Filas de Prioridade

Notas de aula da disciplina IME 04-10820
ESTRUTURAS DE DADOS I

Paulo Eustáquio Duarte Pinto
(pauloedp arroba ime.uerj.br)

julho/2018

Em algumas situações, é necessário manter um conjunto dinâmico de chaves, tal que a principal operação no conjunto é a retirada do elemento de maior prioridade (chave). A manutenção do conjunto ordenado é uma solução para o problema, mas o uso das filas de prioridade fornece uma solução mais simples. Ex: seleção de jobs.

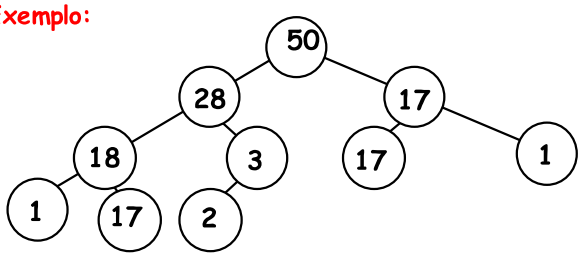
Filas de prioridade são estruturas de dados para as quais se tem operações eficientes para:

- -Seleção do elemento de maior prioridade - $\Theta(1)$
- -Inserção/deleção de elementos - $\Theta(\log n)$
- -Mudanças de prioridade - $\Theta(\log n)$

Heaps

Heaps implementam filas de prioridade. Um Heap é uma árvore binária (virtual!), onde a chave de cada nó é $\geq$ ($\leq$) que a dos descendentes.

Exemplo:



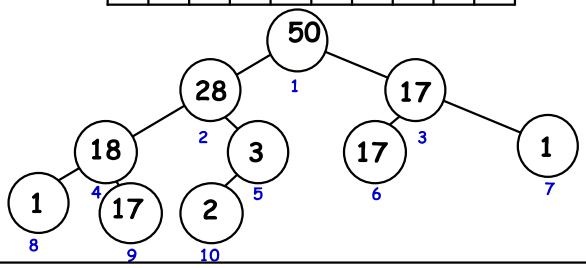
Filas de Prioridade

Heaps

Um Heap é um vetor que pode ser visto como uma árvore binária (virtual!). A raiz é a célula 1 e os filhos do nó i estão nas células 2i e 2i+1, se existirem.

Exemplo:

1	2	3	4	5	6	7	8	9	10
50	28	17	18	3	17	1	1	17	2



Filas de Prioridade

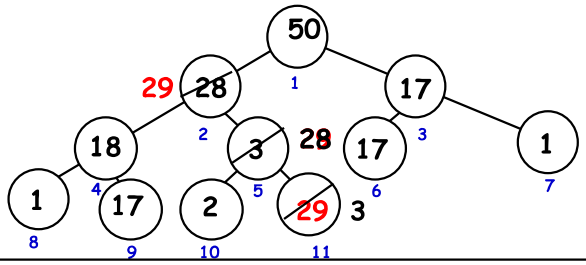
Heaps

Inserção de um novo elemento no Heap.

Solução: inserir no final e executar SOBEHEAP.

Exemplo:

1	2	3	4	5	6	7	8	9	10	11
50	28	17	18	3	17	1	1	17	2	29



Filas de Prioridade

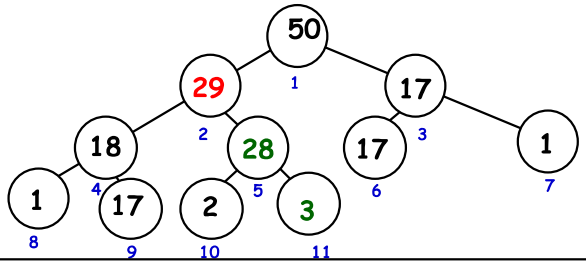
Heaps

Inserção de um novo elemento no Heap.

Solução: inserir no final e executar SOBEHEAP.

