

Leftist heap

AULA 6

TAOCP 5.2.3 Vol. 3

Além das operações usuais de uma fila com prioridades permite que a união ("merge") de duas filas seja feita facilmente.

Estrutura simples, ultrapassada por outras como binomial heaps (CLRS 19) e fibonacci heaps (CLRS 20).

Navigation icons

Navigation icons

Árvores esquerdistas

Cada nó x tem quatro campos:

1. $esq[x]$: filho esquerdo de x ;
2. $dir[x]$: filho direito de x ;
3. $dist[x]$: menor comprimento de um caminho de x

a NIL.

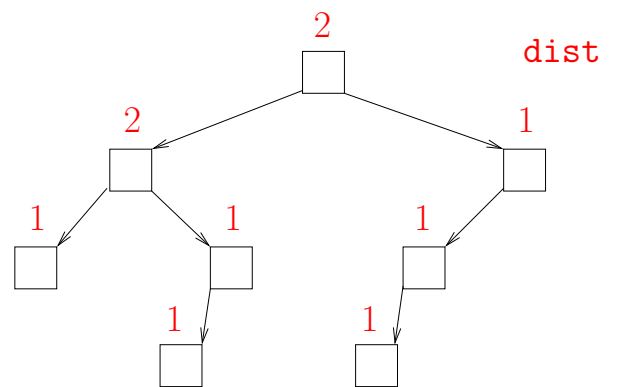
$dist(x)$

- 1 se $x = NIL$
- 2 então devolva 0
- 3 senão devolva $1 + \min\{dist(esq[x]), dist(dir[x])\}$

Navigation icons

Navigation icons

Exemplo



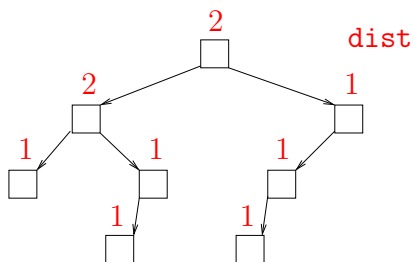
Árvores esquerdistas

Uma árvore é **esquerdista** se

$$dist[esq[x]] \geq dist[dir[x]]$$

para todo nó x ($dist[NIL] = 0$).

Exemplo 1:



Navigation icons

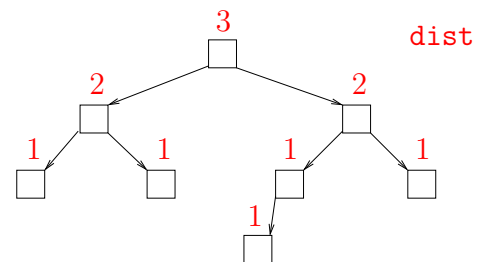
Árvores esquerdistas

Uma árvore é **esquerdista** se

$$dist[esq[x]] \geq dist[dir[x]]$$

para todo nó x ($dist[NIL] = 0$).

Exemplo 2:



Navigation icons

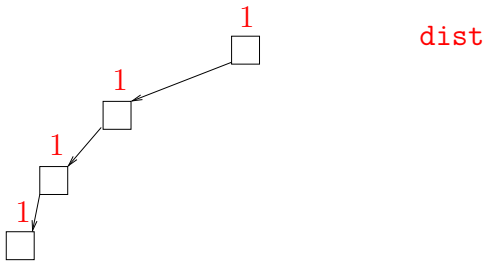
Árvores esquerdistas

Uma árvore é **esquerdista** se

$$\text{dist}[\text{esq}[x]] \geq \text{dist}[\text{dir}[x]]$$

para todo nó x ($\text{dist}[\text{NIL}] = 0$).

Exemplo 3:



Navigation icons

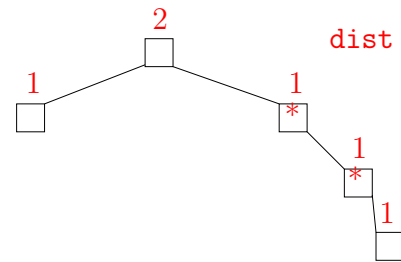
Árvores esquerdistas

Uma árvore é **esquerdista** se

$$\text{dist}[\text{esq}[x]] \geq \text{dist}[\text{dir}[x]]$$

para todo nó x ($\text{dist}[\text{NIL}] = 0$).

