AEDs I - Listas de Prioridades - Heap

Prof. Jurair Rosa

Engenharia de Computação
3^o Período

CEFET/RJ - Centro Federal de Educação Tecnológica Celso Suckow da Fonseca

Campus Petrópolis

Conteúdo cedido gentilmente pelo Prof André Monteiro – CEFET/RJ, apresentado com algumas modificações.

Sumário

Listas de Prioridade

- 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 listas de prioridade fornece uma solução mais simples. Ex: seleção de jobs.

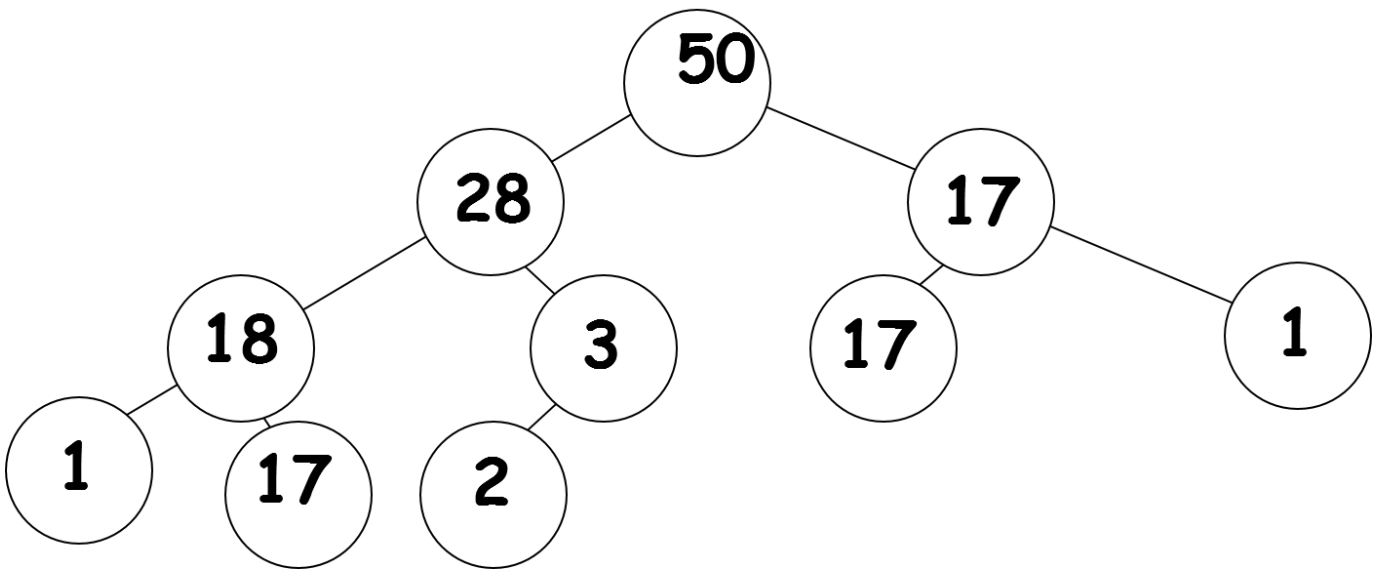
Definição

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

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

Listas de Prioridade - Heaps

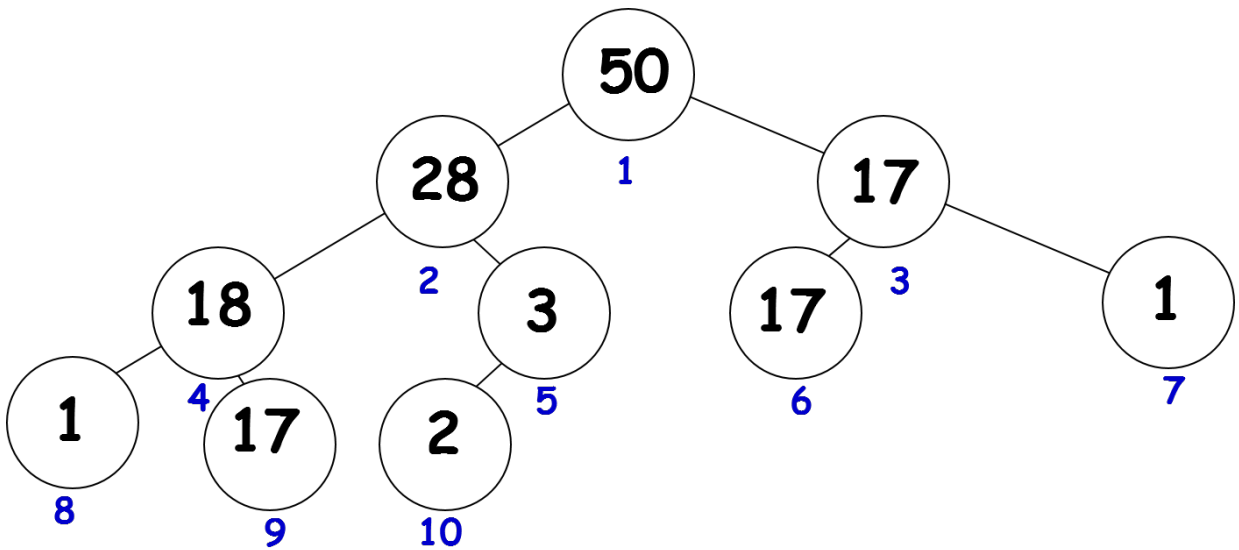
- Heaps implementam listas de prioridade. Um Heap é uma árvore binária (virtual!), onde a chave de cada nó é $\geq$ (ou $\leq$) que a dos descendentes



Listas 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

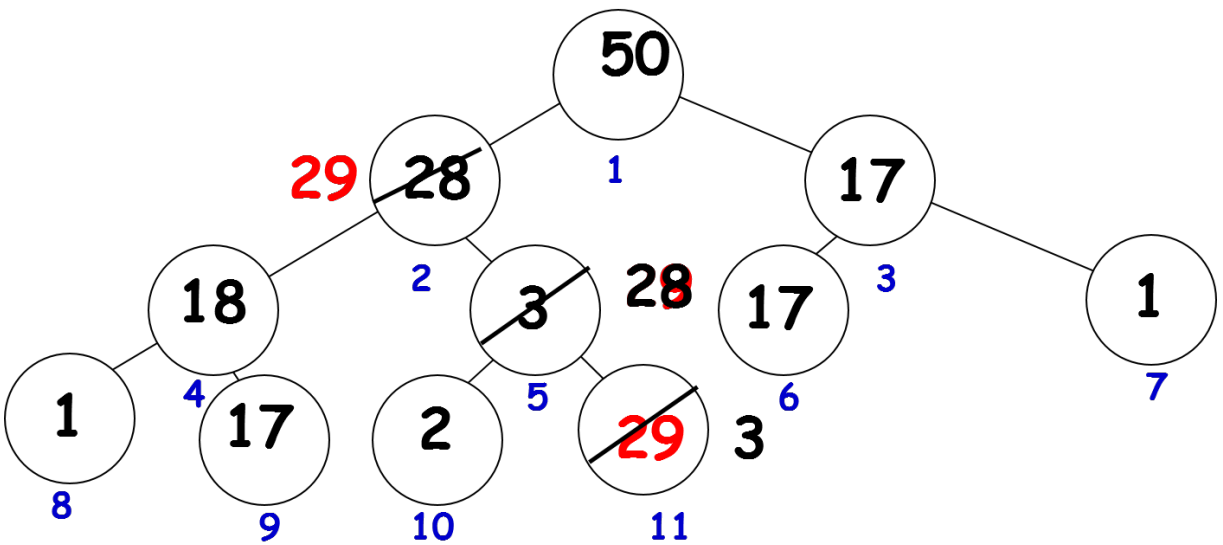
1	2	3	4	5	6	7	8	9	10
50	28	17	18	3	17	1	1	17	2



Listas de Prioridade - Heaps

- Inserção de um novo elemento no Heap
- Solução: inserir no final e executar SOBEHEAP

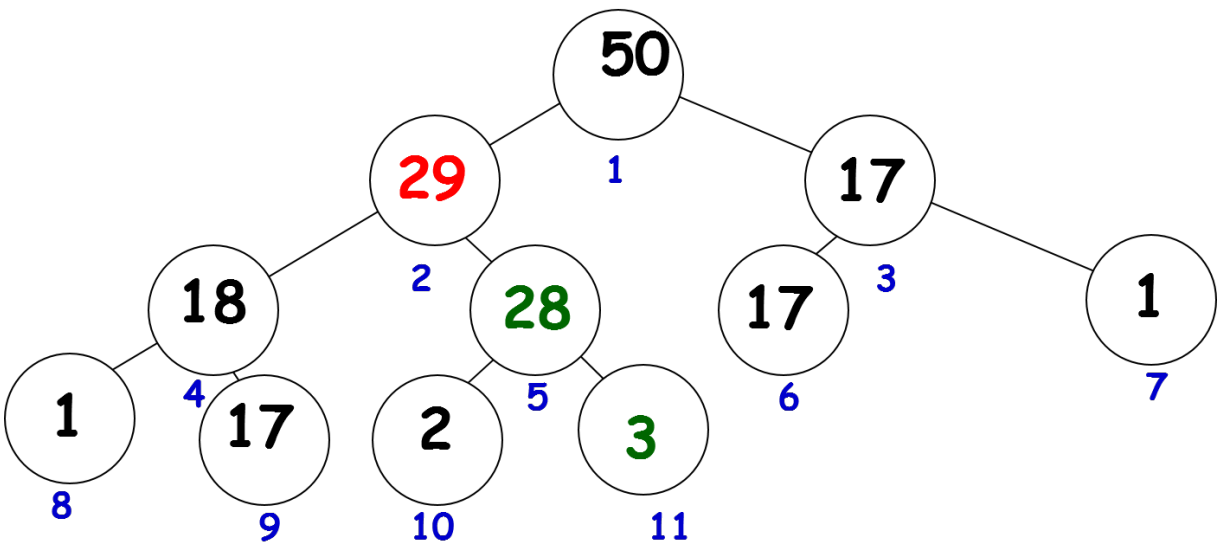
1	2	3	4	5	6	7	8	9	10	11
50	28	17	18	3	17	1	1	17	2	29



Listas de Prioridade - Heaps

- Inserção de um novo elemento no Heap
- Solução: inserir no final e executar SOBEHEAP

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



Listas de Prioridade - Heaps

- Inserção de um novo elemento no Heap, aplicando a função SOBEHEAP

```
SobeHeap(k):
  t ← V[k];
  V[0] ← t;
  Enquanto (V[⌊k/2⌋] < t):
    V[k] ← V[⌊k/2⌋];
    k ← ⌊k/2⌋;
  Fe;
  V[k] ← t;
Fim;
```

Listas de Prioridade - Heaps

- Criação de um Heap a partir de um vetor já preenchido, usando SOBEHEAP

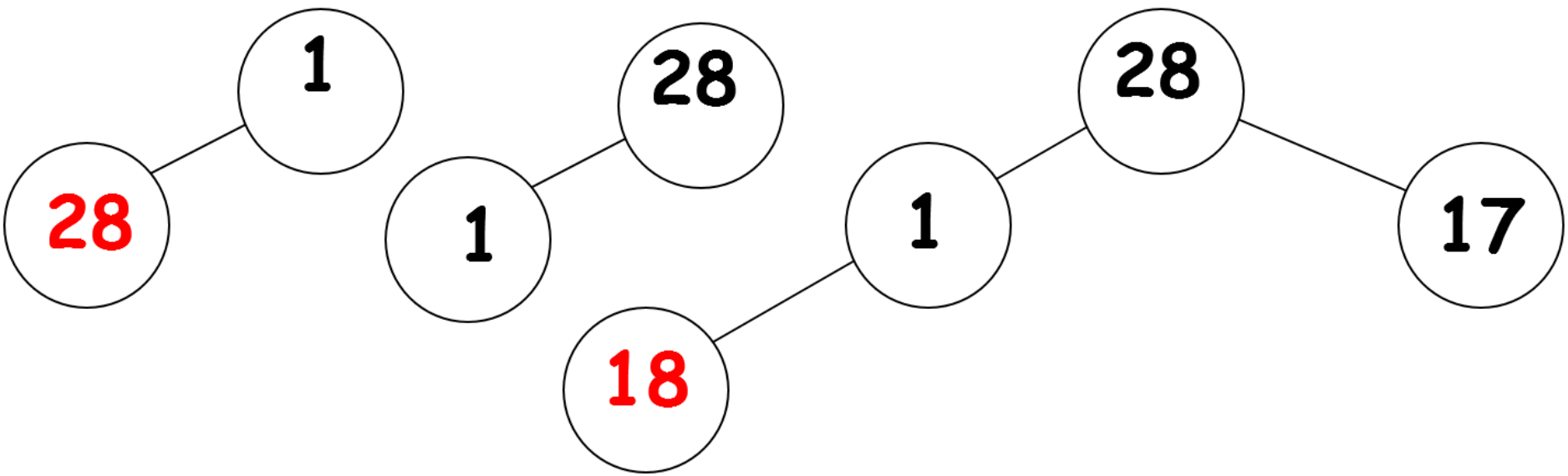
```
CriaHeap:
  Para i de 2 a n:
    SobeHeap(i);
  Fp;
Fim;
```