Exemplo:

1	2	3	4	5	6	7	8	9	10	11
50	29	17	18	28	17	1	1	17	2	3



Heaps

Inserção de um novo elemento no Heap - SOBEHEAP.

```
SobeHeap(k):
#Dados: Vetor V com n elementos, k inteiro

        t ← V[k]
        V[0] ← t

        enquanto (V[⌊k/2⌋] < t):
                V[k] ← V[⌊k/2⌋]
                k ← ⌊k/2⌋

        V[k] ← t
```

Complexidade: O(log n)

Heaps

Criação de um Heap a partir de um vetor já pre-enchido, usando SOBEHEAP.

```
CriaHeap(inteiro  n):
#Dados: Vetor V com n elementos

    para i <- 2 até n incl.:
        SobeHeap(i)
```

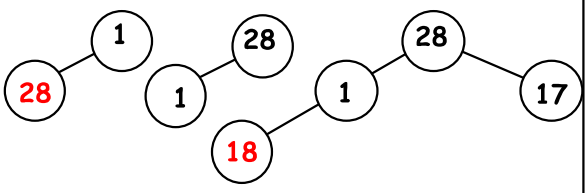
Complexidade: O(n log n)

Heaps

Criação de um Heap a partir de um vetor já pre-enchido, usando SOBEHEAP.

Exemplo:

1	2	3	4	5	6	7	8	9	10
1	28	17	18	2	17	1	17	50	3



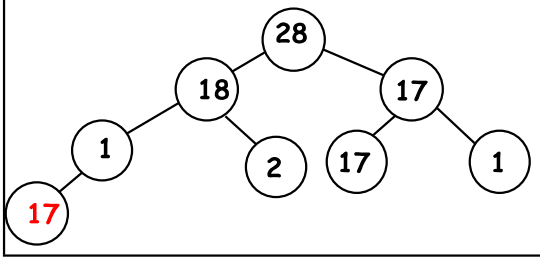
Filas de Prioridade

Heaps

Criação de um Heap a partir de um vetor já pre-enchido, usando SOBEHEAP.

Exemplo:

1	2	3	4	5	6	7	8	9	10
28	18	17	1	2	17	1	17	50	3



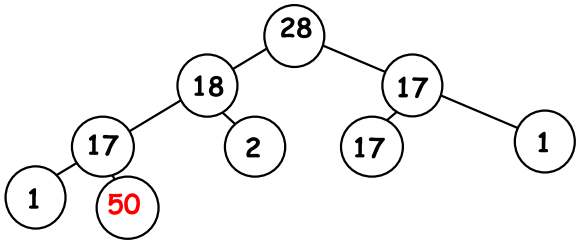
Filas de Prioridade

Heaps

Criação de um Heap a partir de um vetor já pre-enchido, usando SOBEHEAP.

Exemplo:

1	2	3	4	5	6	7	8	9	10
28	18	17	17	2	17	1	1	50	3



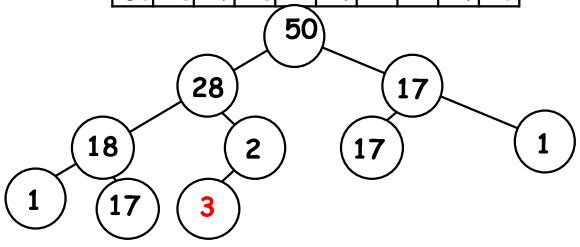
Filas de Prioridade

Heaps

Criação de um Heap a partir de um vetor já pre-enchido, usando SOBEHEAP.

Exemplo:

1	2	3	4	5	6	7	8	9	10
50	28	17	18	2	17	1	1	17	3

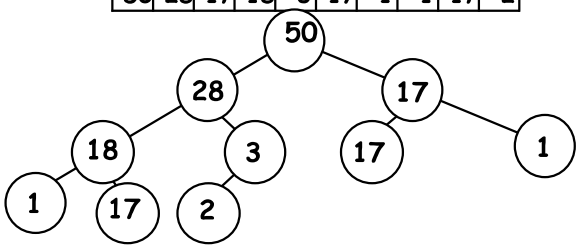


Heaps

Criação de um Heap a partir de um vetor já pre-enchido, usando SOBEHEAP.