Exemplo 4: árvore não-esquerdista



Navigation icons

Caminho direitista

O **caminho direitista** de um nó x é a seqüência

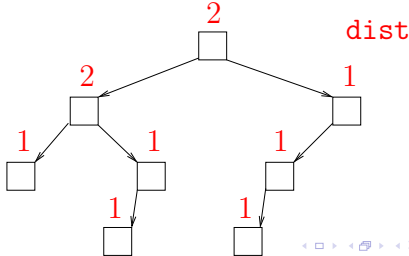
$$\langle x, \text{dir}[x], \text{dir}[\text{dir}[x]], \dots, \text{NIL} \rangle.$$

$\text{dcomp}[x]$:= núm. de nós no caminho direitista de x

$\text{tam}[x]$:= número de nós na árvore de raiz x

Se x é um nó de uma árvore esquerdista, então

$$\text{dist}[x] = \text{dcomp}[x].$$



Navigation icons

Fato estrutural

$\text{tam}[x]$:= número de nós na árvore de raiz x

Se x é um nó de uma árvore esquerdista, então

$$\text{tam}[x] \geq 2^{\text{dist}[x]} - 1.$$

Navigation icons

Fato estrutural

$\text{tam}[x]$:= número de nós na árvore de raiz x

Se x é um nó de uma árvore esquerdista, então

$$\text{tam}[x] \geq 2^{\text{dist}[x]} - 1.$$

Prova: Seja $d := \text{dist}[x]$.

Se $d = 1$, então $\text{tam}[x] \geq 1 = 2^d - 1$.

Suponha que $d \geq 2$ e que a desigualdade vale para $d - 1$.

Temos que $\text{dist}[\text{dir}[x]] = d - 1$ e que existe um nó y na árvore de raiz $\text{esq}[x]$ tal que $\text{dist}[y] = d - 1$.

Navigation icons

Fato estrutural

$\text{tam}[x]$:= número de nós na árvore de raiz x

Fato 2. Se x é um nó de uma árvore esquerdista, então

$$\text{tam}[x] \geq 2^{\text{dist}[x]} - 1.$$

Prova: (continuação)

Logo,

$$\begin{aligned} \text{tam}[x] &= \text{tam}[\text{esq}[x]] + \text{tam}[\text{dir}[x]] + 1 \\ &\geq \text{tam}[y] + \text{tam}[\text{dir}[x]] + 1 \\ &\stackrel{\text{hi}}{\geq} 2^{d-1} - 1 + 2^{d-1} - 1 + 1 \\ &= 2^d - 1 \end{aligned}$$

Navigation icons

Consequência

Se x é um nó de uma árvore esquerdista, então

$$\text{dist}[x] = \text{dcomp}[x] \leq \lfloor \lg(\text{tam}[x]+1) \rfloor = O(\lg \text{tam}[x]).$$

Em particular:

Se x é raiz de uma árvore esquerdista com m nós,

$$\text{dist}[x] = \text{dcomp}[x] \leq \lfloor \lg(m+1) \rfloor = O(\lg m).$$

Prova: $m \geq 2^d - 1 \Rightarrow m+1 \geq 2^d \Rightarrow \lfloor \lg(m+1) \rfloor \geq d$.