- Complexidade - $O(n \ log \ n)$

Listas de Prioridade - Heaps

- Criação de um Heap a partir de um vetor já preenchido, usando SOBEHEAP

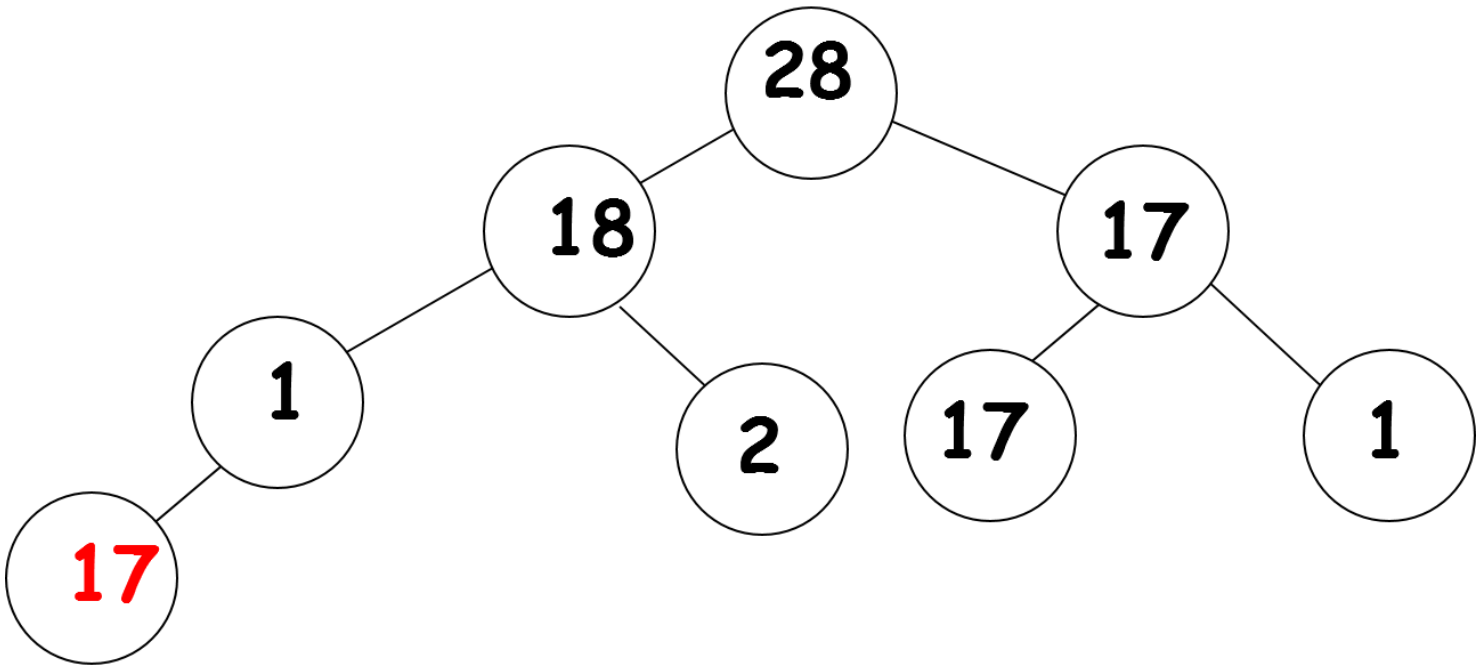
1	2	3	4	5	6	7	8	9	10
1	28	17	18	2	17	1	17	50	3



Listas de Prioridade - Heaps

- Criação de um Heap a partir de um vetor já preenchido, usando SOBEHEAP

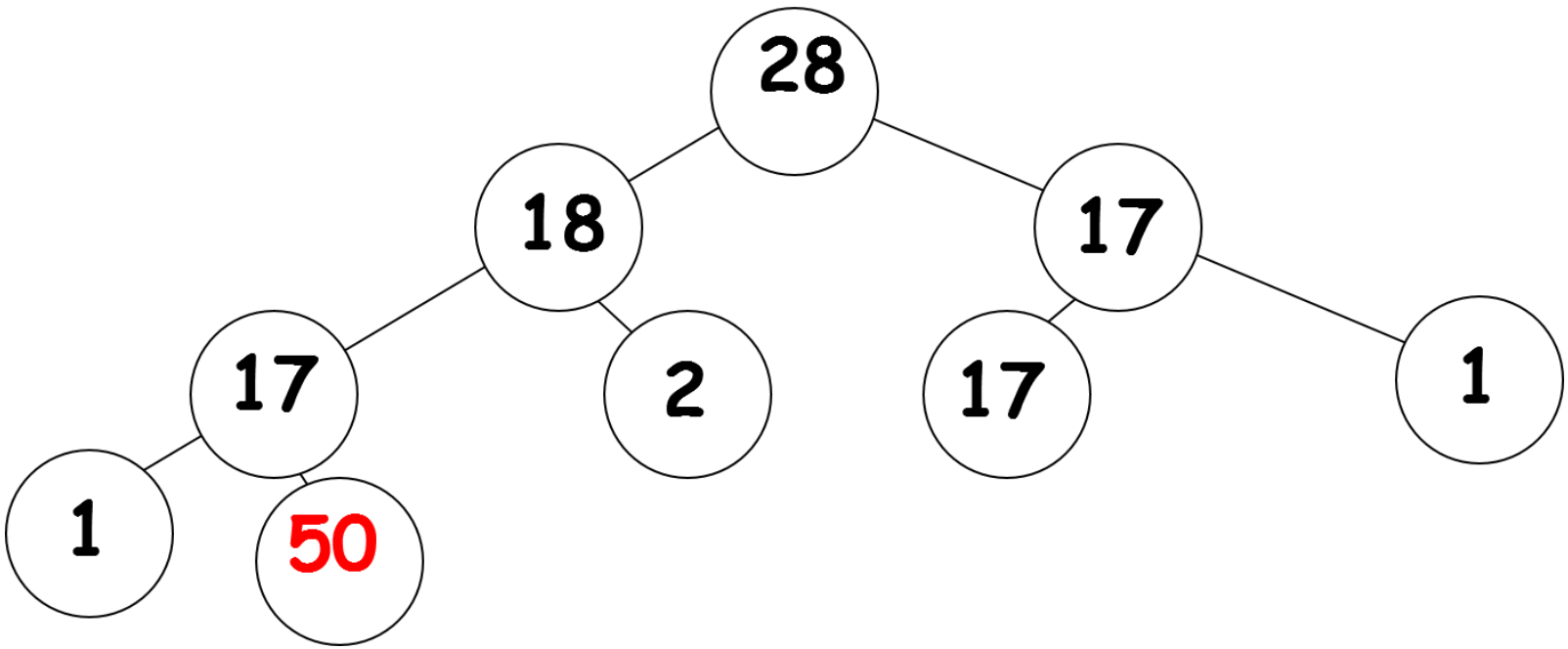
1	2	3	4	5	6	7	8	9	10
28	18	17	1	2	17	1	17	50	3



Listas de Prioridade - Heaps

- Criação de um Heap a partir de um vetor já preenchido, usando SOBEHEAP

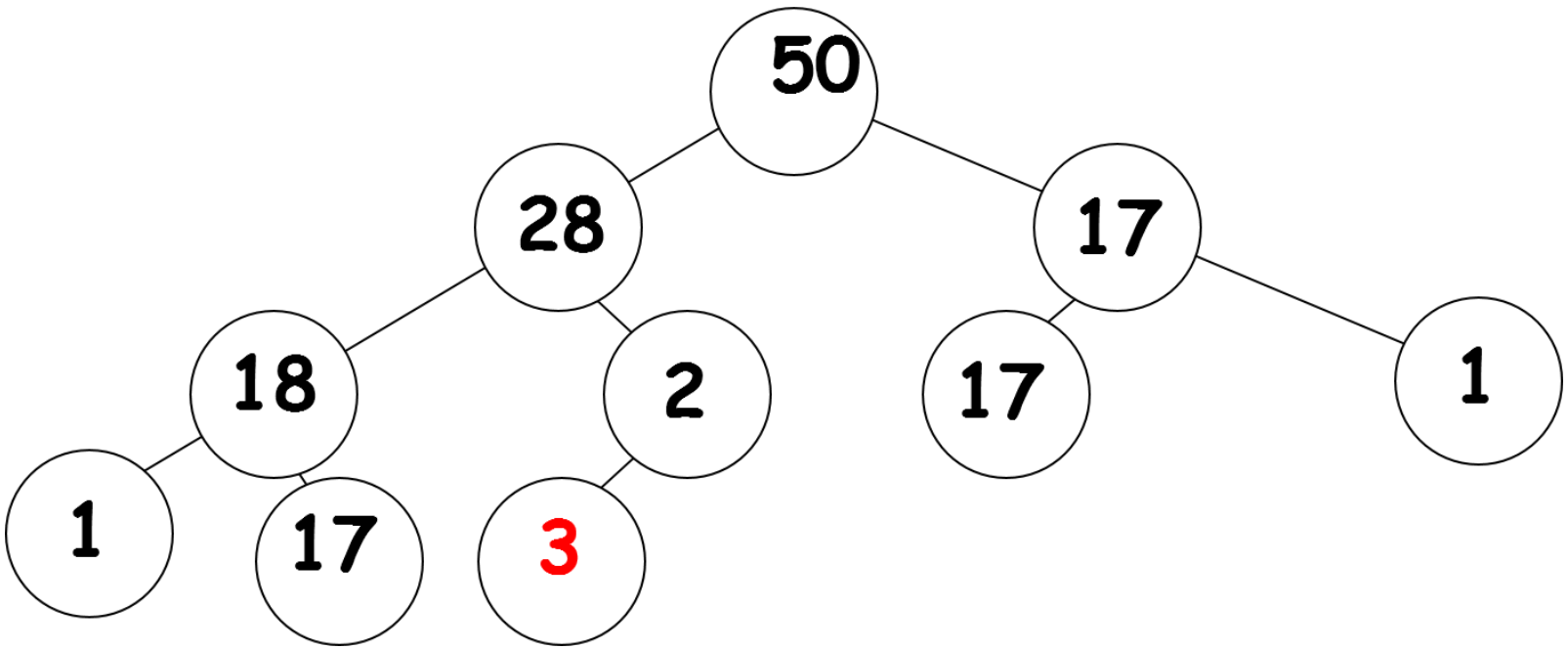
1	2	3	4	5	6	7	8	9	10
28	18	17	1	2	17	17	1	50	3



Listas de Prioridade - Heaps

- Criação de um Heap a partir de um vetor já preenchido, usando SOBEHEAP

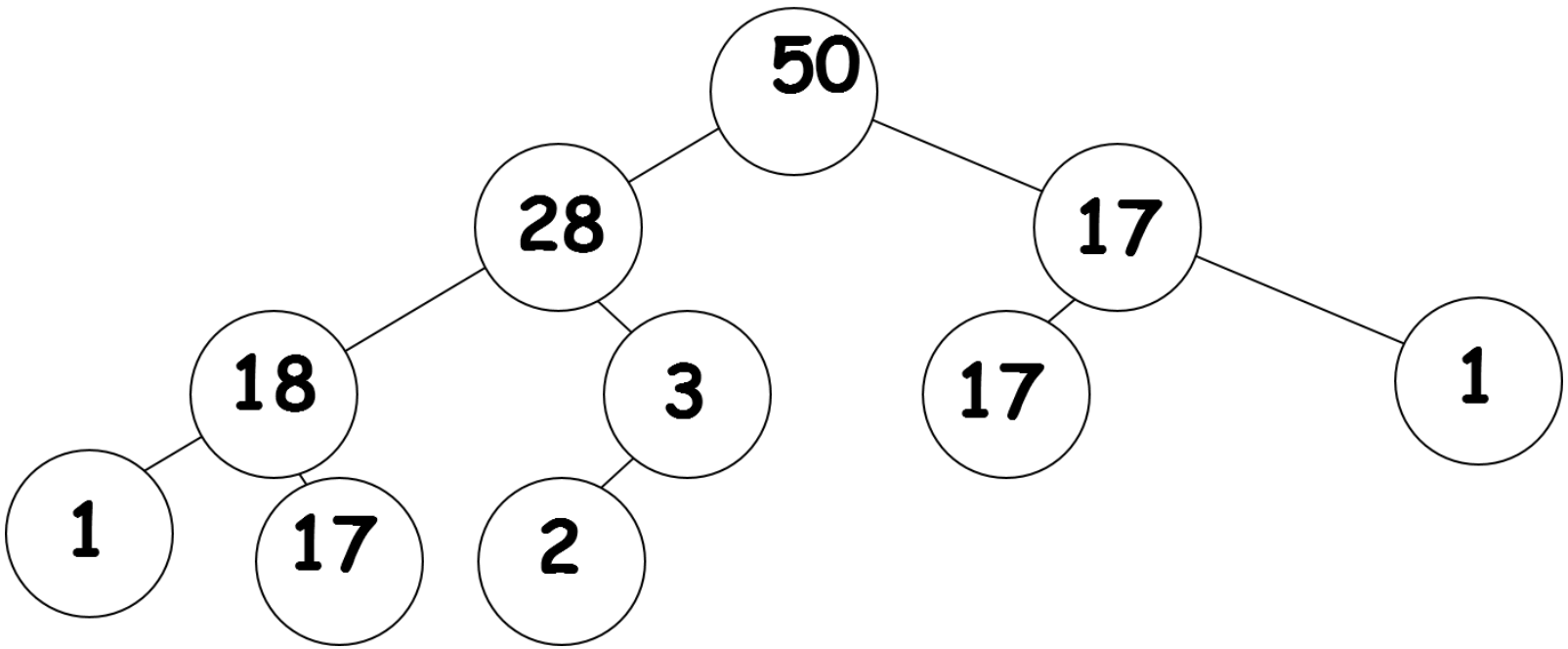
1	2	3	4	5	6	7	8	9	10
50	28	17	18	2	17	1	1	17	3