Exemplo:

1	2	3	4	5	6	7	8	9	10
50	28	17	18	3	17	1	1	17	2



Heaps

Criação de um Heap a partir de um vetor já pre-enchido, usando SOBEHEAP.

Exercício: Criar um Heap a partir do vetor preenchido com o MIXTRING (10 letras) usando SobeHeap

Heaps

Complexidade da criação de um Heap a partir de um vetor já preenchido, usando SOBEHEAP.

NC $= \Sigma_{1 \le i \le (h-1)} i.2^i = \Sigma_{1 \le i \le (h-1)} \Sigma_{i \le j \le (h-1)} 2^j$

$= \Sigma_{1 \le i \le (h-1)} (\Sigma_{0 \le j \le (h-1)} 2^j - \Sigma_{0 \le j \le (i-1)} 2^j)$

$= \Sigma_{1 \le i \le (h-1)} (2^h - 1 - (2^i - 1))$

$= 2^h \Sigma_{1 \le i \le (h-1)} 1 - \Sigma_{1 \le i \le (h-1)} 2^i$

$= 2^h(h-1) - (2^h - 2) = 2^h(h-2) + 2$

Quando $(n+1)$ é potência de 2, temos $2^h = (n+1)$

NC $= (n+1)(\log_2(n+1) - 2) + 2 = (n+1)\log_2(n+1) - 2n$

$= O(n \log n)$

Filas de Prioridade

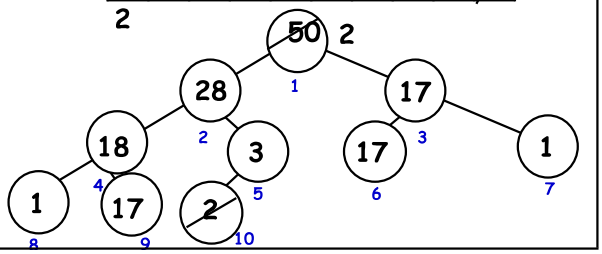
Heaps

Deleção do elemento de maior prioridade no Heap.

Solução: substituir o primeiro pelo último, eliminar no final e executar DESCEHEAP.

Exemplo:

1	2	3	4	5	6	7	8	9	~~10~~
~~50~~ 2	28	17	18	3	17	1	1	17	~~2~~



Filas de Prioridade

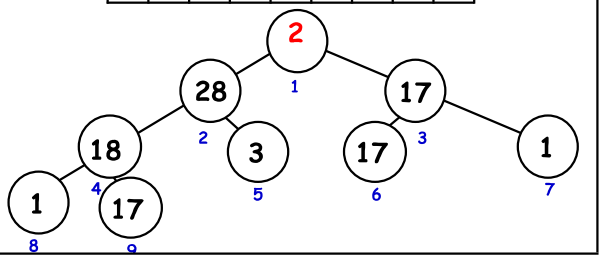
Heaps

Deleção do elemento de maior prioridade no Heap.

Solução: substituir o primeiro pelo último, eliminar no final e executar DESCEHEAP.

Exemplo:

1	2	3	4	5	6	7	8	9
2	28	17	18	3	17	1	1	17



Filas de Prioridade

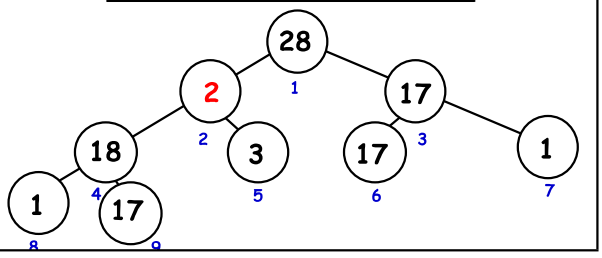
Heaps

Deleção do elemento de maior prioridade no Heap.

