QuickSort"
`https://uva.onlinejudge.org/index.php?option=com_onlinejudge&Itemid=8&category=20&page=show_problem&problem=1751`
Este é um problema de maratona de programação. O problema será considerado completo se for aceito para no site da https://uva.onlinejudge.org

4 Desafio Bonus - 25% extra

Este problema eu pessoalmente acho muito interessante, embora sua solução esteja mais relacionada com a teoria de combinatória. Portanto, não espero que seja resolvido, mas para quem aprecia desafios e aprender algo mais avançado pode considerá-lo desafiador. Duas dicas. Primeiro, este problema requer achar uma fórmula fechada para contar o número de heaps diferentes em árvores multi-árias. Segundo, o resultado excede o que se pode representar como um inteiro, e portanto requer uma implementação adicional para suportar números grandes.

O problema chama-se "10247 - Complete Tree Labeling"
`https://uva.onlinejudge.org/index.php?option=com_onlinejudge&Itemid=8&category=14&page=show_problem&problem=1188`
Este problema será considerado completo se for aceito para no site da https://uva.onlinejudge.org. Além disso, eu quero um texto explicando brevemente como o problema foi resolvido.