Listas de Prioridade - Heaps

- Criação de um Heap a partir de um vetor já preenchido, usando SOBEHEAP

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



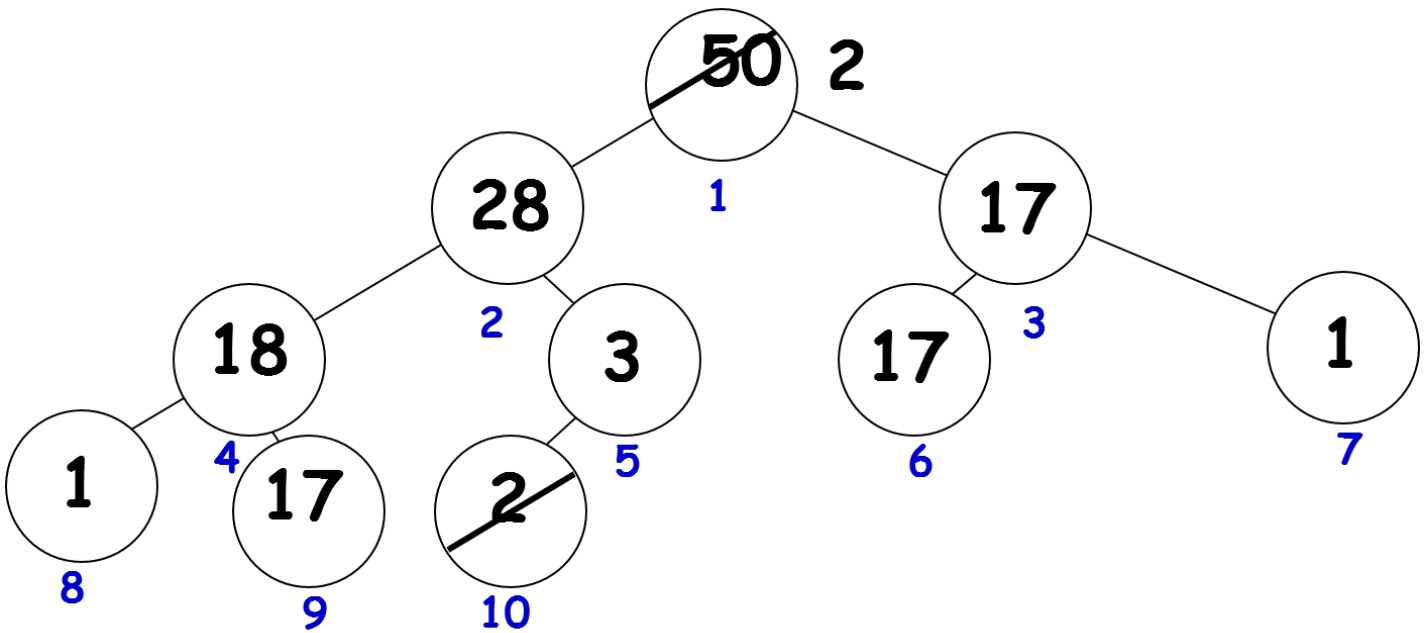
Listas de Prioridade - Heaps

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

Listas de Prioridade - Heaps

- Deleção do elemento de maior prioridade no Heap
- Solução: substituir o primeiro pelo último, eliminar no final e executar DESCEHEAP

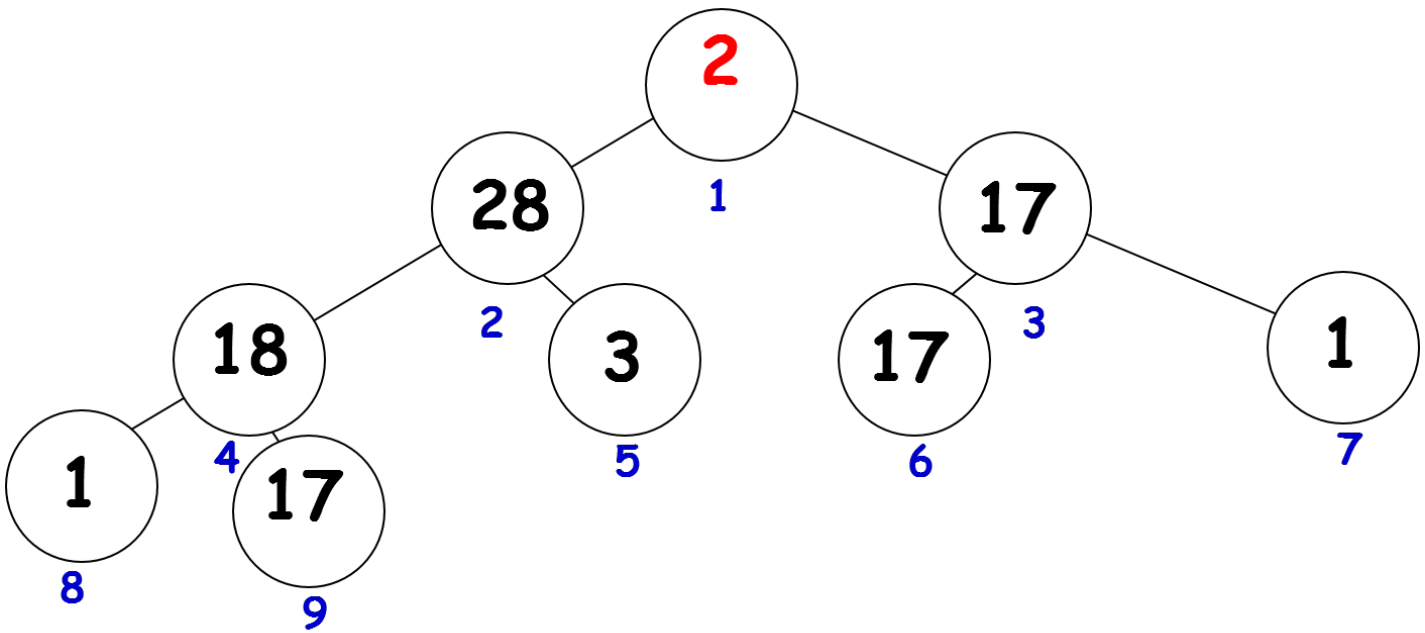
1	2	3	4	5	6	7	8	9	10
50	28	17	18	3	17	1	1	17	2



Listas de Prioridade - Heaps

- Deleção do elemento de maior prioridade no Heap
- Solução: substituir o primeiro pelo último, eliminar no final e executar DESCEHEAP

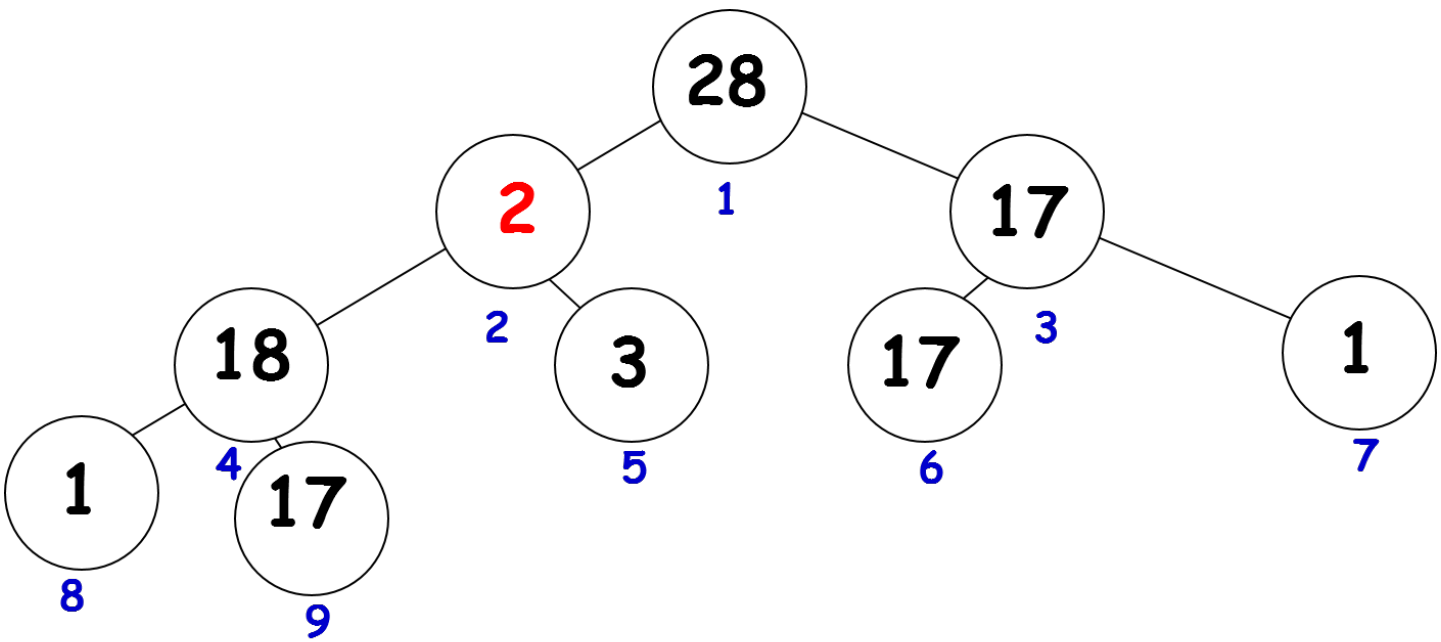
1	2	3	4	5	6	7	8	9
2	28	17	18	3	17	1	1	17



Listas de Prioridade - Heaps

- Deleção do elemento de maior prioridade no Heap
- Solução: substituir o primeiro pelo último, eliminar no final e executar DESCEHEAP

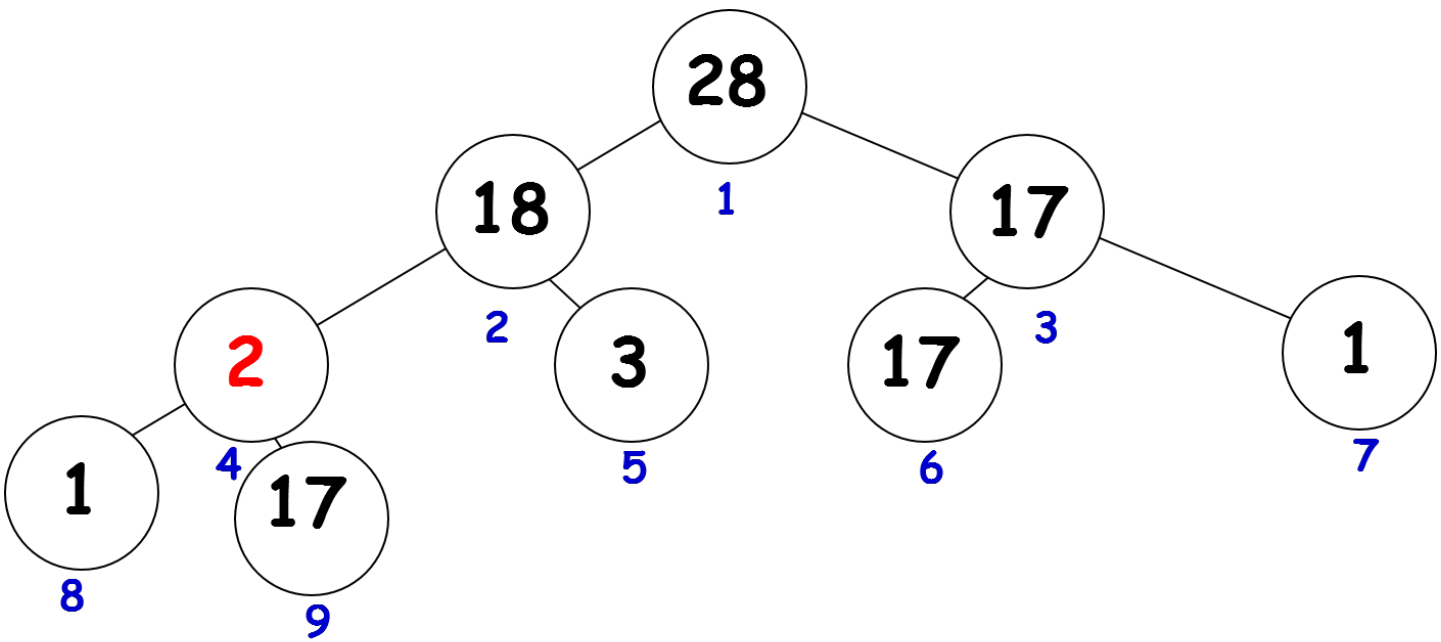
1	2	3	4	5	6	7	8	9
28	2	17	18	3	17	1	1	17



Listas de Prioridade - Heaps

- Deleção do elemento de maior prioridade no Heap
- Solução: substituir o primeiro pelo último, eliminar no final e executar DESCEHEAP