Solução: substituir o primeiro pelo último, eliminar no final e executar DESCEHEAP.

Exemplo:

1	2	3	4	5	6	7	8	9
28	2	17	18	3	17	1	1	17



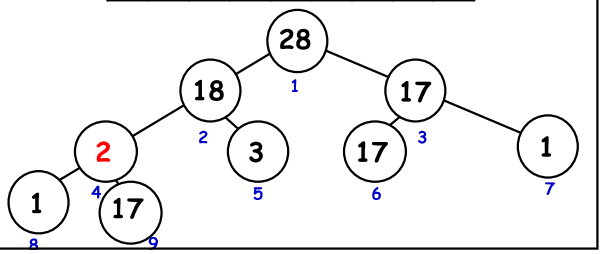
Heaps

Deleção do elemento de maior prioridade no Heap.

Solução: substituir o primeiro pelo último, eliminar no final e executar DESCEHEAP.

Exemplo:

1	2	3	4	5	6	7	8	9
28	18	17	2	3	17	1	1	17



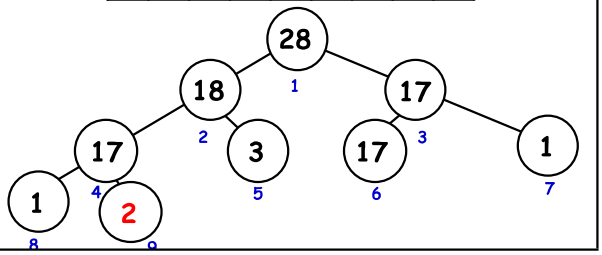
Heaps

Deleção do elemento de maior prioridade no Heap.

Solução: substituir o primeiro pelo último, eliminar no final e executar DESCEHEAP.

Exemplo:

1	2	3	4	5	6	7	8	9
28	18	17	17	3	17	1	1	2



Heaps

Diminuição da prioridade de um elemento- DESCEHEAP.

```
DesceHeap(inteiro  k, inteiro  m):
#Dados:  Vetor  V com m elementos,  k inteiro
        t ← V[k]

        enquanto (k ≤ ⌊m/2⌋):
                j ← 2*k
                se ((j < m) e (V[j] < V[j+1])):
                        j ← j+1
                se (t ≥ V[j]):
                        parar
                senão:
                        V[k] ← V[j]
                        k ← j

        V[k] ← t
```

Heaps

Criação de um Heap a partir de um vetor já pre-enchido, usando DESCEHEAP.

```
CriaHeap(inteiro  n):
#Dados:  Vetor  V com n elementos
     para  i <- ⌊n/2⌋ até 1 incl.:
          DesceHeap(i,  n)
```

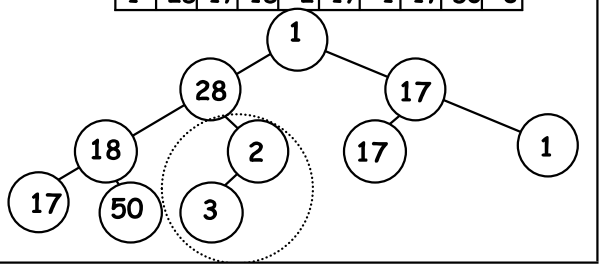
Complexidade: Θ(n)!!

Heaps

Criação de um Heap a partir de um vetor já pre-enchido, usando DESCEHEAP.

Exemplo:

1	2	3	4	5	6	7	8	9	10
1	28	17	18	2	17	1	17	50	3

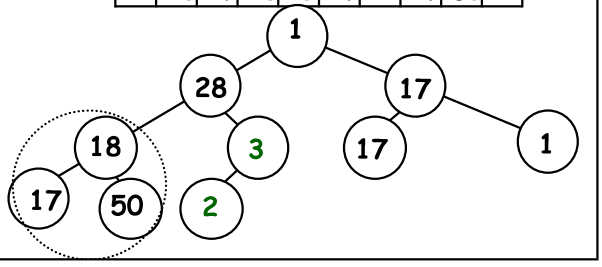


Heaps

Criação de um Heap a partir de um vetor já pre-enchido, usando DESCEHEAP.

