Heap Sort: Uma Análise Detalhada

Esta apresentação explora o algoritmo Heap Sort. Vamos detalhar seu funcionamento, estrutura e aplicações. Incluiremos um exemplo prático em Python. Prepare-se para uma imersão completa!

👻 Feito por Luciano dos Santos Nascimento e Wesley Silva Araújo 👻

O Que é Heap Sort e Suas Vantagens?

Definição

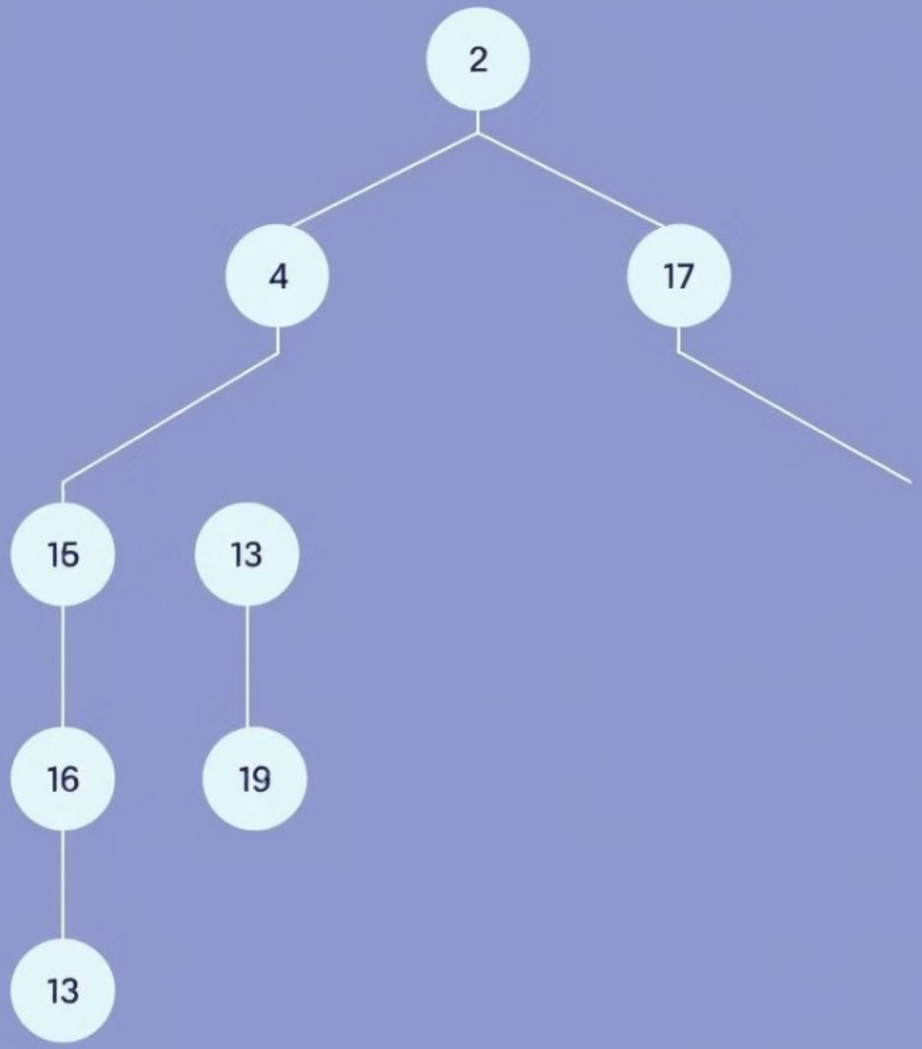
Heap Sort é um algoritmo de ordenação eficiente. Ele utiliza a estrutura de dados Heap.

Vantagens

Possui bom desempenho em diversos cenários. É uma boa escolha para grandes volumes de dados.

Como Funciona?

Primeiro, constrói um Heap a partir dos dados. Depois, extrai os elementos na ordem correta.



Entendendo a Estrutura de Dados Heap

1 O Que é Heap?

É uma árvore binária completa. Satisfaz a propriedade Heap (max-heap ou min-heap).

2 Max-Heap

O valor de cada nó é maior ou igual ao de seus filhos.