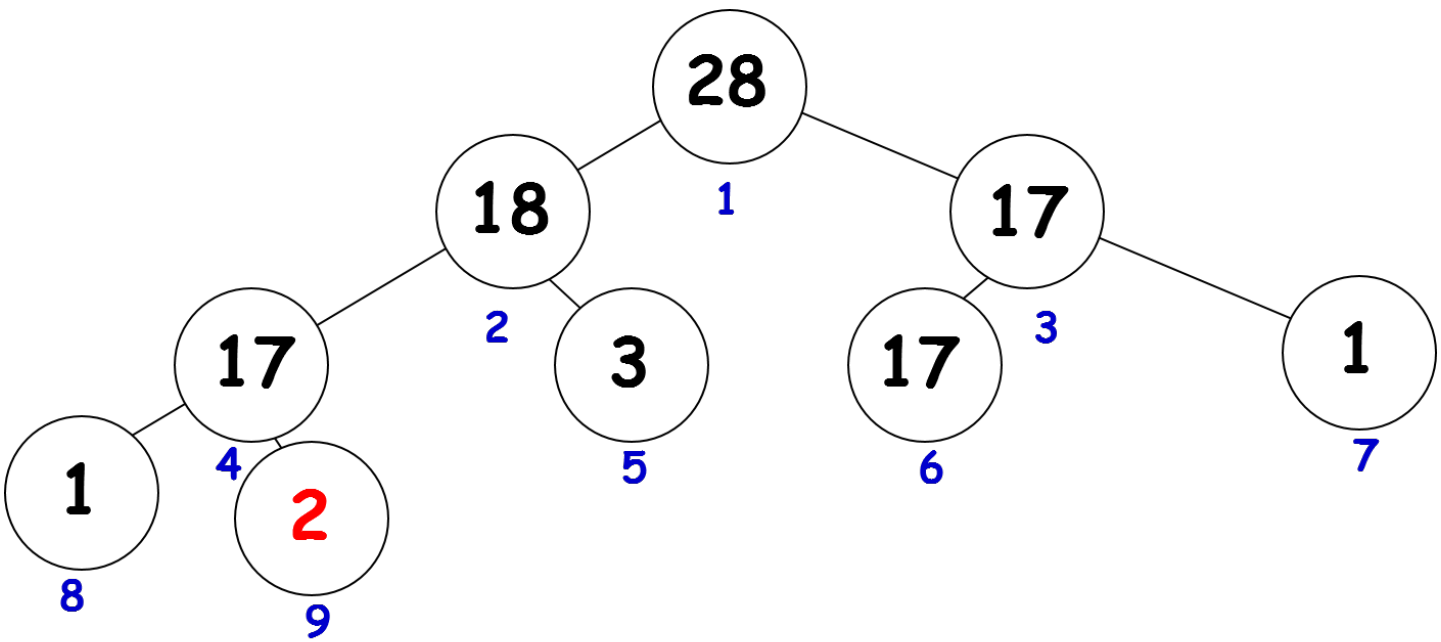
1	2	3	4	5	6	7	8	9
28	18	17	2	3	17	1	1	17



Listas de Prioridade - Heaps

- Deleção do elemento de maior prioridade no Heap
- Solução: substituir o primeiro pelo último, eliminar no final e executar DESCEHEAP

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



Listas de Prioridade - Heaps

- Diminuição da prioridade de um elemento, aplicando a função DESCEHEAP

```
DesceHeap(k, m): // k = pos. do nó pai; m = tam. vetor
  t ← V[k]; // t = conteúdo do nó pai
  x ← (k ≤ ⌊m/2⌋); // Verifica se tem filho(s)
  Enquanto (x):
    j ← 2*k; // Calcula filho da esq.
    Se (j < m) e (V[j] < V[j+1]) Então
      j ← j+1; // Filho dir. existe; é maior que esq.
    Se (t ≥ V[j]) Então
      x ← false; // Filho selecionado menor que pai
    Senão
      V[k] ← V[j]; // Filho maior que pai; Troca!
      k ← j; // Atualiza pos. do nó pai
      x ← (k ≤ ⌊m/2⌋); // Verifica se tem filho(s)
  Fe;
  V[k] ← t;
Fim;
```

Listas de Prioridade - Heaps

- Criação de um Heap a partir de um vetor já preenchido, usando DESCEHEAP

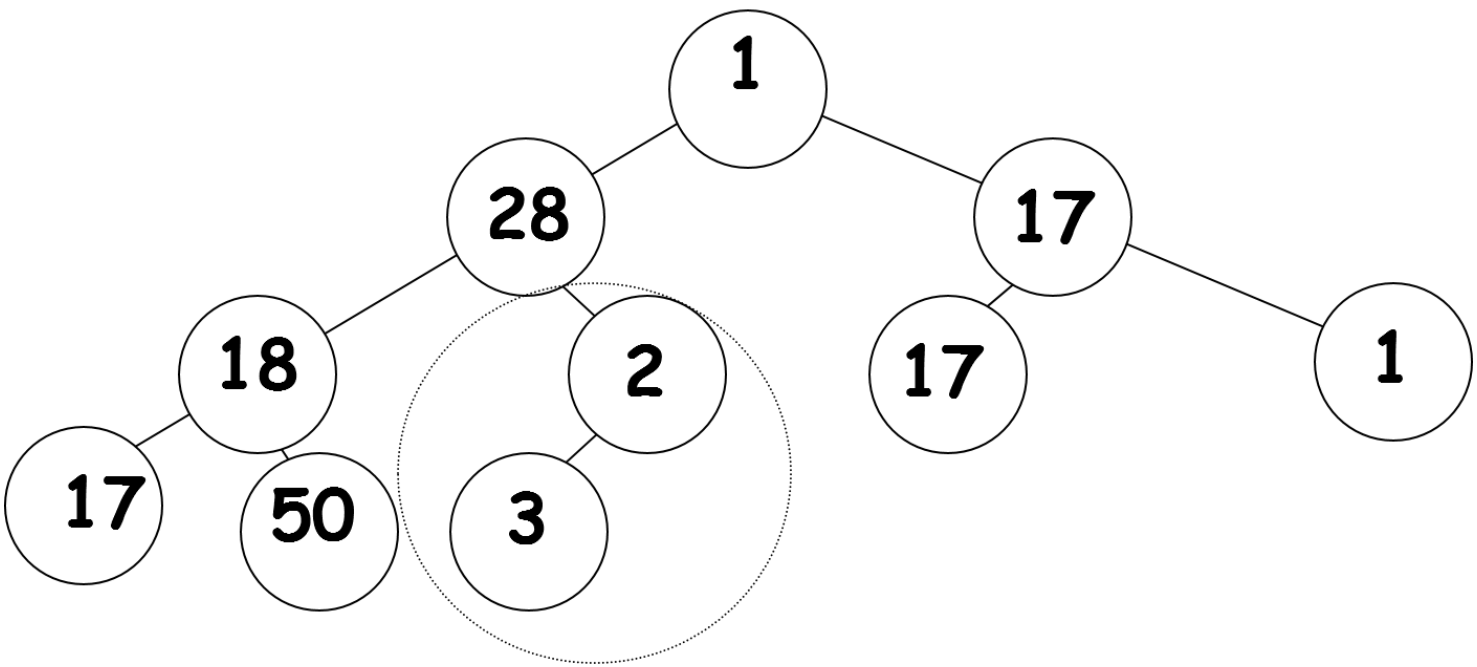
```
CriaHeap:
  Para i decrescendo de ⌊n/2⌋ até 1:
    DesceHeap(i, n);
  Fp;
Fim;
```

- Complexidade - $O(n)$

Listas de Prioridade - Heaps

- Criação de um Heap a partir de um vetor já preenchido, usando DESCEHEAP

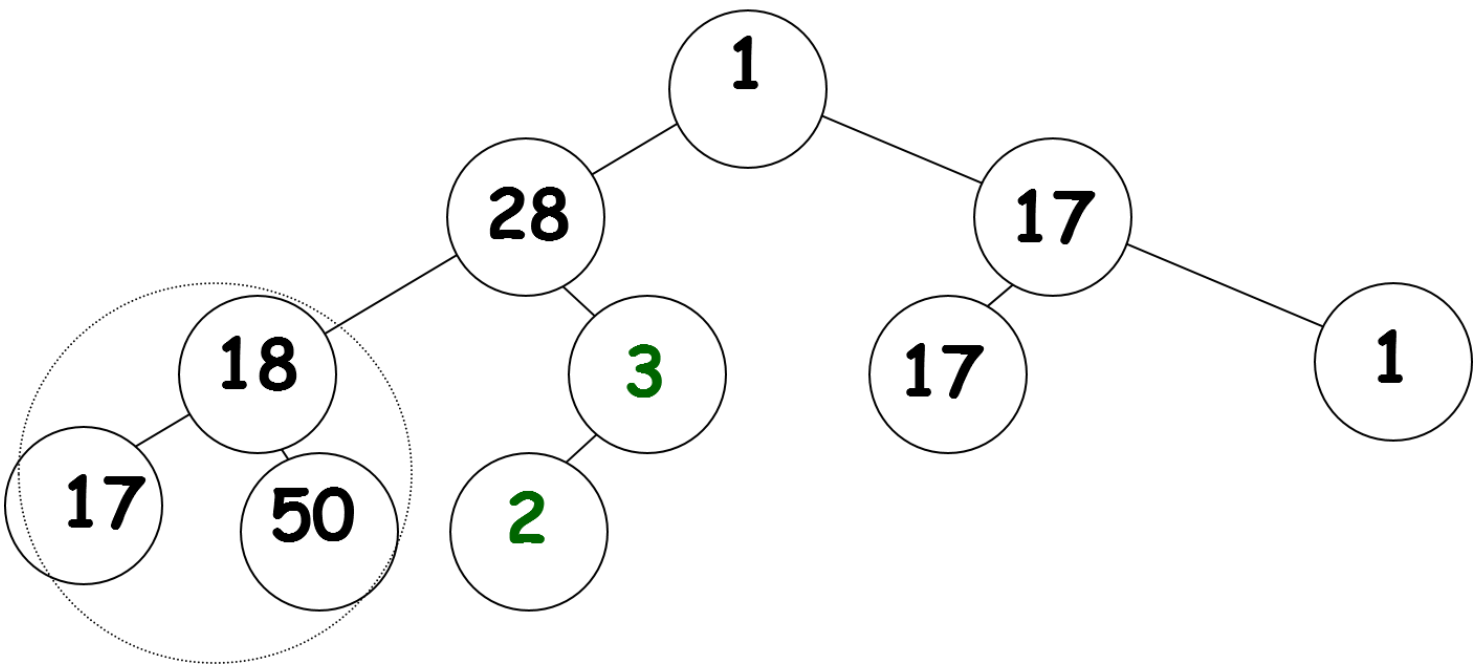
1	2	3	4	5	6	7	8	9	10
1	28	17	18	2	17	1	17	50	3



Listas de Prioridade - Heaps

- Criação de um Heap a partir de um vetor já preenchido, usando DESCEHEAP

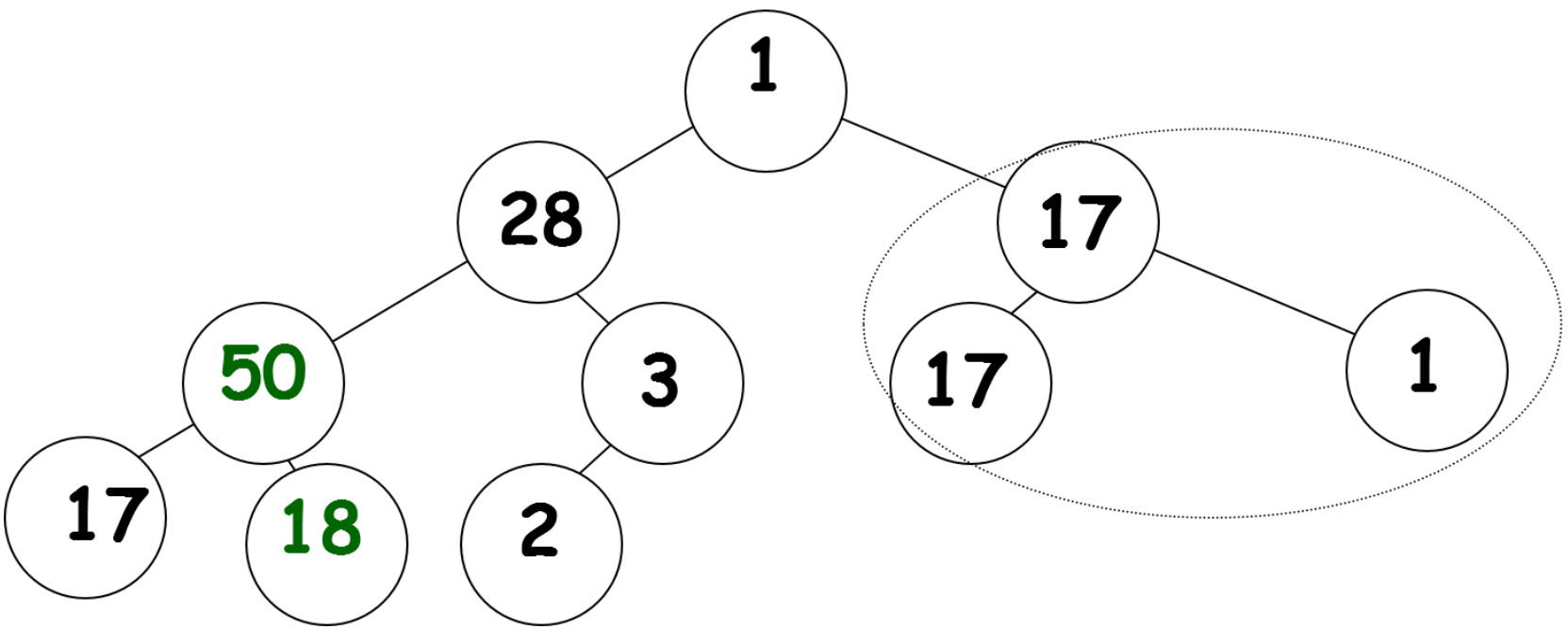
1	2	3	4	5	6	7	8	9	10
1	28	17	18	3	17	1	17	50	2



