P2 — Estruturas de Dados — Prof. Vinícius Gusmão — 2021/2

1) Indique V(erdadeiro) ou F(also), sem justificar. (0,75 pontos por item)

a) Uma árvore binária de busca balanceada com n chaves tem complexidade de pior caso O(log n) para buscar um elemento.
b) Uma heap binária de mínimo retorna o menor elemento de um conjunto de n elementos em em tempo O(1), e o maior elemento do conjunto em tempo O(log n).
c) O tempo de acesso a uma chave armazenada em uma tabela hash com n chaves é, no pior caso, O(*alfa*), onde *alfa* é o fator de carga da tabela.
d) Para consultar se um conjunto de strings possui elementos que comecem com certo prefixo, é em geral mais vantajoso ter esse conjunto armazenado numa trie do que num hash set.
e) O QuickSort tem esse nome porque, na prática, sua performance é muito boa, embora sua complexidade de tempo de pior caso seja O(n^2).
f) Para conseguir tempo médio O(1) nas operações de busca e inserção em uma tabela hash com encadeamento externo, é indispensável que o array subjacente seja dimensionado com tamanho sempre maior ou igual ao número de chaves que serão inseridas.
g) A sequência 5, 6, 11, 7, 9, 15, 20, 8, 22, 11, 30, 45, 60, 19, 800 corresponde a uma heap binária de mínimo, onde o elemento de índice j na sequência (cujo primeiro elemento tem índice 1) é o pai, na heap, dos elementos de índice $2j$ e $2j + 1$, para todo j de 1 a 7.
h) Sob certas condições, é possível ordenar uma lista de n elementos em tempo O(n).

2) Suponha que as pontuações dos jogadores num jogo multiplayer online sejam armazenadas em estruturas que guardem dois campos por jogador — o identificador (id) do jogador (um inteiro) e a pontuação daquele jogador (outro inteiro). Lembrando que uma AVL tree é uma árvore binária de busca que se mantém sempre balanceada, que estrutura de dados para armazenamento das contas em memória, *dentre as opções dadas*, apresentaria o melhor desempenho médio *para cada uma das operações abaixo, isoladamente*? (1,0 ponto por item)

a) alteração da pontuação, dado o id do jogador
( ) heap de mínimo ( ) array ordenado ( ) hash map ( ) trie

b) consulta do jogador com a k-ésima maior pontuação
( ) heap de máximo ( ) array ordenado ( ) hash table ( ) AVL tree

c) consulta do jogador com a menor pontuação
( ) heap de máximo ( ) heap de mínimo ( ) hash table ( ) AVL tree

d) listagem de todos os jogadores com pontuação entre *min* e *max* dados
( ) heap de mínimo ( ) lista encadeada ( ) hash table ( ) AVL tree