Algoritmos e Estruturas de Dados II

1. Observe o vetor abaixo.

$$\begin{pmatrix} 2 & 7 & 8 & 5 & 3 \end{pmatrix}$$

   (a) Para o *bubbleSort*, ele é pior, médio ou melhor caso? E para o *insertionSort*?

   (b) Simule a execução do *bubbleSort* sobre este vetor contando comparações e trocas.

   (c) Repita o item anterior para o *insertSort*.

   (d) Os números nos itens b e c são diferentes dos esperados para os casos que você listou no item a? Se sim, explique a diferença.

   (e) Simule a execução do *selectSort* sobre o vetor, contando as comparações e trocas. Esses números correspondem ao desempenho esperado do algoritmo?

   (f) Simule a execução do *mergeSort* sobre o vetor, contando as comparações. Esses números correspondem ao desempenho esperado do algoritmo?

2. Para cada vetor abaixo, categorize-o em caso (pior, médio ou melhor) de execução do *quickSort* e escreva o número de comparações esperado.

   - 1 2 3 4 5 7 8
   - 4 1 2 3 5 7 8
   - 8 7 5 4 3 2 1
   - 4 2 7 1 3 5 8
   - 7 1 3 2 4 5 8
   - 4 2 1 3 7 5 8
   - 2 7 8 5 3 4 1

3. Para os vetores do exercício anterior, simule a execução do *quickSort*

   (a) sobre o vetor mais custoso

   (b) sobre o vetor menos custoso

   (c) sobre algum vetor de custo não extremo

4. Para um vetor de N elementos, o algoritmo denominado *bucketSort* opera da seguinte maneira:

   - Cria N baldes (vetores)
   - Para cada elemento X no vetor a ser ordenado, insere X no balde de número $\frac{NX}{max(vetor)}$
   - Ordena os baldes com algum outro algoritmo de ordenação
   - Insere os baldes no vetor, em ordem (todos do primeiro balde, todos do segundo, e assim por diante)

   Implemente o *bucketSort* em pseudocódigo e, depois, em C.

5. É possível melhorar a execução do *quickSort* aproveitando da velocidade do *insertSort* quando este é aplicado sobre um vetor *quase* ordenado.

   (a) Implemente um *quickSort* que pare as chamadas recursivas quando os subvetores forem de tamanho menor que uma constante k e que chame o *insertSort* no vetor quase ordenado (depois das chamadas recursivas).

   (b) Sugira um método para escolha do valor de k.

6. Organize um tipo abstrato de dados *crossfiteiro* que contemple a vida dos praticantes de *crossfit* sob as seguintes especificações (constantes e proporções não são definidas - faça como preferir):

   Atributos

   - Score de `fitness`, que é inicializado baixo
   - Score de `motivação`, que é inicializado alto
   - Vetor de amigos, que são outros *crossfiteiro*s
   - Vetor de fotos (com o número de "curtidas" representando cada foto, onde -1 indica que não há foto no índice)
   - `curtidas`, que é a soma do vetor de fotos com exceção de valores iguais a -1

   Funções

   - `adicionaAmigo`: dados dois *crossfiteiro*s, insere um na lista de amigos do outro
   - `removeAmigo`

- `postaFoto`: dado um *crossfiteiro*, adiciona uma foto ao seu vetor de fotos. As curtidas na foto devem ser uma função de `fitness`, do número de amigos e de `motivação`, sendo que a última só deve impactar no valor de curtidas se for suficientemente alta (a ideia é que a *vibe* de um *crossfiteiro* chegue ao ponto de ser cativante para seus amigos, também *crossfiteiro*s)
- `removeFoto`
- `treina`: dado um *crossfiteiro* e um tempo, atualiza `fitness` em função do tempo e de `motivação`. Se `motivação` for suficientemente baixa, `fitness` deve ser decrementada e a função `postaFoto` deve ser chamada (nosso *crossfiteiro* foi ao treino com preguiça e ficou só usando o celular)
- `atualiza`: dado um *crossfiteiro*, realiza operações de manutenção:
  - Se o vetor de fotos lotou, o *crossfiteiro* está enchendo a paciência das pessoas nas redes sociais e deve perder um quarto dos amigos;
  - A motivação deve ser decrementada de um valor fixo;
  - A variável `curtidas` deve ser atualizada (iterando sobre o vetor de fotos).

  Observação: você pode executar a simulação disso num *loop* de dias com algumas ações (treino, postagem de foto, etc) que aconteçam dependendo do valor de um inteiro aleatório, por exemplo.

7. Imagine que todos os *crossfiteiros* do exercício anterior tenham entrado em uma distopia clichê e agora o valor de suas amizades é medido pela influência que seus amigos têm nas redes sociais. Adapte seus programas dos exercícios 5 e 6 para que a função `atualiza` ordene o vetor de amigos de cada *crossfiteiro* segundo o número de `curtidas`, onde o amigo mais importante fica na posição 1, o segundo na posição 2, etc.