Exercícios - métodos de ordenação eficientes

1) Qual será a complexidade do quick sort se os itens do vetor forem todos iguais? Ex.: [10, 10, 10, 10, … , 10].

2) No quick sort, é possível trocar a comparação "p < r" por "p != r"?

3) Reescrever o código de separação do quick sort para que o primeiro item do vetor seja usado como pivô. Isso irá mudar a complexidade do algoritmo?

4) Adaptar a função separa() para que ela separe um vetor contendo apenas números positivos e negativos em dois vetores: um contendo apenas números positivos e um contendo apenas números negativos.

5) Existe algum caso onde o desempenho da ordenação por inserção é melhor do que o desempenho do quicksort?

6) Na aula passada estudamos o quick sort usando o método de particionamento proposto por Lomuto, que ganhou popularidade em vários livros de algoritmos. Entretanto, originalmente, o quick sort foi implementado usando o método de particionamento de Hoare. Pesquise sobre este método.

7) Vimos que uma prática comum no quick sort é selecionar o primeiro ou último item do vetor como pivô. Pesquise outras formas de selecionar um pivô para o quick sort e qual seu impacto no desempenho do algoritmo.

8) Existe algum caso onde o desempenho da ordenação por inserção é melhor que o desempenho do merge sort?

9) Você possui um vetor de livros contendo 10.000 livros. Cada livro contém várias informações, entre elas está o preço. Você quer montar um vetor com os 100 livros mais baratos. Você pensa em duas formas de selecionar os 100 livros mais baratos:
1 - Você repete 100 vezes uma busca onde percorre todo o vetor procurando o livro de menor preço.
2 - Você ordena o vetor usando o método merge sort e depois seleciona somente os 100 primeiros livros do vetor.
Qual método você escolheria? Por quê?