Heap esquerdista

$H :=$ árvore

$\text{raiz}[H] :=$ raiz de H

$\text{prior}[x] :=$ prioridade do nó x

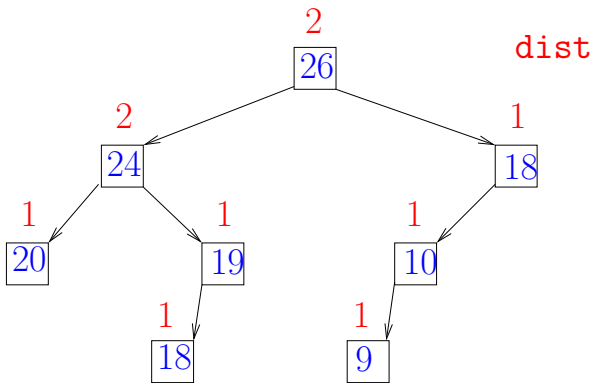
$\text{pai}[x] :=$ pai do nó x

Um **heap esquerdista** H é uma árvore esquerdista que satisfaz

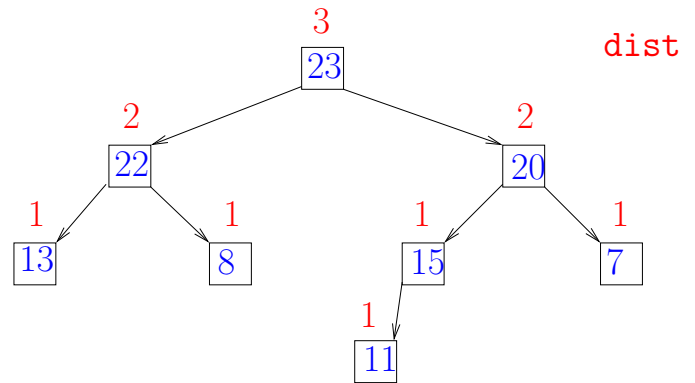
$$\text{prior}[\text{pai}[x]] \geq \text{prior}[x]$$

para todo nó $x \neq \text{raiz}[H]$.

