

Tutoria 08

QUESTÃO 1.

Vetor original: [20, 48, 27, 43, 63, 9, 92, 31]

MergeSort:

Divisão Inicial:

Esquerda: [20, 48, 27, 43]

Direita: [63, 9, 92, 31]

Lado Esquerdo:

Dividir [20, 48, 27, 43] -> [20, 48] e [27, 43]

Ordenar e Mesclar [20, 48] -> [20, 48]

Ordenar e Mesclar [27, 43] -> [27, 43]

Merge da Esquerda: [20, 48] + [27, 43] = [20, 27, 43, 48]

Lado Direito:

Dividir [63, 9, 92, 31] -> [63, 9] e [92, 31]

Ordenar e Mesclar [63, 9] -> [9, 63]

Ordenar e Mesclar [92, 31] -> [31, 92]

Merge da Direita: [9, 63] + [31, 92] = [9, 31, 63, 92]

Merge Final:

[20, 27, 43, 48] e [9, 31, 63, 92] -> [9, 20, 27, 31, 43, 48, 63, 92]

QuickSort:

Primeira Partição:

Pivô: 31

Esquerda: [20, 27, 9]

Direita: [43, 63, 48, 92]

Estado do Vetor: [20, 27, 9] 31 [43, 63, 48, 92]

Partição da Esquerda ([20, 27, 9]):

Pivô: 9

Esquerda: []

Direita: [20, 27]

Estado do Subvetor: 9 [20, 27]

Partição da Direita ([20, 27]):

Pivô: 27

Esquerda: [20]

Direita: []

Estado do Subvetor: 20 27 (Fim da recursão à esquerda)

Partição da Direita ([43, 63, 48, 92]):

Pivô: 92

Esquerda: [43, 63, 48]

Direita: []

Estado do Subvetor: [43, 63, 48] 92

Partição da Esquerda ([43, 63, 48]):

Pivô: 48

Esquerda: [43]

Direita: [63]

Estado do Subvetor: 43 48 63

Vetor Final Ordenado: [9, 20, 27, 31, 43, 48, 63, 92]

ShellSort:

Gaps utilizados: 7, 3 e 1.

Passo 1: $h = 7$

Compara $V[0]$ e $V[7]$ (20 e 31): Não troca.

Vetor após $h=7$: [20, 48, 27, 43, 63, 9, 92, 31]

Passo 2: $h = 3$

Subgrupo 1 ($i = 0, 3, 6$): [20, 43, 92] -> [20, 43, 92] (Sem trocas)

Subgrupo 2 ($i = 1, 4, 7$): [48, 63, 31] -> [31, 48, 63] (Trocas: 31 move para o início)

Subgrupo 3 ($i = 2, 5$): [27, 9] -> [9, 27] (Troca: 9 e 27)

Vetor após $h=3$: [20, 31, 9, 43, 48, 27, 92, 63]

Passo 3: $h = 1$

[20, 31, 9, 43, 48, 27, 92, 63]

[20, 9, 31, 43, 48, 27, 92, 63]

[9, 20, 31, 43, 48, 27, 92, 63]

[9, 20, 31, 43, 27, 48, 92, 63]

[9, 20, 31, 27, 43, 48, 92, 63]

[9, 20, 27, 31, 43, 48, 92, 63]

[9, 20, 27, 31, 43, 48, 63, 92]

Vetor Final: [9, 20, 27, 31, 43, 48, 63, 92]

QUESTÃO 2.

O MergeSort é um algoritmo estável que garante desempenho $O(n \log n)$ em qualquer situação, sendo ideal para grandes volumes de dados, embora exija memória extra para a intercalação.

O QuickSort, apesar de instável e possuir um pior caso de $O(n^2)$, é geralmente o mais rápido na prática por ser in-place e possuir constantes menores em sua execução.

Já o ShellSort é uma melhoria do Insertion Sort que reduz o custo de movimentação de elementos distantes usando saltos, sendo eficiente para vetores de tamanho moderado sem a necessidade de recursividade.

QUESTÃO 3.

Um Heap é uma árvore binária quase completa que mantém uma ordem específica entre pais e filhos, sendo a base para filas de prioridade e o algoritmo HeapSort. No Heap Máximo, o pai é sempre maior ou igual aos filhos, mantendo o maior valor na raiz para acesso imediato. No Heap Mínimo, o pai é sempre menor ou igual aos filhos, garantindo o menor valor no topo. Sua principal vantagem é permitir a inserção e remoção do elemento prioritário em tempo logarítmico.

QUESTÃO 4.

Para o conjunto [14, 12, 56, 77, 465, 1, 8, 96], a construção do Heap Máximo resulta na organização [465, 96, 56, 77, 12, 1, 8, 14], enquanto o Heap Mínimo organiza-se como [1, 12, 8, 77, 465, 56, 14, 96]. O custo para construir essas estruturas a partir de um vetor desordenado é de $O(n)$, pois o algoritmo de construção otimizado (heapify) processa a árvore de baixo para cima, minimizando o número de trocas necessárias.