Exemplo:

1	2	3	4	5	6	7	8	9	10
1	28	17	18	3	17	1	17	50	2

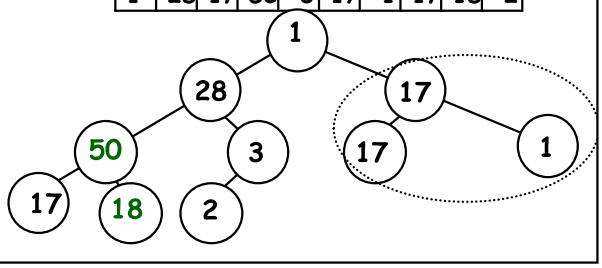


Heaps

Criação de um Heap a partir de um vetor já pre-enchido, usando DESCEHEAP.

Exemplo:

1	2	3	4	5	6	7	8	9	10
1	28	17	50	3	17	1	17	18	2

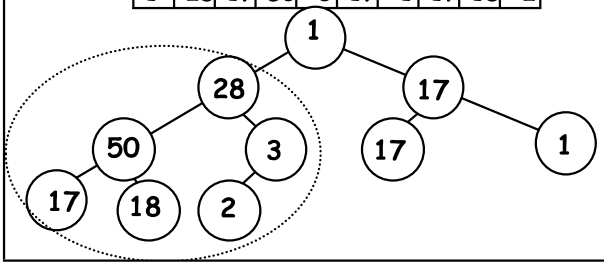


Heaps

Criação de um Heap a partir de um vetor já pre-enchido, usando DESCEHEAP.

Exemplo:

1	2	3	4	5	6	7	8	9	10
1	28	17	50	3	17	1	17	18	2

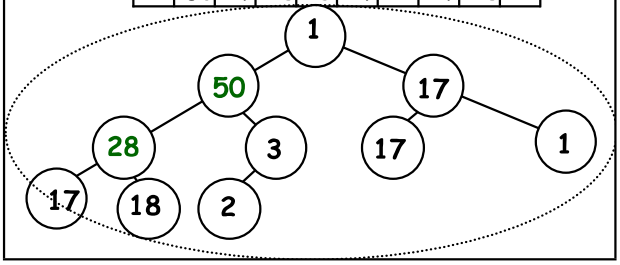


Heaps

Criação de um Heap a partir de um vetor já pre-enchido, usando DESCEHEAP.

Exemplo:

1	2	3	4	5	6	7	8	9	10
1	50	17	28	3	17	1	17	18	2

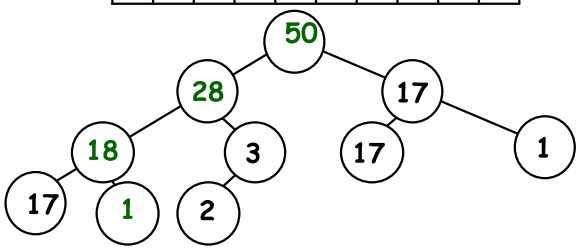


Heaps

Criação de um Heap a partir de um vetor já pre-enchido, usando DESCEHEAP.

Exemplo:

1	2	3	4	5	6	7	8	9	10
50	28	17	18	3	17	1	17	1	2



Heaps

Criação de um Heap a partir de um vetor já pre-enchido, usando DESCEHEAP.

Exercício: Criar um Heap a partir do vetor preenchido com o MIXTRING (10 letras) usando DesceHeap.

Heaps

Complexidade da criação de um Heap a partir de um vetor já preenchido, usando DESCEHEAP.

