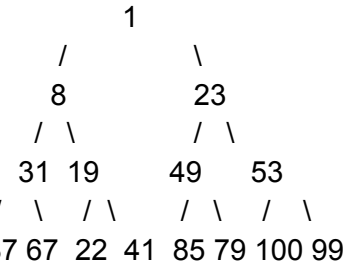
Questão 1.

Custo de construção: O(n)

Vetor resultante:
1, 8, 23, 31, 19, 49, 53, 87, 67, 22, 41, 85, 79, 100, 99

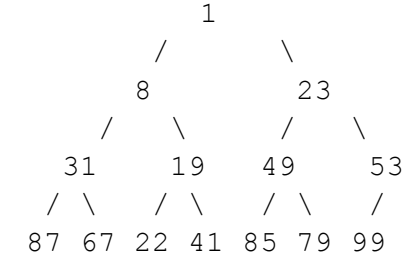
Representação em árvore:



Questão 2.

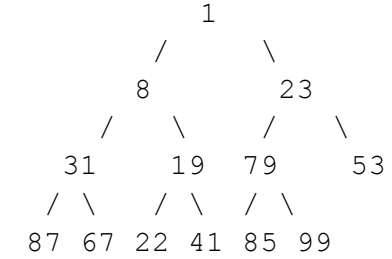
Remover 100:
Vetor resultante: 1, 8, 23, 31, 19, 49, 53, 87, 67, 22, 41, 85, 79, 99

Representação em árvore:



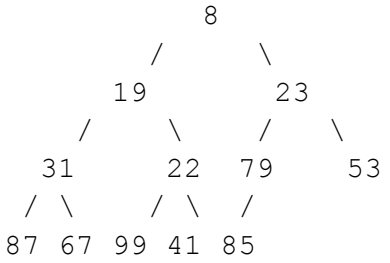
Remover 49:
Vetor resultante: 1, 8, 23, 31, 19, 79, 53, 87, 67, 22, 41, 85, 99

Representação em árvore:



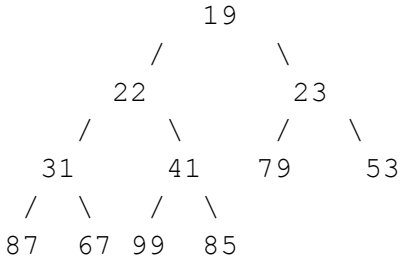
Remover 1:
Vetor resultante: 8, 19, 23, 31, 22, 79, 53, 87, 67, 99, 41, 85

Representação em árvore:



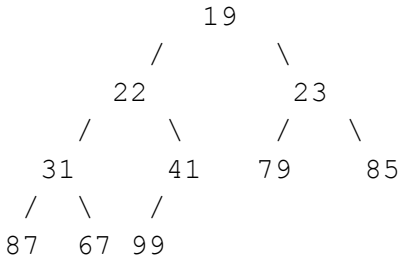
Remover 8:
Vetor resultante: 19, 22, 23, 31, 41, 79, 53, 87, 67, 99, 85

Representação em árvore:



Remover 53:
Vetor resultante: 19, 22, 23, 31, 41, 79, 85, 87, 67, 99

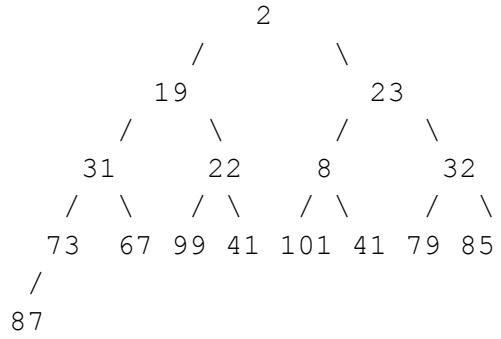
Representação em árvore:



Questão 3.

Inserção dos valores: 2, 101, 41, 8, 32 e 73.
Vetor Resultante: 2, 19, 23, 31, 22, 8, 32, 73, 67, 99, 41, 101, 41, 79, 85, 87

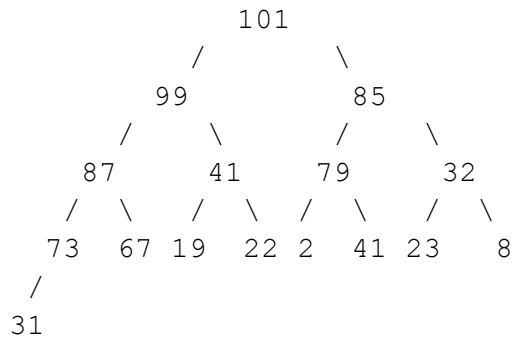
Representação em árvore:



Questão 4.

Vetor Resultante: 101, 99, 85, 87, 41, 79, 32, 73, 67, 19, 22, 2, 41, 23, 8, 31

Representação em árvore:



Questão 5.

A utilização do Heap Sort é recomendada principalmente em sistemas que exigem um desempenho de pior caso garantido em O(n log n), proporcionando uma previsibilidade que algoritmos como o Quick Sort não oferecem, e em cenários com extrema restrição de memória, visto que ele opera com complexidade de espaço O(1).