3 Min-Heap

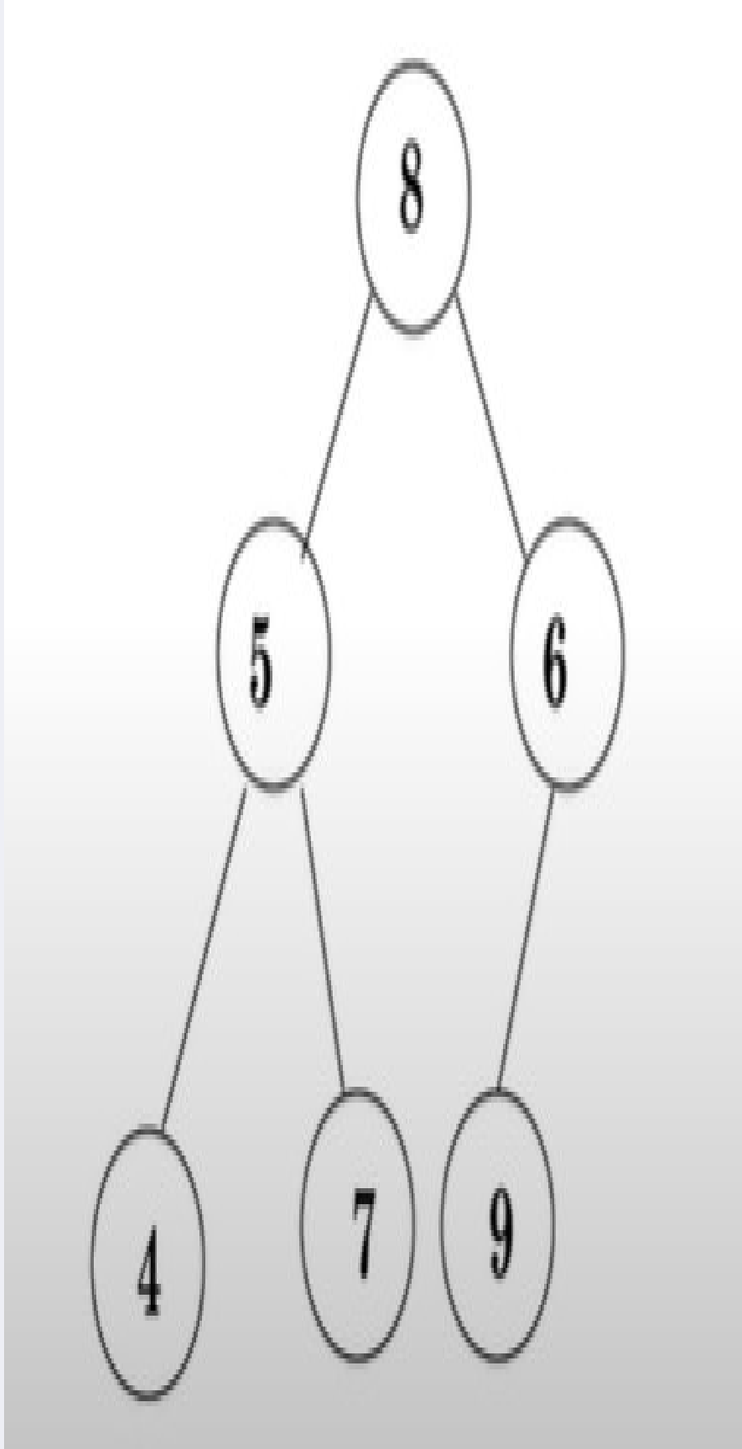
O valor de cada nó é menor ou igual ao de seus filhos.

Como o Algoritmo Heap Sort Funciona: Passo a Passo

1

Construir o Heap

Transformar o array ([8, 5, 6, 4, 7, 9]) de entrada em um Heap.