$NC = 2\Sigma_{1 \le i \le (h-1)} 2^{i-1}(h - i)$
$= 2(\Sigma_{1 \le i \le (h-1)} h.2^{i-1} - \Sigma_{1 \le i \le (h-1)} i.2^{i-1})$
$= 2(h\Sigma_{1 \le i \le (h-1)} 2^{i-1} - (1/2)\Sigma_{i1 \le i \le (h-1)} i.2^{i})$
$= 2(h(2^{h-1} - 1) - (2^{h-1} - (h - 2) + 1))$
$= 2(h.2^{h-1} - h - h.2^{h-1} - h + 2^{h} - 1)$
$= 2(2^{h} - h - 1) \quad = 2^{h+1} - 2.h - 2$

Quando $(n + 1)$ é potência de 2, temos $2^h = (n + 1)$

$NC = 2((n+1) - \log_2(n + 1) - 1) =$
$= 2(n+1 - \log_2(n + 1) - 1) =$
$= 2(n - \log_2(n + 1)) =$
$= \Theta(n)$

Heaps

Operações em um Heap:

Inserção:

Inserção no final do vetor + SOBEHEAP.

Retirada do elemento de maior prioridade:

Substituição pelo último elemento + DESCEHEAP.

Modificação de prioridade:

SOBEHEAP ou DESCEHEAP.

Deleção:

Substituição pelo último elemento + Modificação de prioridade.

Heapsort

Idéia: criar um Heap e, sucessivamente, trocar o primeiro elemento com o último, diminuir o Heap e acertá-lo.

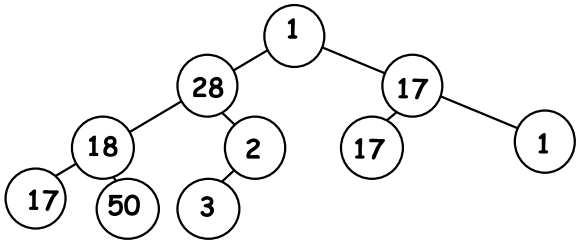
```
Heapsort(inteiro  n):
#Dados:  Vetor V com n elementos
        CriaHeap(n)

        para i <- n até 2 incl.:
                Troca (1, i);
                DesceHeap(1,  i-1);
```


Heapsort

Exemplo:

1	2	3	4	5	6	7	8	9	10
1	28	17	18	2	17	1	17	50	3



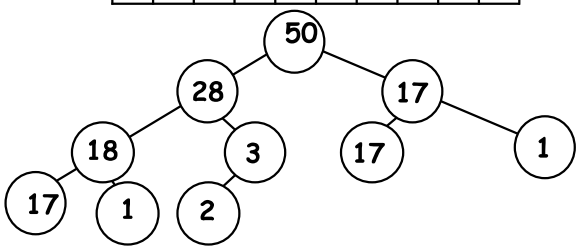
Filas de Prioridade

Heapsort

Passo 1: Criação do Heap.

Exemplo:

1	2	3	4	5	6	7	8	9	10
50	28	17	18	3	17	1	17	1	2



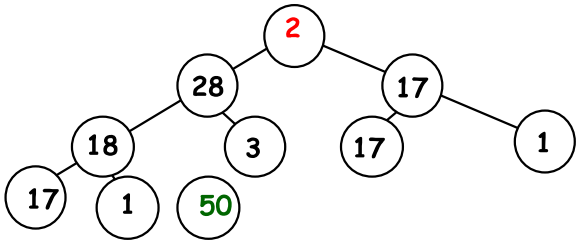
Filas de Prioridade

Heapsort

Passo 2: i = 10 → Troca primeiro com último.

Exemplo:

1	2	3	4	5	6	7	8	9	*10*
2	28	17	18	3	17	1	17	1	*50*



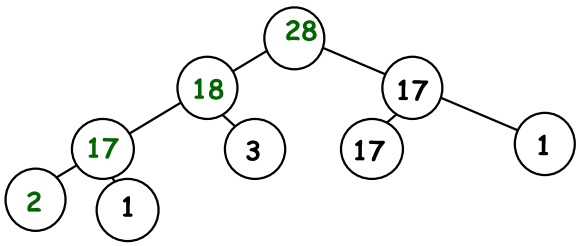
Filas de Prioridade

Heapsort

Passo 2: i = 10 → DesceHeap(1,9).

