QUESTÕES DE ARVORE E GRAFOS

1. RESPOSTA

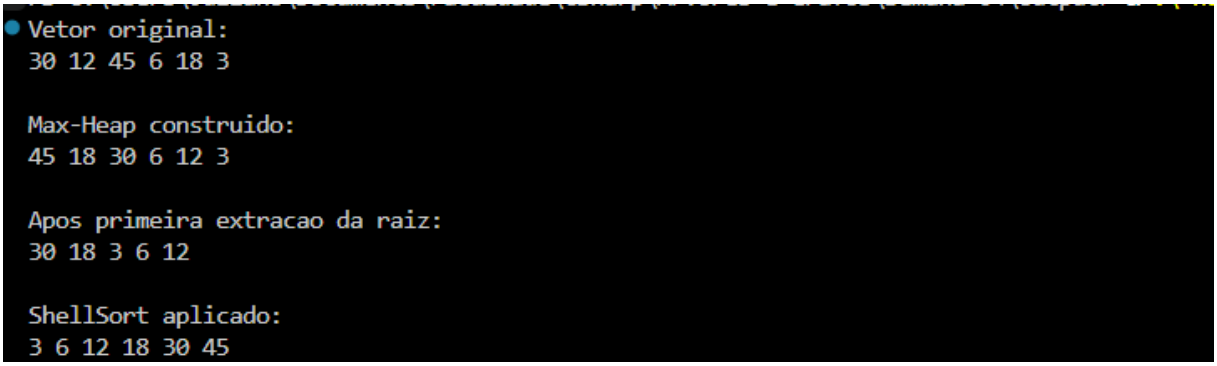
O HeapSort possui complexidade O(n log n) garantida em todos os casos, pois devido a divisão heap é O(n) e cada extração n da raiz é O(log n), então quando se junta, o seu custo fica O(n log n)

Essa estrutura é baseada em uma árvore binária completa, o que garante altura sempre proporcional a log n, independentemente da ordem inicial dos dados.

Já o ShellSort não possui complexidade previsível devido ao seu desempenho depender da sequência de gaps utilizada, além que as diferentes sequências produzem diferentes complexidades.

Portanto, o HeapSort tem desempenho garantido, enquanto o ShellSort é dependente da estratégia adotada.

2. RESPOSTA



3. RESPOSTA

A) Apenas II e IV

I – ShellSort é estável.

Como explicado na questão 01, o ShellSort não é estável, devido aos gaps e sua sequencia

II – HeapSort utiliza estrutura baseada em árvore binária completa.

Sim, o Heap Sort aplica de um sistema de divisão que divide o vetor em uma árvore binária para sua organização (as “folhas” e o “tronco”).

III – HeapSort depende da sequência de gaps.

HeapSort, não se utiliza de gaps (saltos) para sua ordenação e sim galhos.

IV – ShellSort é uma melhoria do Insertion Sort.

Correto, o shellsort é uma melhoria do insertion sort, pois aplica inserções de acordo com os gaps.

4. RESPOSTA

Em um sistema que processa milhões de registros diariamente, o algoritmo mais indicado é o HeapSort, uma vez que possui complexidade garantida O(n log n). Portanto, não depende da distribuição inicial dos dados e tem comportamento previsível.

Em sistemas grandes, previsibilidade e estabilidade de desempenho são fundamentais.

5. RESPOSTA

A estrutura Heap é utilizada como fila de prioridade em algoritmos de grafos, pois permite inserir elementos com prioridade, remover rapidamente o elemento de menor ou maior valor e atualizar prioridades de maneira eficiente. Essa característica é fundamental em algoritmos que precisam selecionar repetidamente o próximo elemento mais relevante. Um exemplo clássico é o algoritmo de Dijkstra's algorithm, que utiliza um Min-Heap para selecionar sempre o vértice com a menor distância conhecida até o momento. Sem o uso de heap, a complexidade do algoritmo é $O(V^2)$. Já com a utilização de heap, a complexidade passa a ser $O((V + E) \log V)$, tornando o algoritmo significativamente mais eficiente em grafos grandes.