Listas de Prioridade - Heaps

- Criação de um Heap a partir de um vetor já preenchido, usando DESCEHEAP

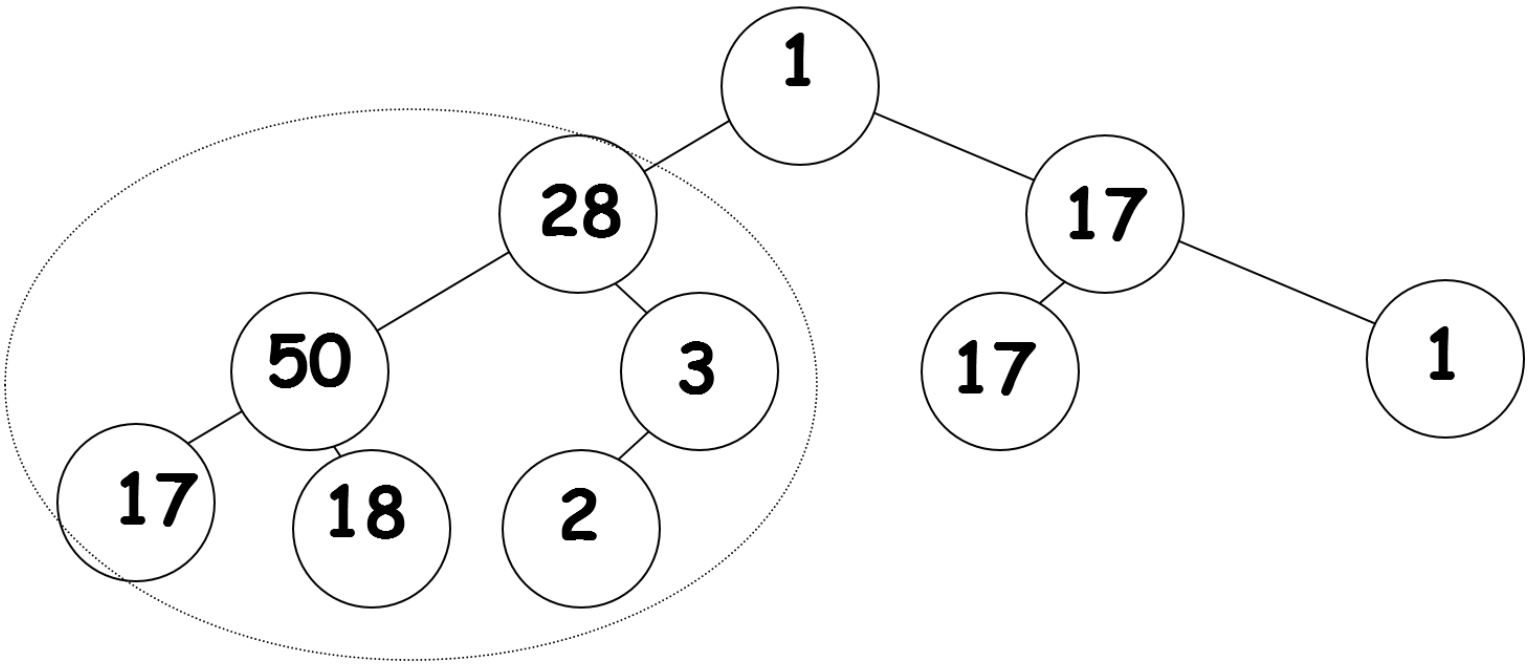
1	2	3	4	5	6	7	8	9	10
1	28	17	50	3	17	1	17	18	2



Listas de Prioridade - Heaps

- Criação de um Heap a partir de um vetor já preenchido, usando DESCEHEAP

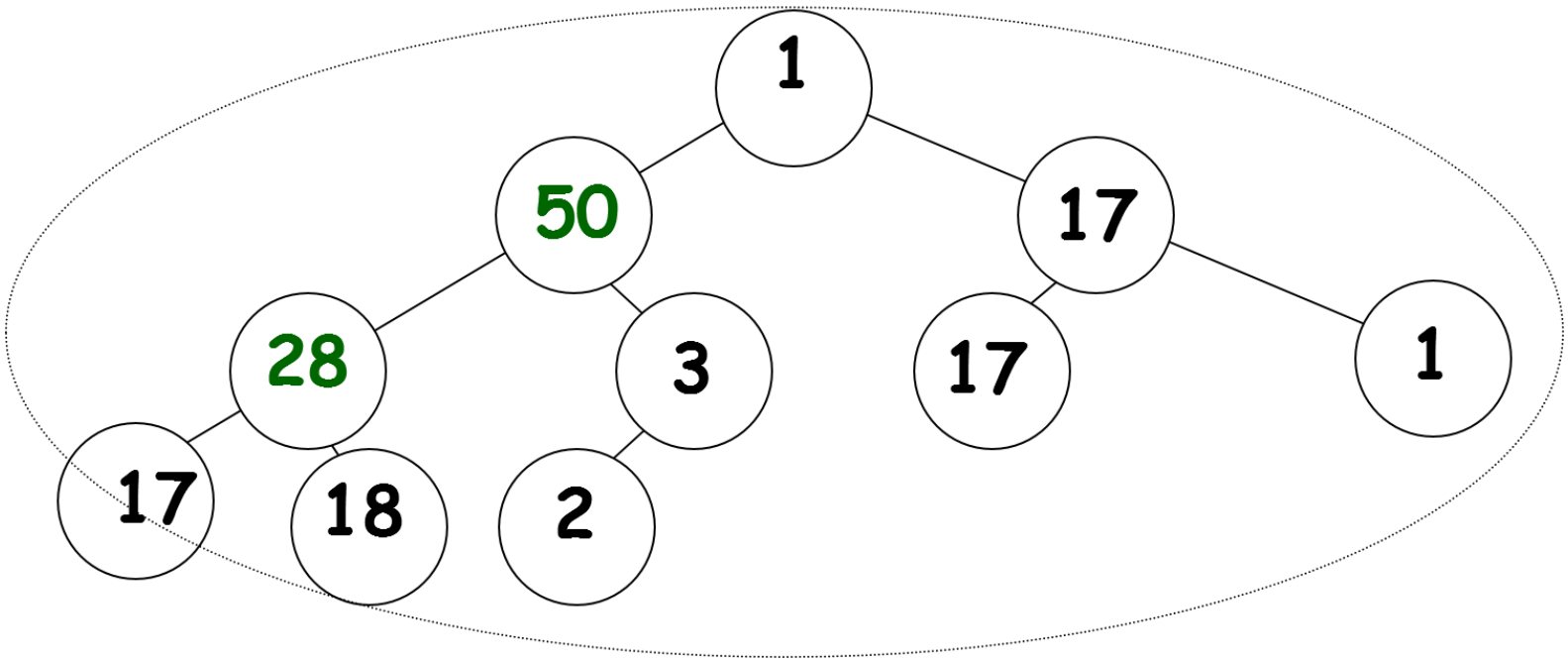
1	2	3	4	5	6	7	8	9	10
1	28	17	50	3	17	1	17	18	2



Listas de Prioridade - Heaps

- Criação de um Heap a partir de um vetor já preenchido, usando DESCEHEAP

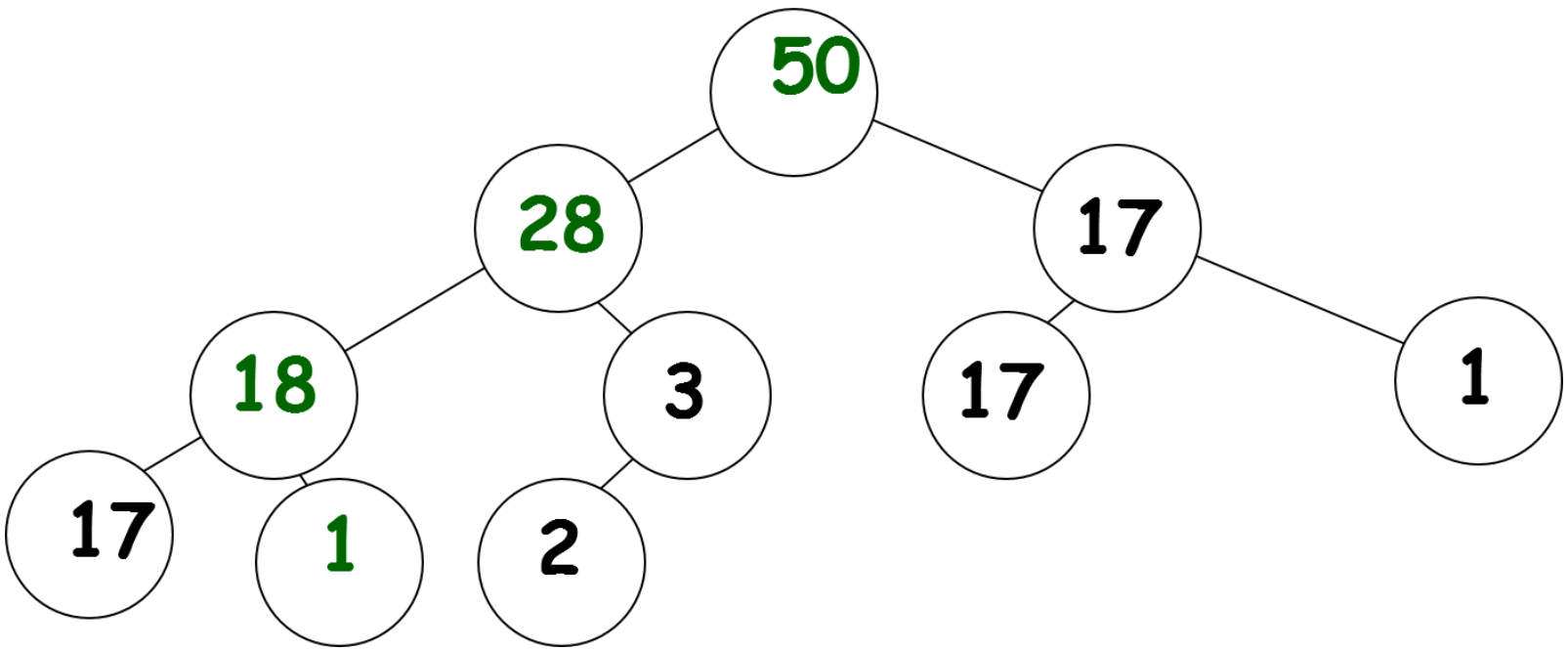
1	2	3	4	5	6	7	8	9	10
1	50	17	28	3	17	1	17	18	2



Listas de Prioridade - Heaps

- Criação de um Heap a partir de um vetor já preenchido, usando DESCEHEAP

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



Listas de Prioridade - Heaps

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

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

Listas de Prioridade - Heaps

HEAPSORT

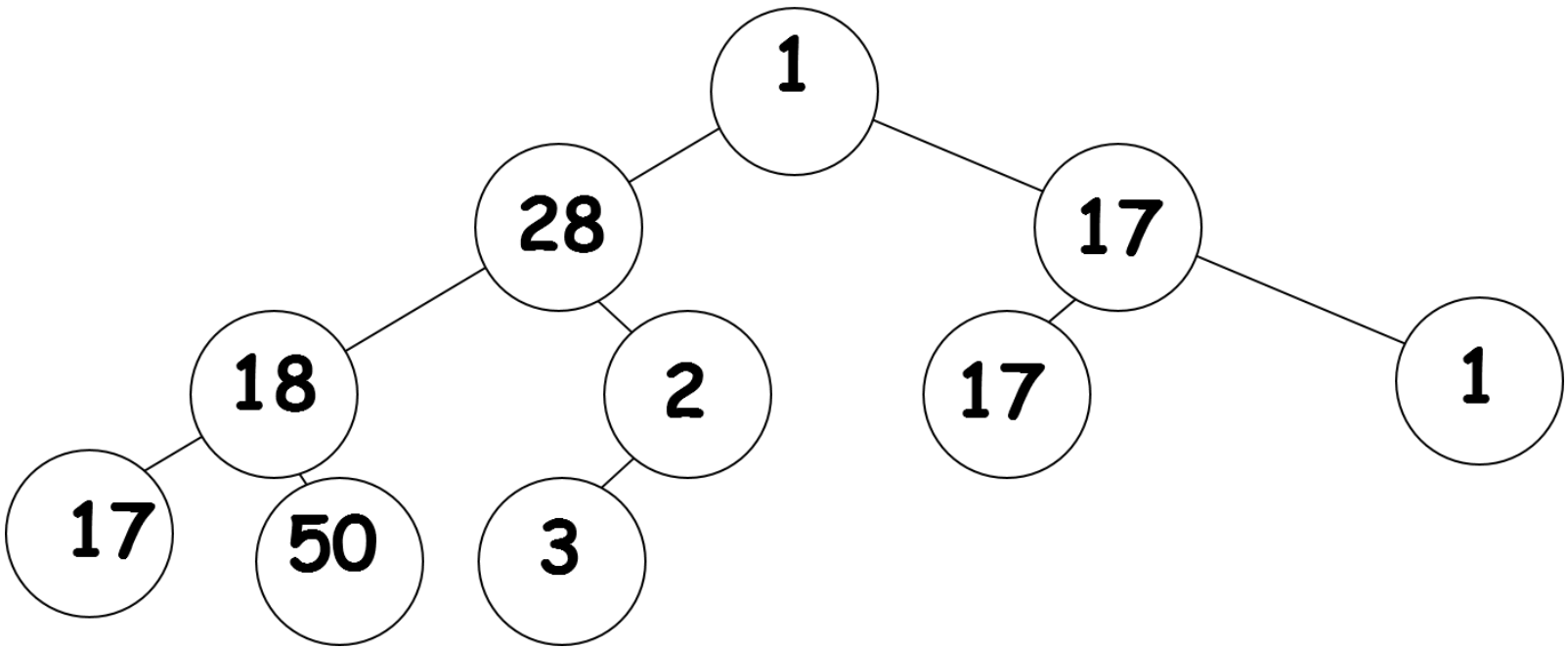
A idéia é criar um Heap e, sucessivamente, trocar o primeiro elemento com o último, diminuir o Heap e acertá-lo

```
Heapsort:
  CriaHeap();
  Para i decrescendo de n a 2:
    Troca(1, i);
    DesceHeap(1, i-1);
  Fp;
Fim;
```