Exemplo:

1	2	3	4	5	6	7	8	9	*10*
28	18	17	17	3	17	1	2	1	*50*



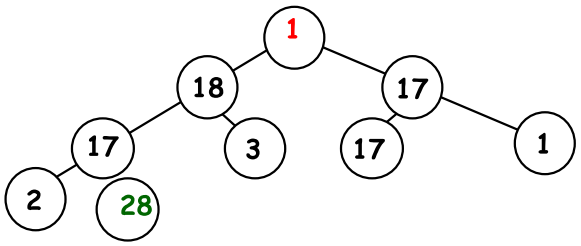
Filas de Prioridade

Heapsort

Passo 2: i = 9 → Troca primeiro com último.

Exemplo:

1	2	3	4	5	6	7	8	9	10
1	18	17	17	3	17	1	2	28	50



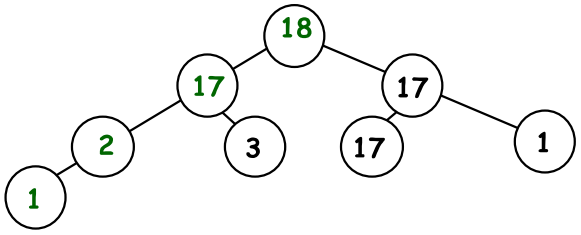
Filas de Prioridade

Heapsort

Passo 2: i = 9 → DesceHeap(1,8).

Exemplo:

1	2	3	4	5	6	7	8	9	10
18	17	17	2	3	17	1	1	28	50



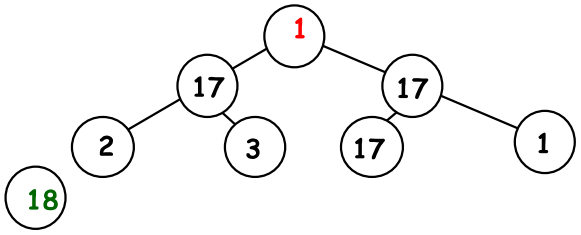
Filas de Prioridade

Heapsort

Passo 2: i = 8 → Troca primeiro com último.

Exemplo:

1	2	3	4	5	6	7	8	9	10
1	17	17	2	3	17	1	18	28	50

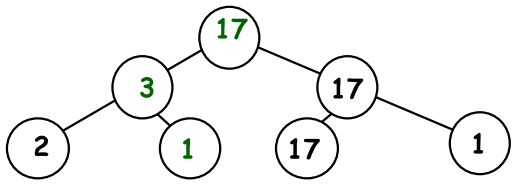


Heapsort

Passo 2: i = 8 → DesceHeap(1,7).

Exemplo:

1	2	3	4	5	6	7	8	9	10
17	3	17	2	1	17	1	18	28	50

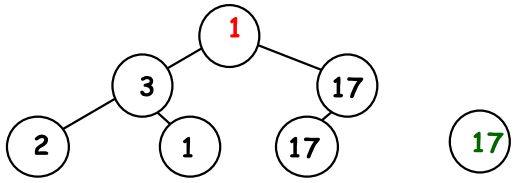


Heapsort

Passo 2: i = 7 → Troca primeiro com último.

Exemplo:

1	2	3	4	5	6	7	8	9	10
1	3	17	2	1	17	17	18	28	50

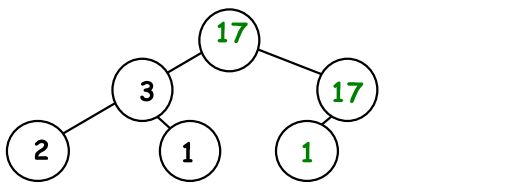


Heapsort

Passo 2: i = 7 → DesceHeap(1,6).

Exemplo:

1	2	3	4	5	6	7	8	9	10
17	3	17	2	1	1	17	18	28	50

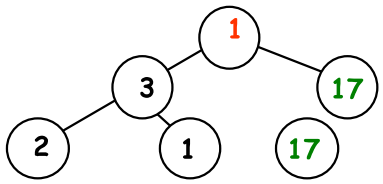


Heapsort

Passo 2: i = 6 → Troca primeiro com último.