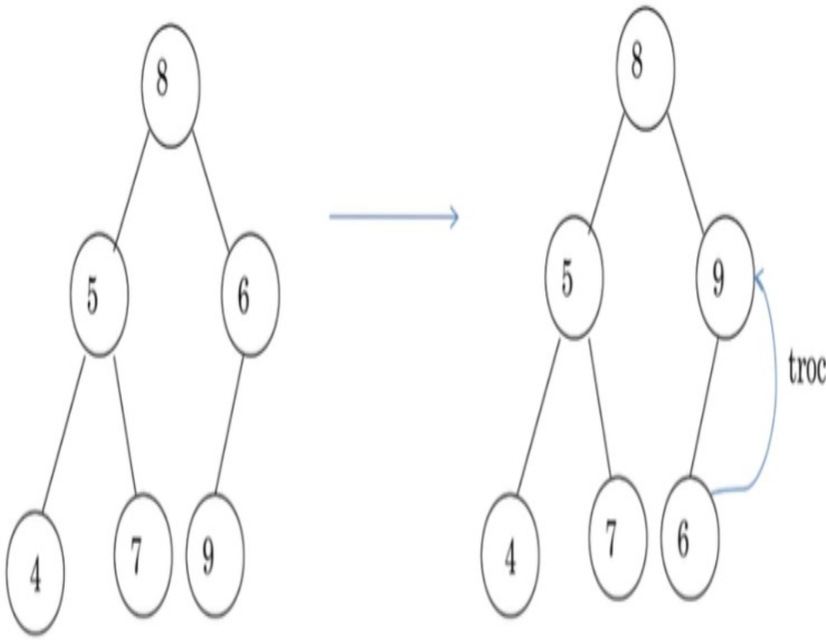


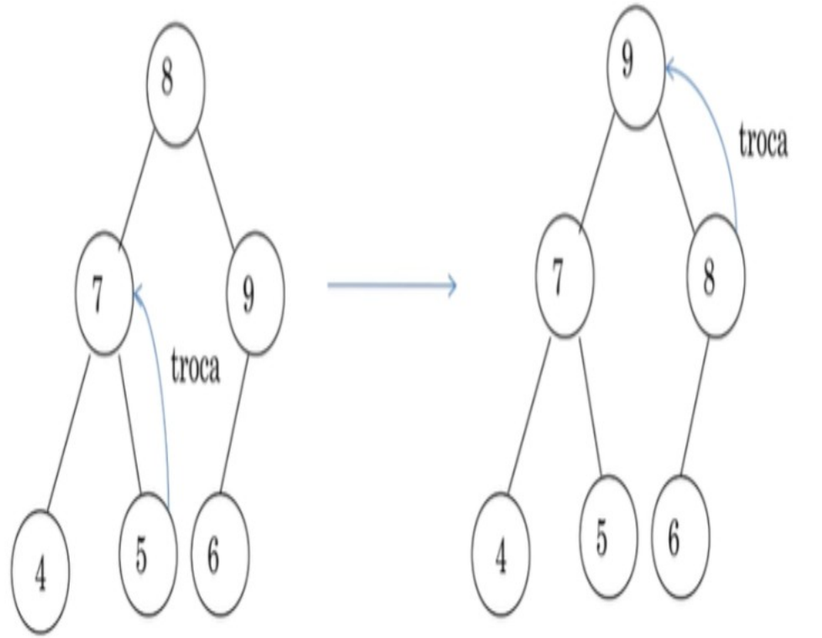
Como o Algoritmo Heap Sort Funciona: Passo a Passo

2

Reorganizar e extrair o maior valor

Remover o elemento raiz (máximo ou mínimo).



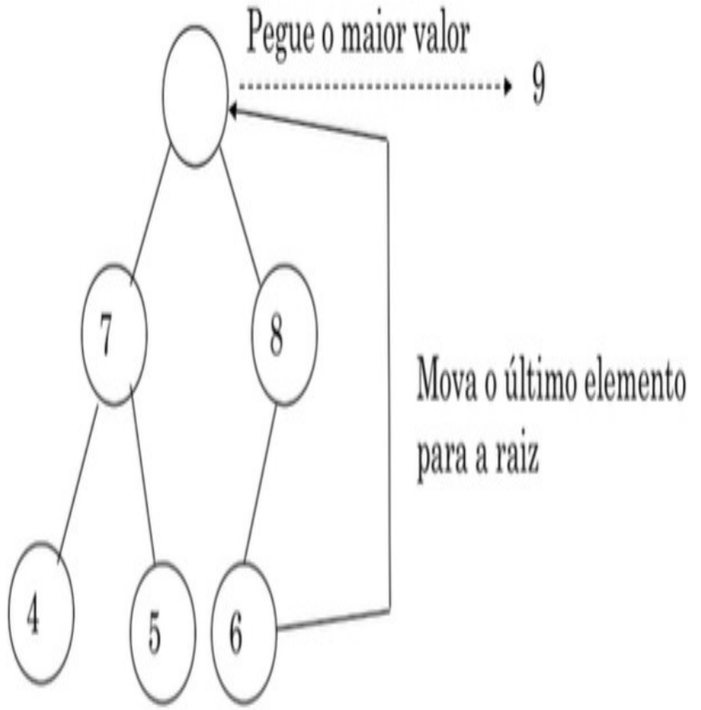


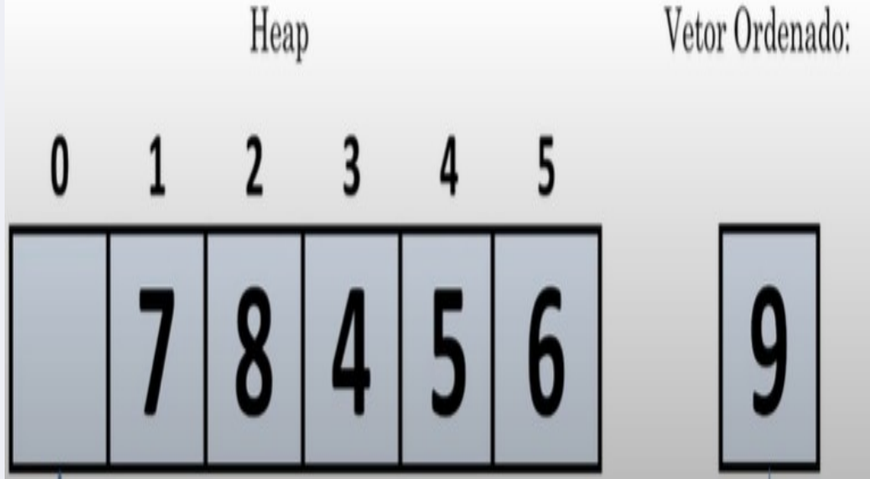
Como o Algoritmo Heap Sort Funciona: Passo a Passo

3

Heapify

Reorganizar o Heap após a remoção.