Listas de Prioridade - Heaps

- HEAPSORT

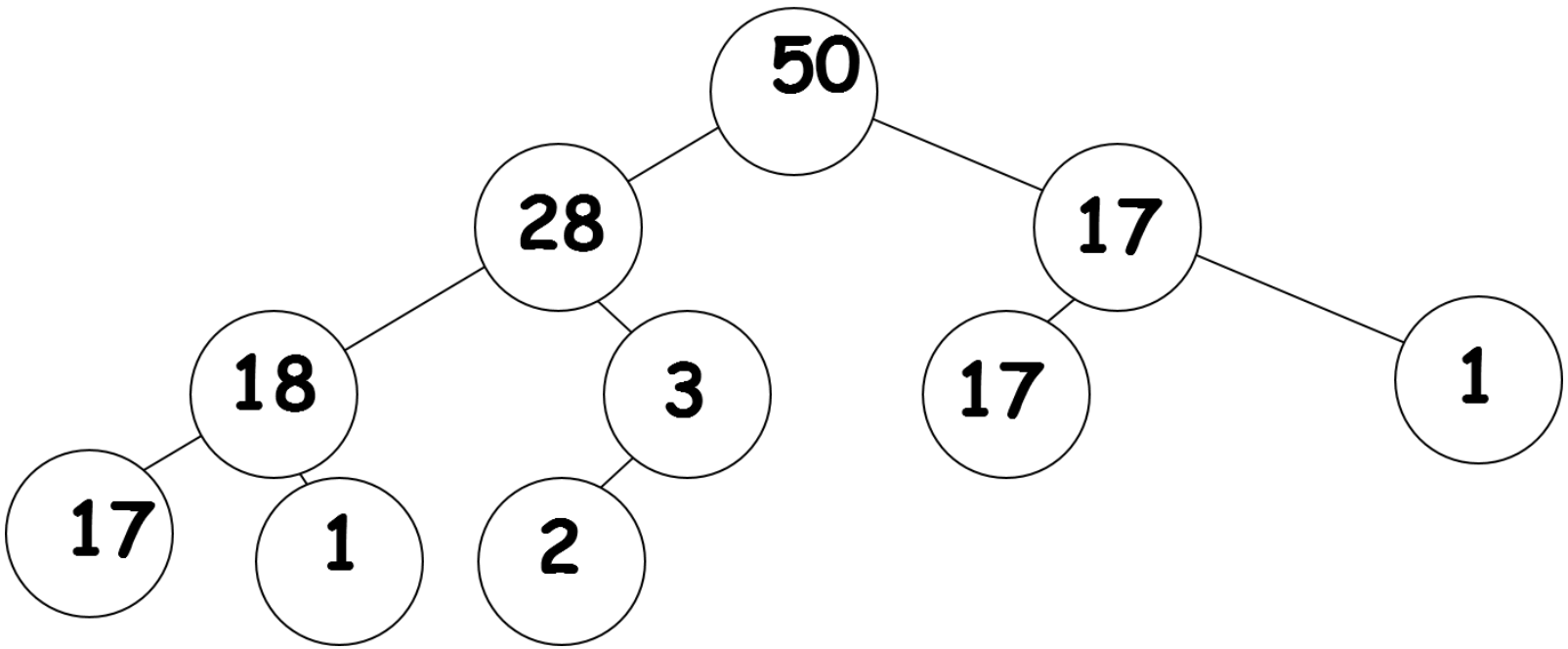
1	2	3	4	5	6	7	8	9	10
1	28	17	18	2	17	1	17	50	3



Listas de Prioridade - Heaps

- HEAPSORT. Passo 1 - Criação do Heap

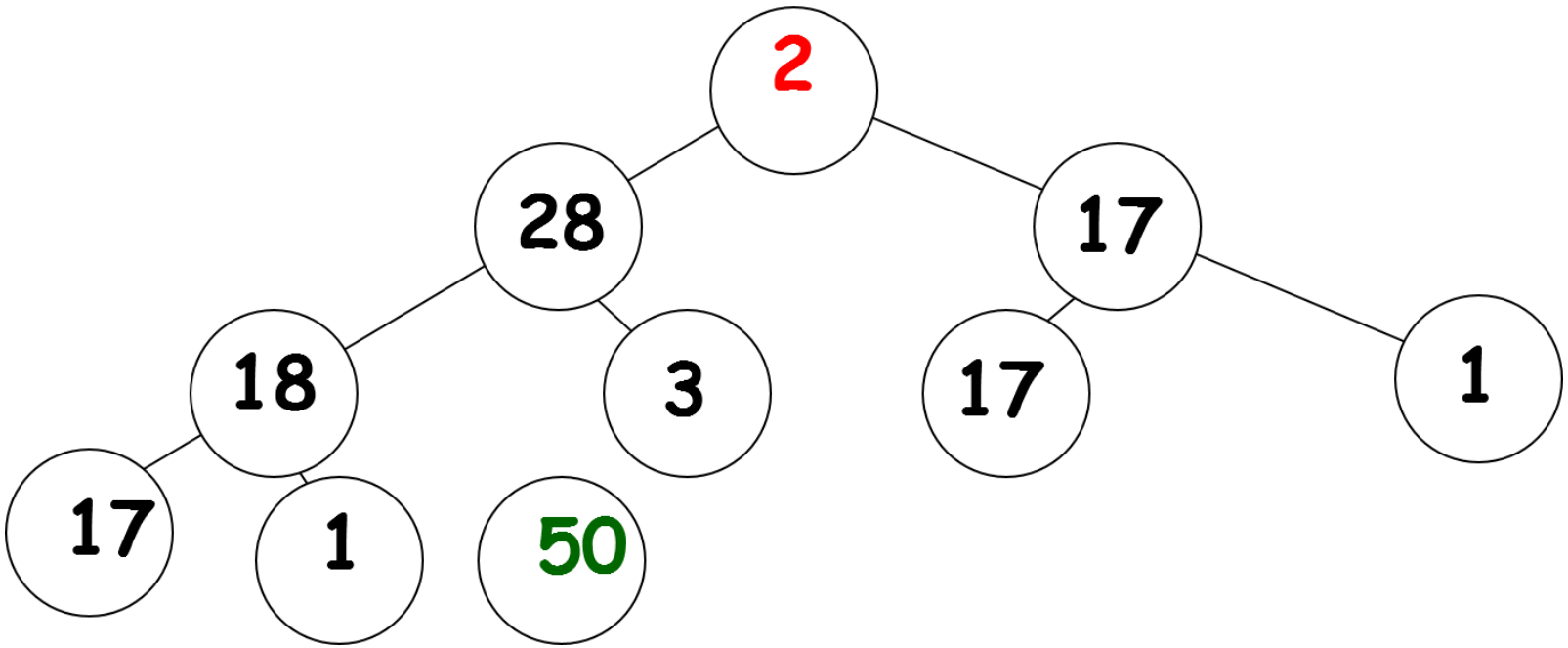
1	2	3	4	5	6	7	8	9	10
50	28	17	18	3	17	1	17	1	2



Listas de Prioridade - Heaps

- HEAPSORT. Passo 2 - i = 10 - Troca

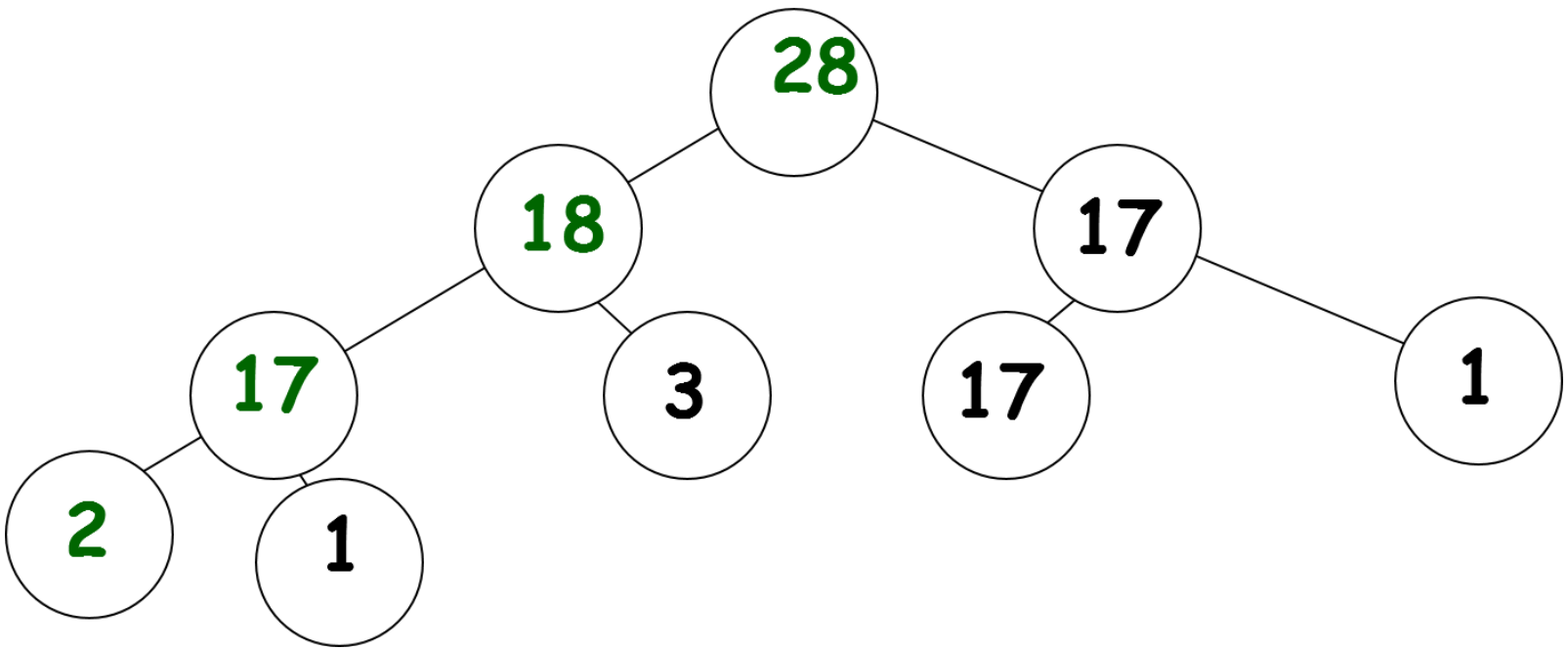
1	2	3	4	5	6	7	8	9	10
2	28	17	18	3	17	1	17	1	50



Listas de Prioridade - Heaps

- HEAPSORT. Passo 2 - i = 10 - DesceHeap (1,9)

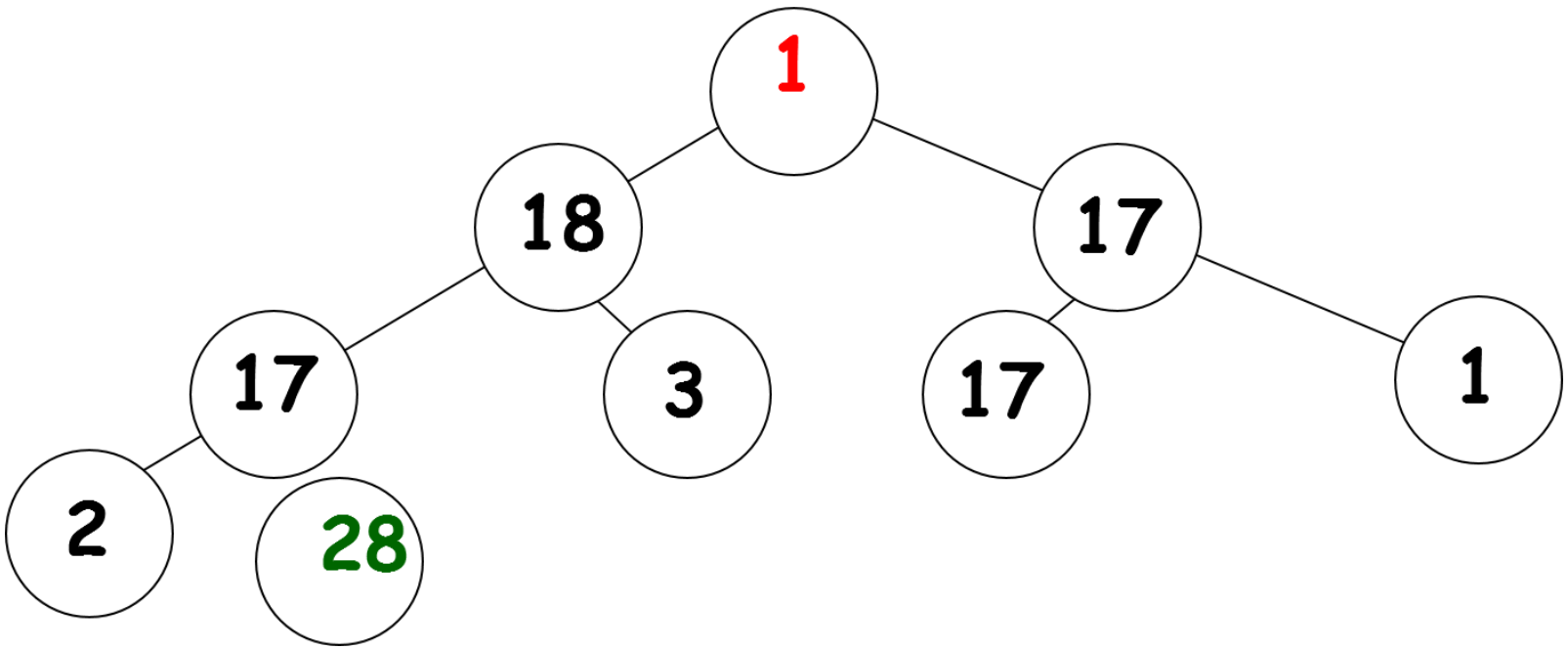
1	2	3	4	5	6	7	8	9	10
28	18	17	17	3	17	1	2	1	50