Exemplo:

1	2	3	4	5	6	7	8	9	10
1	3	17	2	1	17	17	18	28	50

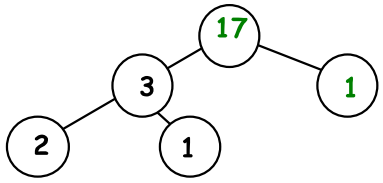


Heapsort

Passo 2: i = 6 → DesceHeap(1,5).

Exemplo:

1	2	3	4	5	6	7	8	9	10
17	3	1	2	1	17	17	18	28	50

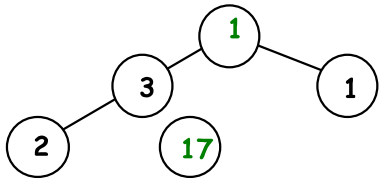


Heapsort

Passo 2: i = 5 → Troca primeiro com último.

Exemplo:

1	2	3	4	5	6	7	8	9	10
1	3	1	2	17	17	17	18	28	50

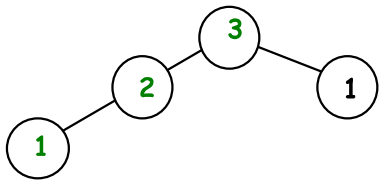


Heapsort

Passo 2: i = 5 → DesceHeap(1,4).

Exemplo:

1	2	3	4	5	6	7	8	9	10
3	2	1	1	17	17	17	18	28	50

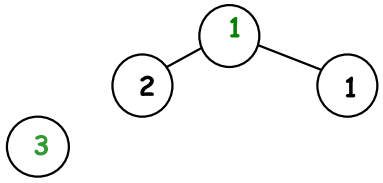


Heapsort

Passo 2: i = 4 → Troca primeiro com último.

Exemplo:

1	2	3	4	5	6	7	8	9	10
1	2	1	3	17	17	17	18	28	50

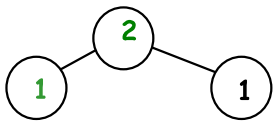


Heapsort

Passo 2: i = 4 → DesceHeap(1,3).

Exemplo:

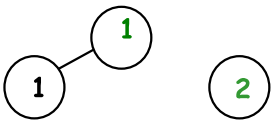
1	2	3	4	5	6	7	8	9	10
2	1	1	3	17	17	17	18	28	50



Heapsort

Passo 2: i = 3 → Troca primeiro com último.

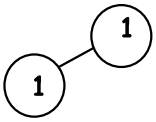
	1	2	3	4	5	6	7	8	9	10
Exemplo:	1	1	2	3	17	17	17	18	28	50



Heapsort

Passo 2: i = 3 → DesceHeap(1,2).

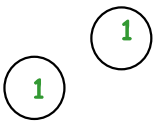
	1	2	3	4	5	6	7	8	9	10
Exemplo:	1	1	2	3	17	17	17	18	28	50



Heapsort

Passo 2: i = 2 → Troca primeiro com último.

	1	2	3	4	5	6	7	8	9	10
Exemplo:	1	1	2	3	17	17	17	18	28	50



Heapsort

Passo 2: i = 2 → DesceHeap (1,1).

Exemplo:

1	2	3	4	5	6	7	8	9	10
1	1	2	3	17	17	17	18	28	50

Heapsort

HEAPSORT

Exercício: Ordenar o MIXTRING (10 letras) usando Heapsort.

Heapsort

Análise do HEAPSORT:

Complexidade:

Pior caso: O(n log n) vetor inv. ordenado
Melhor caso: O(n) chaves iguais

Estabilidade (manutenção da ordem relativa de chaves iguais):

Algoritmo não estável

Memória adicional:

Nenhuma

Usos especiais:

Algoritmo de uso geral

FIM