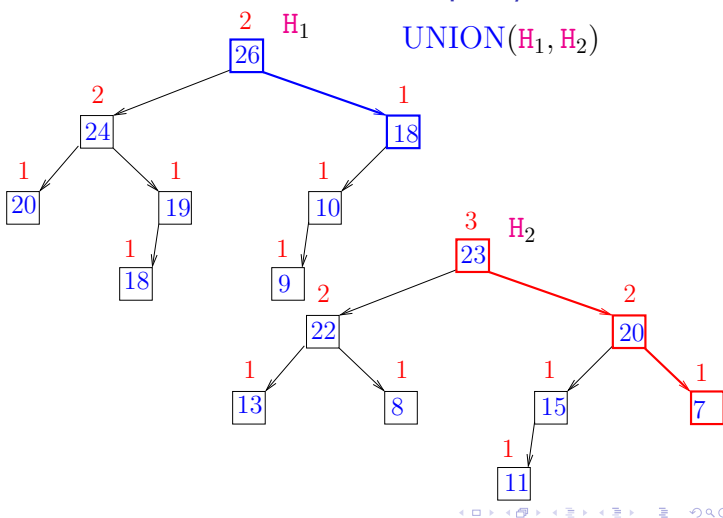
Heap esquerdista



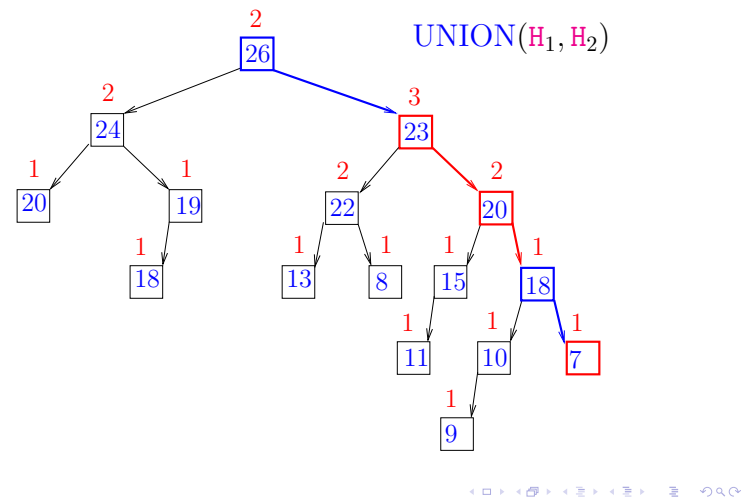
Heap esquerdista



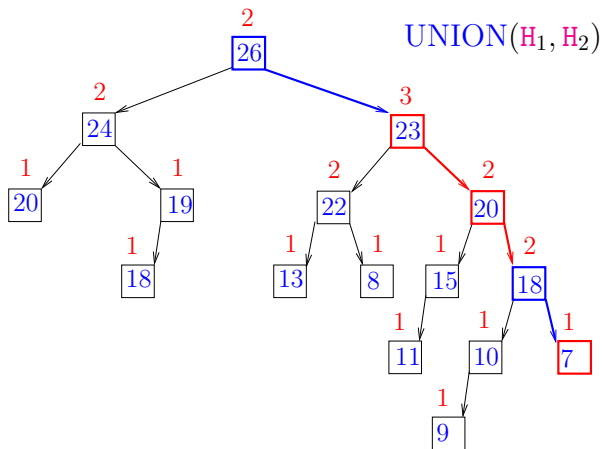
Rotina básica de manipulação



Rotina básica de manipulação

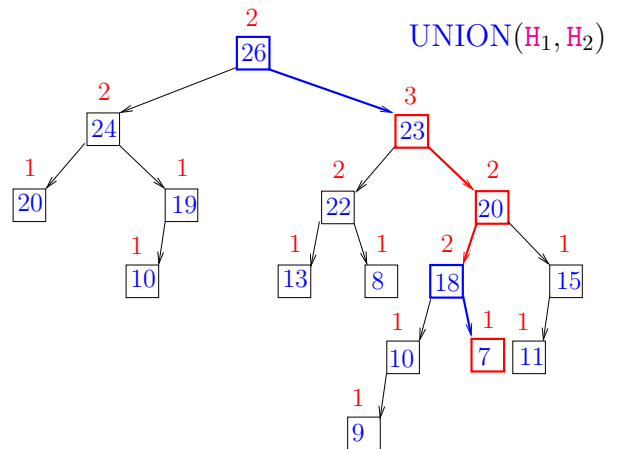


Rotina básica de manipulação



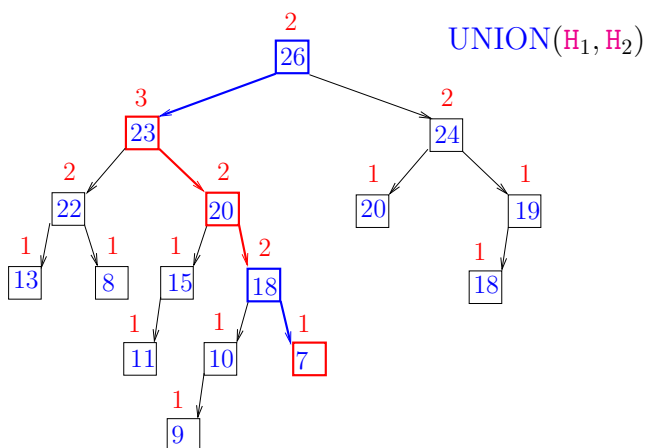
Navigation icons

Rotina básica de manipulação



Navigation icons

Rotina básica de manipulação



Navigation icons

Rotina básica de manipulação

```

LEFTIST-HEAP-UNION (H1, H2)
1  se raiz[H1] = NIL então devolva H2
2  se raiz[H2] = NIL então devolva H1
3  se prior[raiz[H1]] < prior[raiz[H2]]
4     então H1 ↔ H2
5  x1 ← raiz[H1]  x2 ← raiz[H2]
6  se esq[x1] = NIL então esq[x1] ← x2
7  senão H' ← MakeLeftistHeap()
8     raiz[H'] ← dir[x1]
9     H' ← LeftistHeapMerge(H', H2)
10  dir[x1] ← raiz[H']
11  se dist[esq[x1]] < dist[dir[x1]]
12     então esq[x1] ↔ dir[x1]
13  dist[x1] ← dist[dir[x1]] + 1
14  devolva H1
    
```

Navigation icons

Consumo de tempo

O consumo de tempo do algoritmo LeftistHeapMerge no pior caso é proporcional a

$$dcomp(raiz[H_1]) + dcomp(raiz[H_2]) = O(\lg m)$$

onde $m = tam(raiz[H_1]) + tam(raiz[H_2])$

O consumo de tempo do algoritmo LeftistHeapMerge é $O(\lg m)$.

Navigation icons

Fila com heap esquerdista

Rotina que insere um nó x em um heap esquerdista H .

A rotina supõe que $prior[x]$ já foi definido.

```

LEFTIST-HEAP-INSERT (H, x)
1  H' ← MakeLeftistHeap()
2  esq[x] ← NIL
3  dir[x] ← NIL
4  dist[x] ← 1
5  raiz[H'] ← x
6  H ← LeftistHeapMerge(H, H')
    
```

Consome tempo $O(\lg m)$.

Navigation icons

Fila com heap esquerdista

Rotina que remove e devolve o nó de maior prioridade de um heap esquerdista H .

LEFTIST-HEAP-EXTRACT (H)

```
1  $x \leftarrow \text{raiz}[H]$ 
2  $H' \leftarrow \text{MakeLeftistHeap}()$ 
3  $\text{raiz}[H'] \leftarrow \text{esq}[x]$ 
4  $\text{raiz}[H] \leftarrow \text{dir}[x]$ 
5  $H \leftarrow \text{LeftistHeapMerge}(H, H')$ 
6 devolva  $x$ 
```

Consome tempo $O(\lg m)$.