Listas de Prioridade - Heaps

- HEAPSORT. Passo 2 - i = 9 - Troca

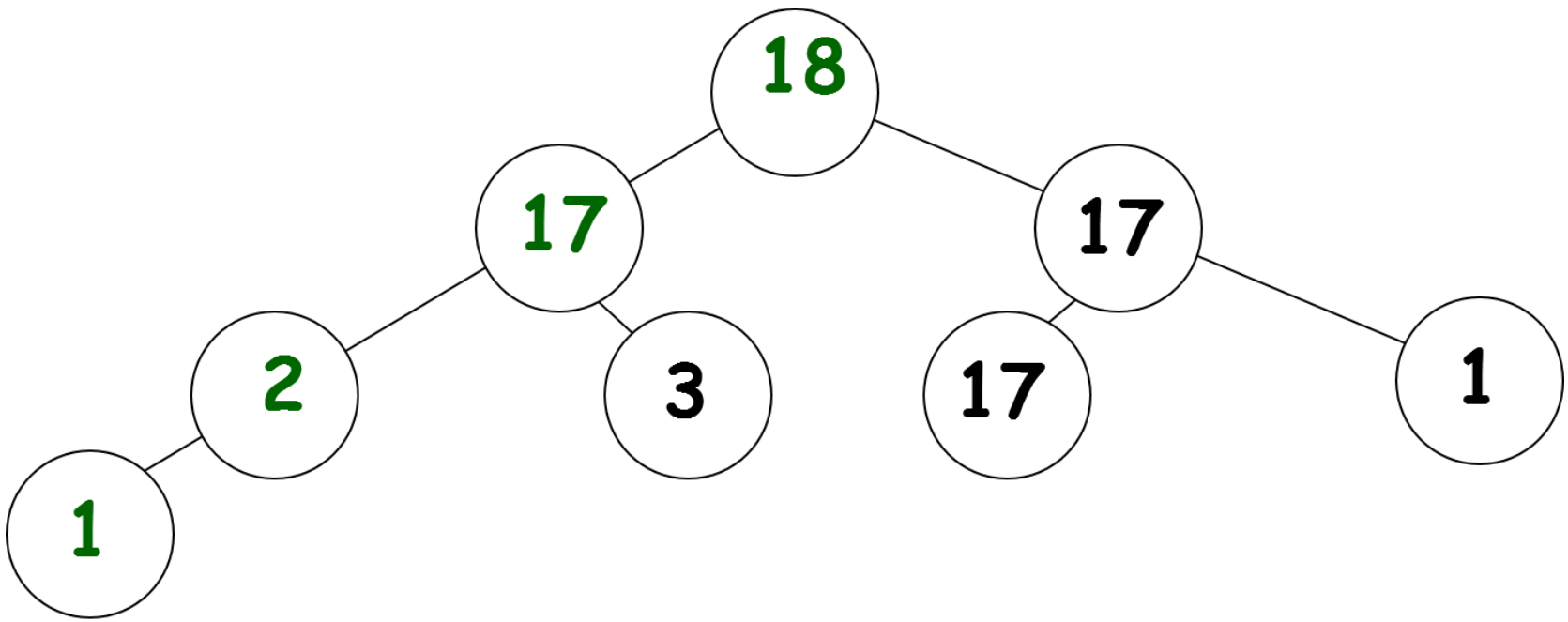
1	2	3	4	5	6	7	8	9	10
1	18	17	17	3	17	1	2	28	50



Listas de Prioridade - Heaps

- HEAPSORT. Passo 2 - i = 9 - DesceHeap (1, 8)

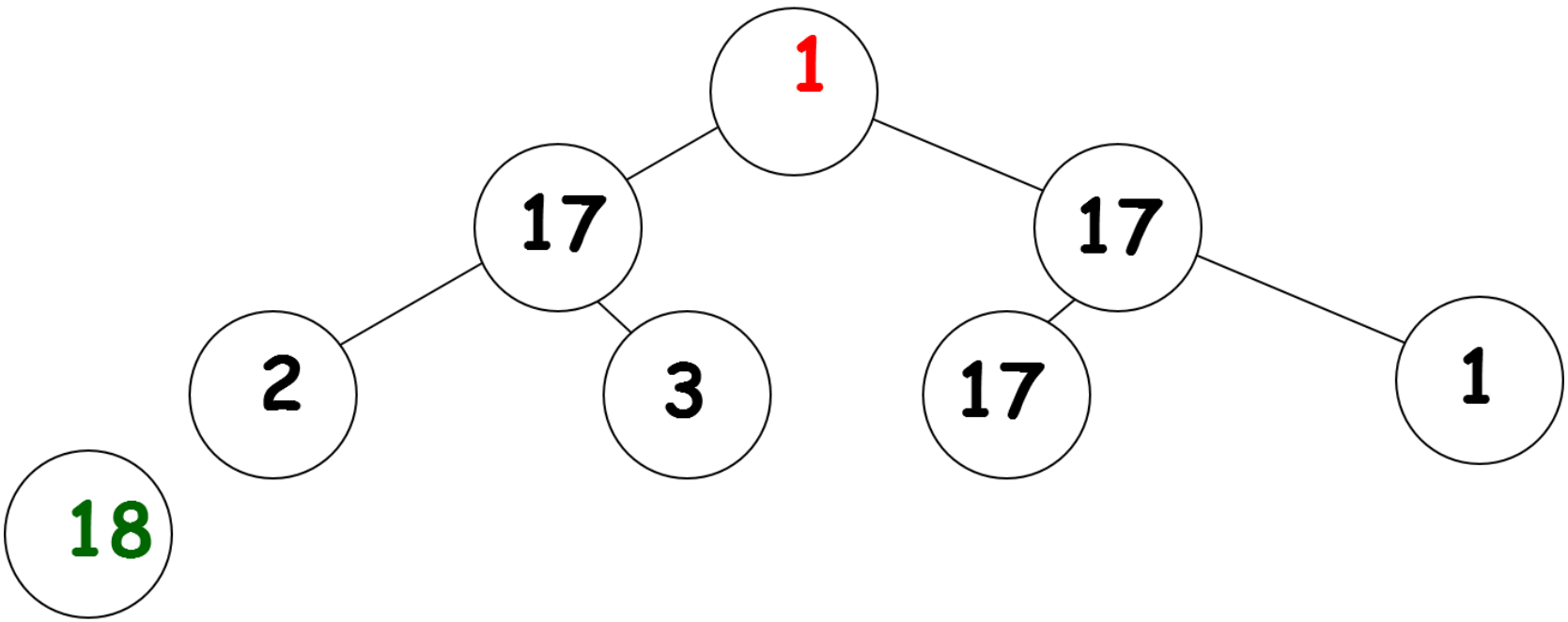
1	2	3	4	5	6	7	8	9	10
18	17	17	2	3	17	1	1	28	50



Listas de Prioridade - Heaps

- HEAPSORT. Passo 2 - i = 8 - Troca

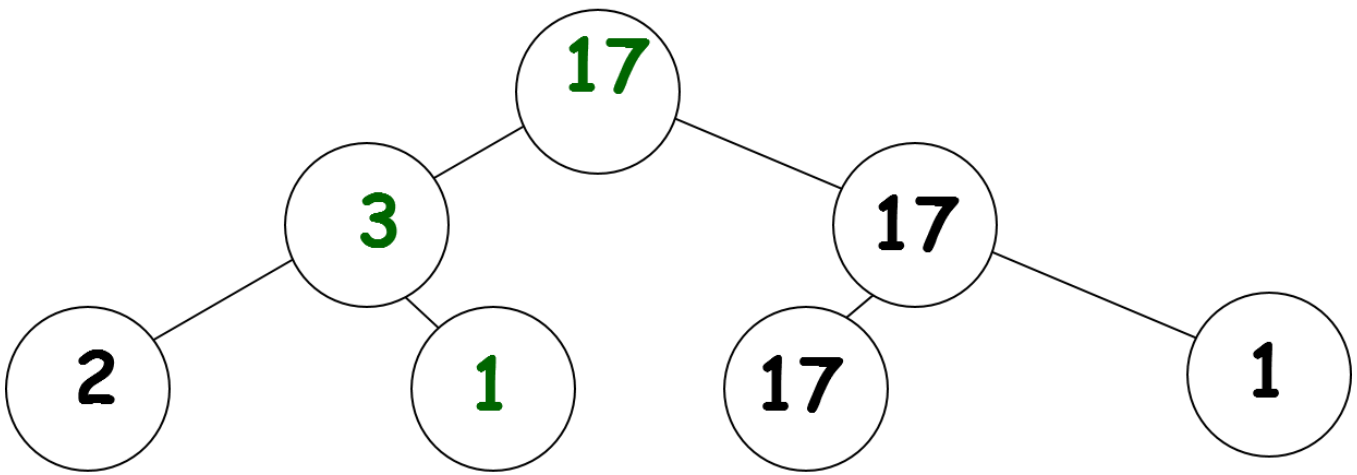
1	2	3	4	5	6	7	8	9	10
1	17	17	2	3	17	1	18	28	50



Listas de Prioridade - Heaps

- HEAPSORT. Passo 2 - i = 8 - DesceHeap (1, 7)

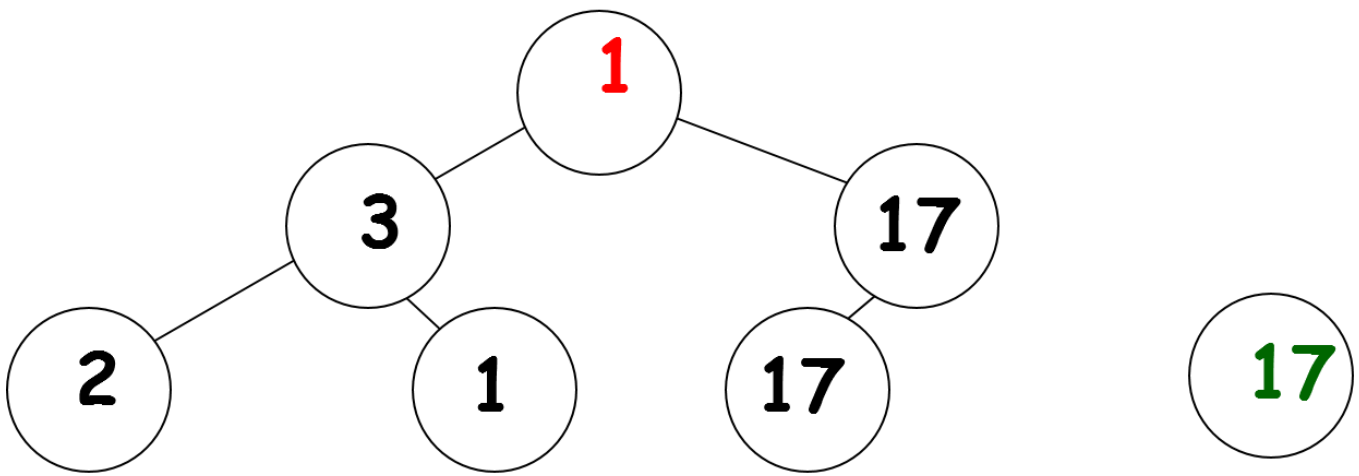
1	2	3	4	5	6	7	8	9	10
17	3	17	2	1	17	1	18	28	50



Listas de Prioridade - Heaps

- HEAPSORT. Passo 2 - i = 7 - Troca

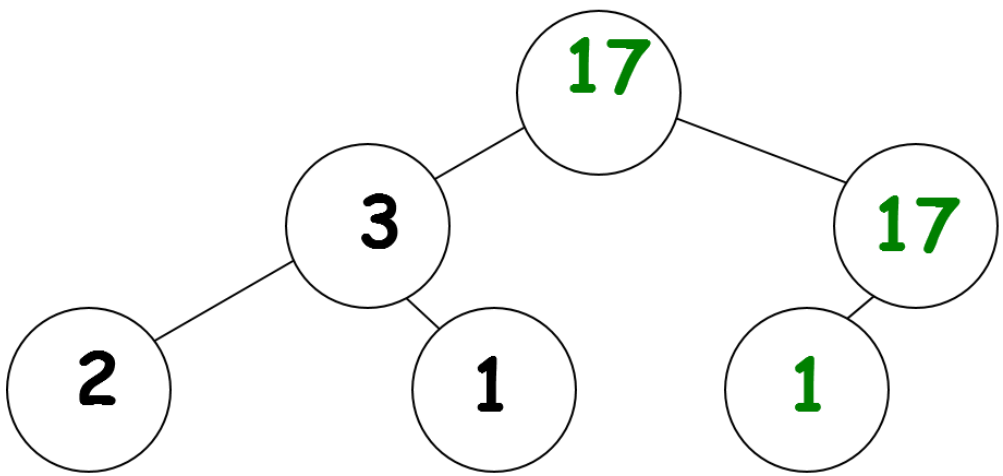
1	2	3	4	5	6	7	8	9	10
1	3	17	2	1	17	17	18	28	50