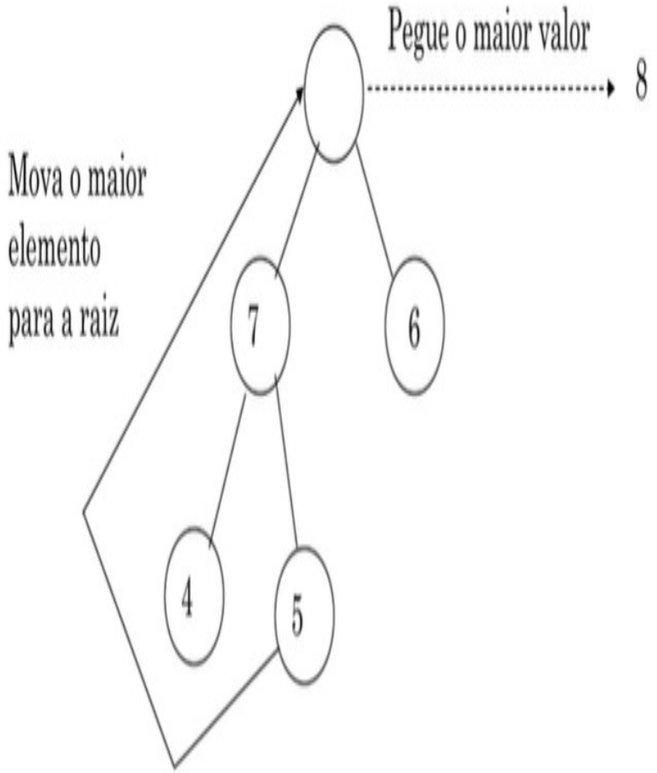


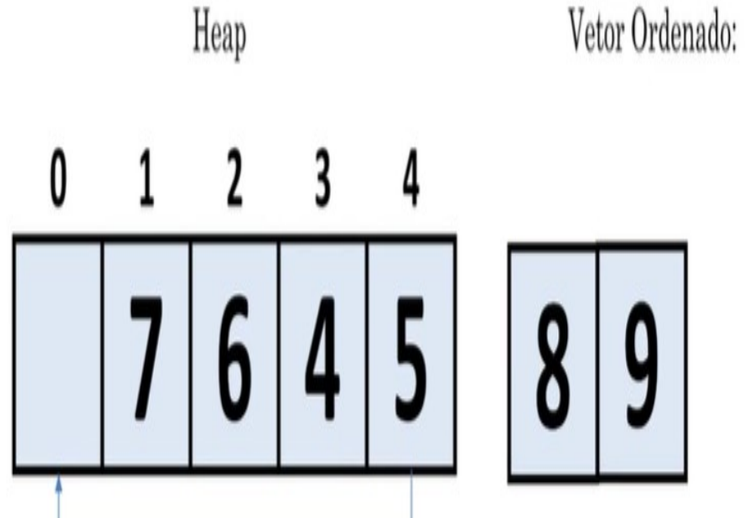
Como o Algoritmo Heap Sort Funciona: Passo a Passo

4

Repetir

Repetir os passos até que o Heap esteja vazio.





Onde o Heap Sort é Utilizado na Prática?

Sistemas Embarcados

Utilizado em sistemas com memória limitada.

Jogos

Em jogos para ordenar elementos com prioridade.

Roteadores

Em roteadores para gerenciamento de tráfego.

Implementação do Heap Sort em Linguagem Python: Exemplo Prático

```python
def heapify(l, n, i):
    maior = i
    esquerda = 2 * i + 1
    direita = 2 * i + 2
    if esquerda < n and l[esquerda] > l[maior]:
        maior = esquerda
    if direita < n and l[direita] > l[maior]:
        maior = direita
    if maior != i:
        l[i], l[maior] = l[maior], l[i]
        heapify(l, n, maior)
def heapSort(l):
    n = len(l)
    for i in range(n // 2 - 1, -1, -1):
        heapify(l, n, i)
    for i in range(n - 1, 0, -1):
        l[i], l[0] = l[0], l[i]
        heapify(l, i, 0)
    return l
```

Conclusão: Recapitulando o Heap Sort

Heap Sort é um algoritmo de ordenação poderoso. Ele oferece bom desempenho e aplicações diversas. Sua complexidade O(n log n) o torna eficiente. Entender seus princípios é valioso para qualquer programador.