Gabarito

01 –
a) O algoritmo TESTE verifica e corrige as prioridades dos nós internos da árvore em relação a seus descendentes. Como os nós i > n/2 de um heap são nós-folha, o algoritmo TESTE basicamente não realizará nenhuma correção.

b) O primeiro passo é transformar o vetor A = [10 13 12 17] em um heap máximo. Executando o algoritmo MAX-HEAP, apresentado em sala, temos que o vetor A resultante será [17 13 12 10]. Em seguida, a 1.ª posição do vetor é trocada com a última, e o procedimento TESTE é executado na 1.ª posição considerando o vetor (virtualmente) com tamanho n – 1. Esse processo é repetido até que o heap tenha tamanho igual a 2.

02 –
a) A complexidade dessa execução é Theta(n log n) — caso médio do algoritmo.
S O R T
S O R
O R S
O
S

b) Não é possível melhorar, pois, o melhor caso é assintoticamente equivalente ao caso médio.

03 – B

04 –
a) A – Mergesort
B – Heapsort
C – Quicksort

b) O pivô encontra-se em uma das extremidades do vetor.

c) Sim, atentando para o fato que se o vetor tiver todos os elementos iguais, então sua complexidade no tempo é linear em n.

05 –
a) A = [4 0 2 0 1]

C = [2 1 1 0 1] C[0..m], onde m = 4
C = [2 3 4 4 5]

B = [_ _ 1 _ _] C = [2 2 4 4 5]
B = [_ 0 1 _ _] C = [1 2 4 4 5]
B = [_ 0 1 2 _] C = [1 2 3 4 5]
B = [0 0 1 2 _] C = [0 2 3 4 5]
B = [0 0 1 2 4] C = [0 2 3 4 4]

b) O tempo de execução esperado é Theta(n), já que m é O(n).

c) Sim, é estável. Elementos iguais ocorrem no vetor ordenado na mesma ordem em que aparecem na entrada.

06 – B
07 – D
08 – A
09 – C
10 – D
11 – D
12 – D
13 – D
14 – A
15 – E
16 – E
17 – B
18 – D
19 – D
20 – C
21 – C