Listas de Prioridade - Heaps

- HEAPSORT. Passo 2 - i = 7 - DesceHeap (1, 6)

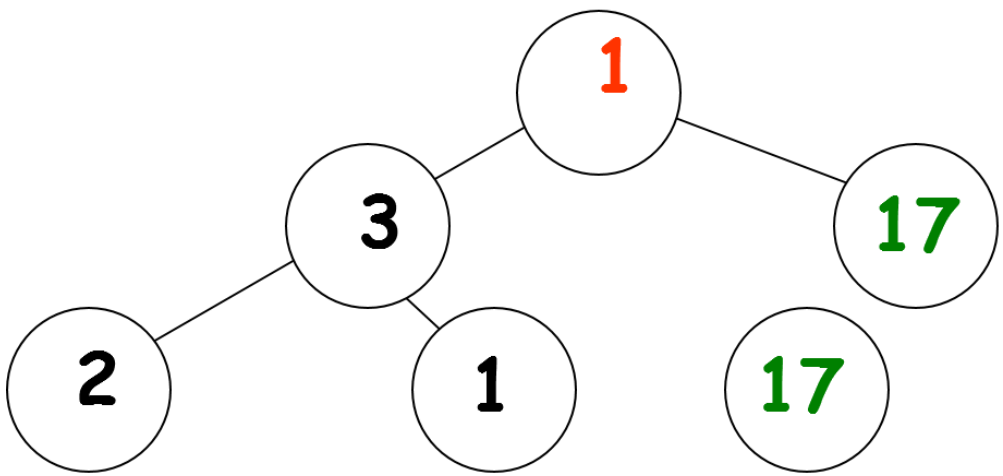
1	2	3	4	5	6	7	8	9	10
17	3	17	2	1	1	17	18	28	50



Listas de Prioridade - Heaps

- HEAPSORT. Passo 2 - i = 6 - Troca

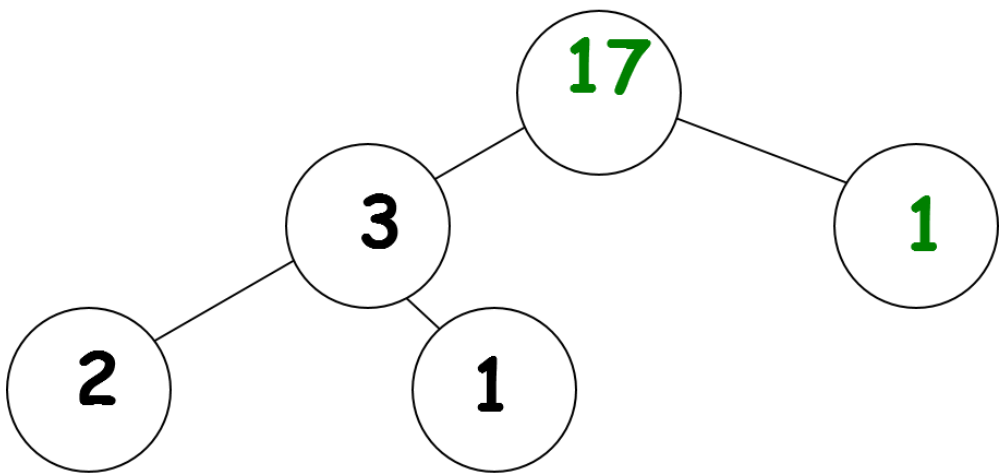
1	2	3	4	5	6	7	8	9	10
1	3	17	2	1	17	17	18	28	50



Listas de Prioridade - Heaps

- HEAPSORT. Passo 2 - i = 6 - DesceHeap (1, 5)

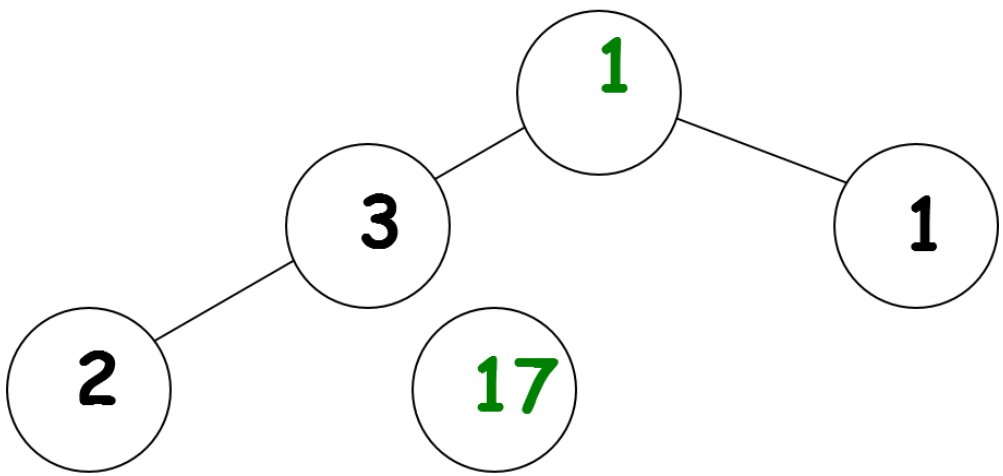
1	2	3	4	5	6	7	8	9	10
17	3	1	2	1	17	17	18	28	50



Listas de Prioridade - Heaps

- HEAPSORT. Passo 2 - i = 5 - Troca

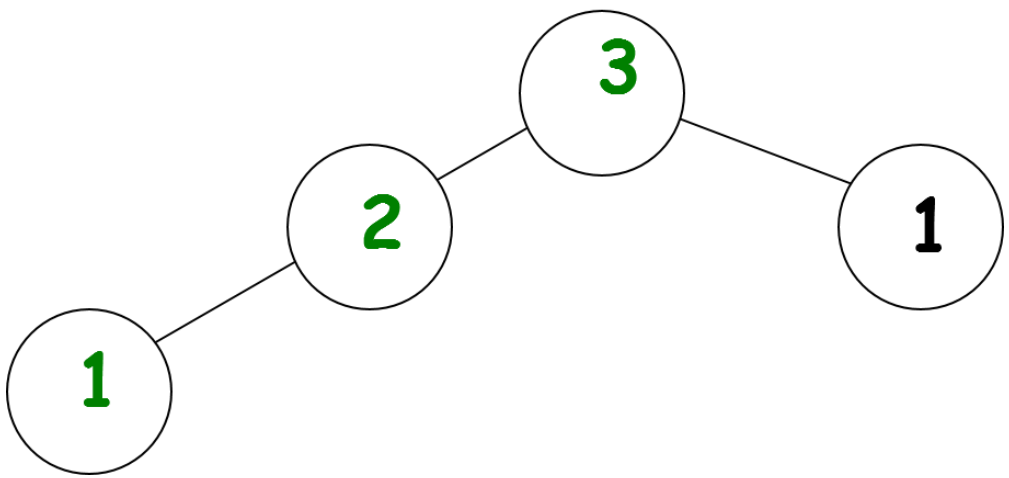
1	2	3	4	5	6	7	8	9	10
1	3	1	2	17	17	17	18	28	50



Listas de Prioridade - Heaps

- HEAPSORT. Passo 2 - i = 5 - DesceHeap (1, 4)

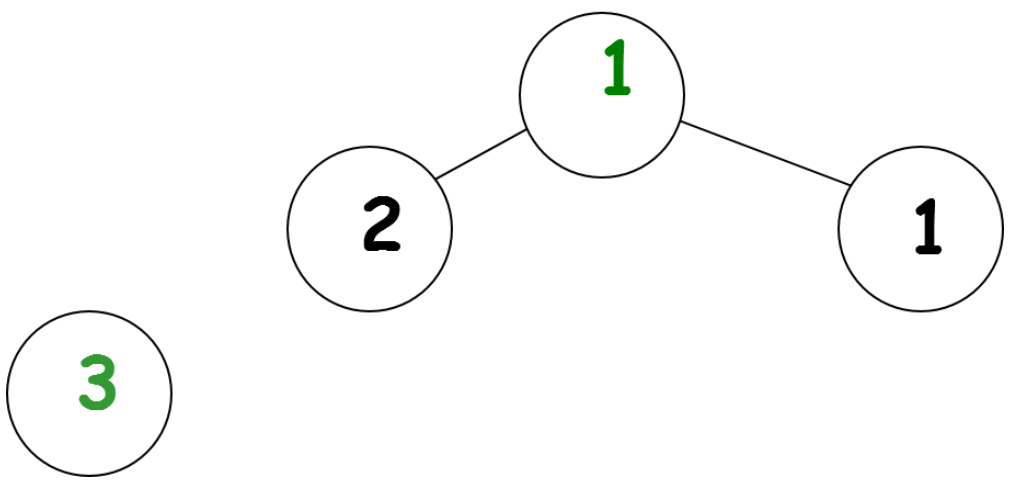
1	2	3	4	5	6	7	8	9	10
3	2	1	1	17	17	17	18	28	50



Listas de Prioridade - Heaps

- HEAPSORT. Passo 2 - i = 4 - Troca

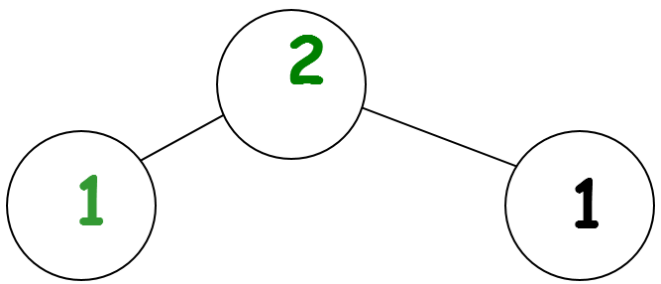
1	2	3	4	5	6	7	8	9	10
1	2	1	3	17	17	17	18	28	50



Listas de Prioridade - Heaps

- HEAPSORT. Passo 2 - i = 4 - DesceHeap (1, 3)

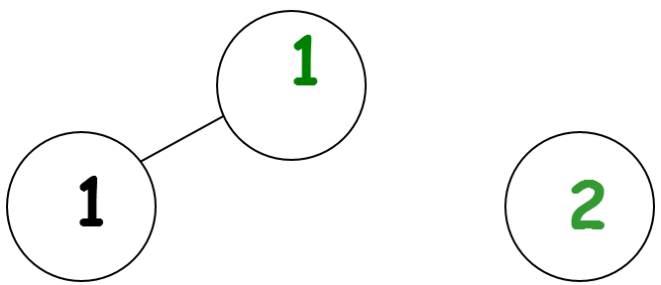
1	2	3	4	5	6	7	8	9	10
2	1	1	3	17	17	17	18	28	50



Listas de Prioridade - Heaps

- HEAPSORT. Passo 2 - i = 3 - Troca

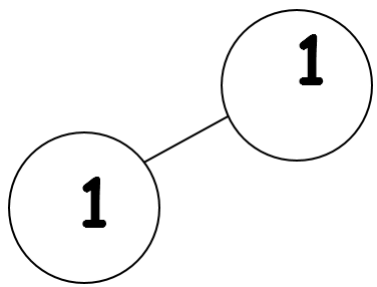
1	2	3	4	5	6	7	8	9	10
1	1	2	3	17	17	17	18	28	50



Listas de Prioridade - Heaps

- HEAPSORT. Passo 2 - i = 3 - DesceHeap (1, 2)

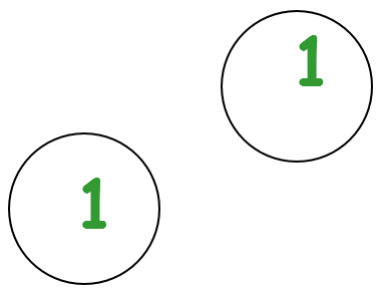
1	2	3	4	5	6	7	8	9	10
1	1	2	3	17	17	17	18	28	50



Listas de Prioridade - Heaps

- HEAPSORT. Passo 2 - i = 2 - Troca

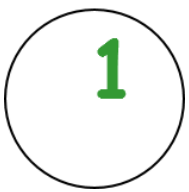
1	2	3	4	5	6	7	8	9	10
1	1	2	3	17	17	17	18	28	50



Listas de Prioridade - Heaps

- HEAPSORT. Passo 2 - i = 2 - DesceHeap (1, 1)

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



Análise do HEAPSORT

Complexidade

- Pior caso: $O(n\ log\ n)$ vetor ordenado decrescente
- Melhor caso: $O(n)$ chaves iguais

Memória adicional

Nenhuma

Usos especiais

Algoritmo de uso geral

Listas de Prioridade - Heaps

- Ordenar o MIXTRING (10 letras) usando Heapsort

Leitura complementar

Jayme, capítulo 6 (todo)
Cormen, capítulo 6 (todo)


That's all Folks!