AULA 6

Leftist heap

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).

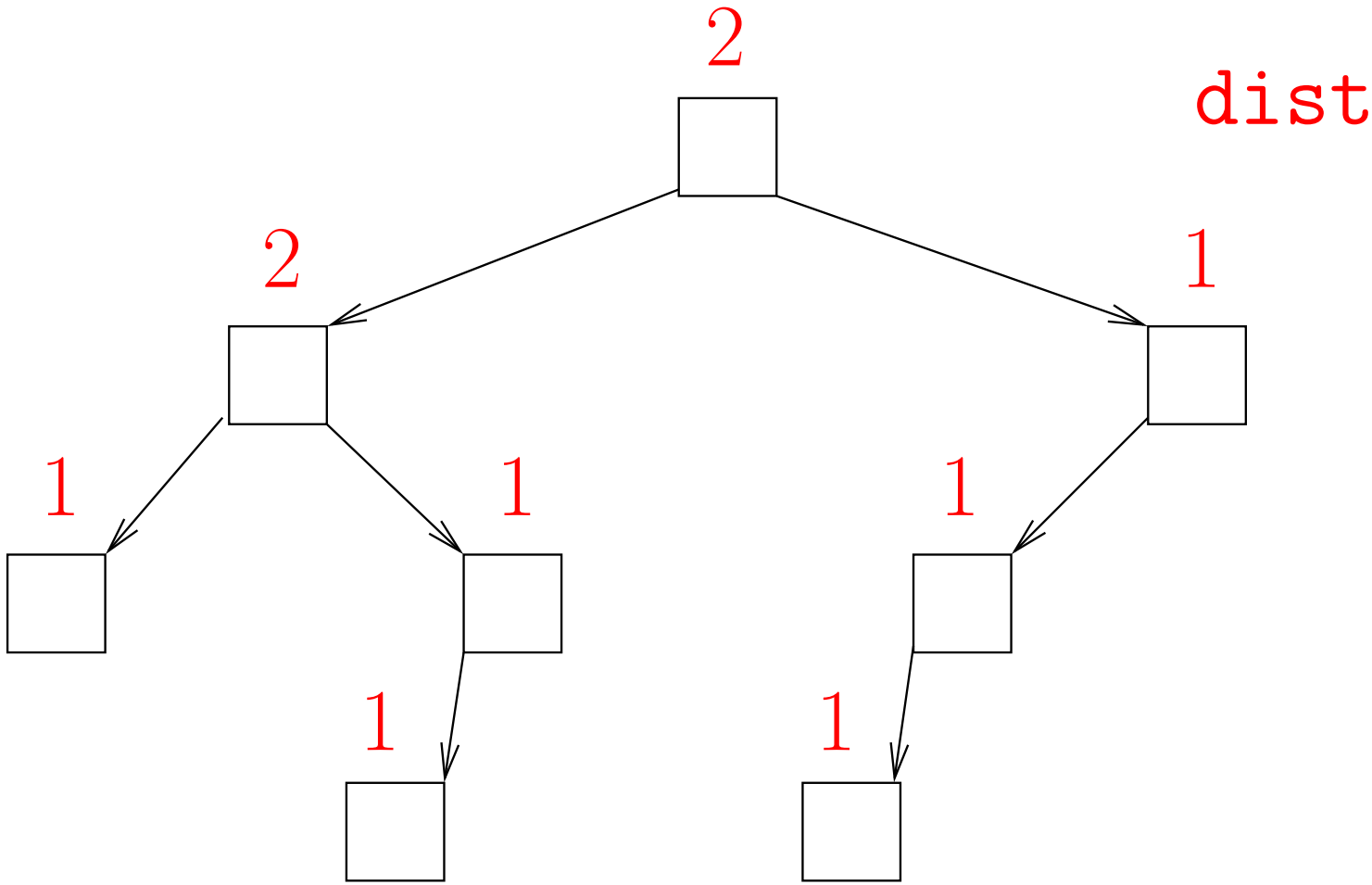
Árvores esquerdistas

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])}
```

Exemplo



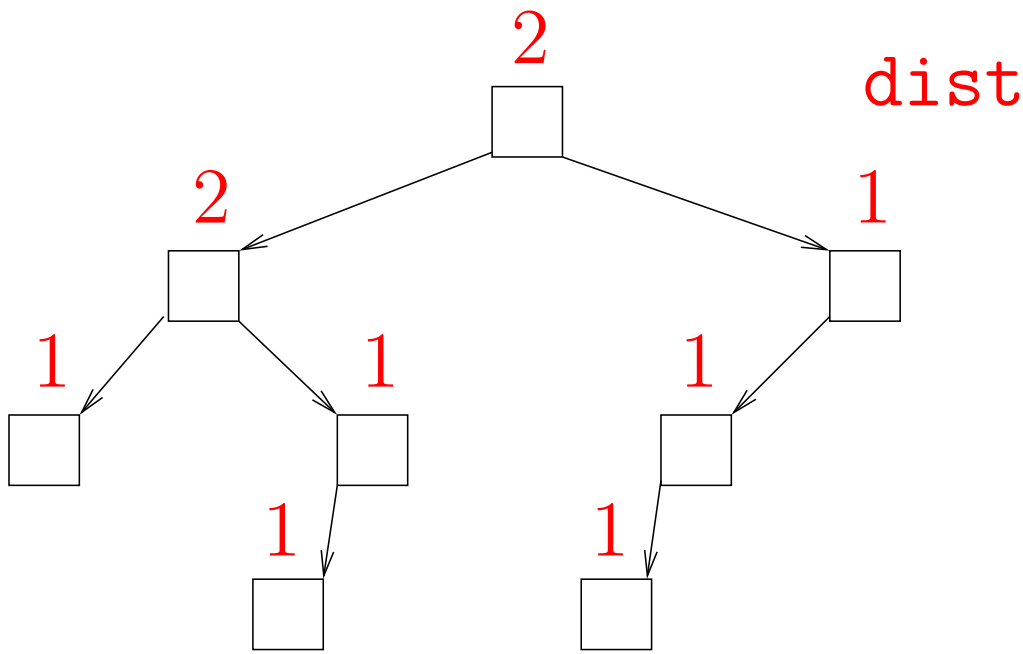
Árvores esquerdistas

Uma árvore é esquerdista se

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

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

Exemplo 1:



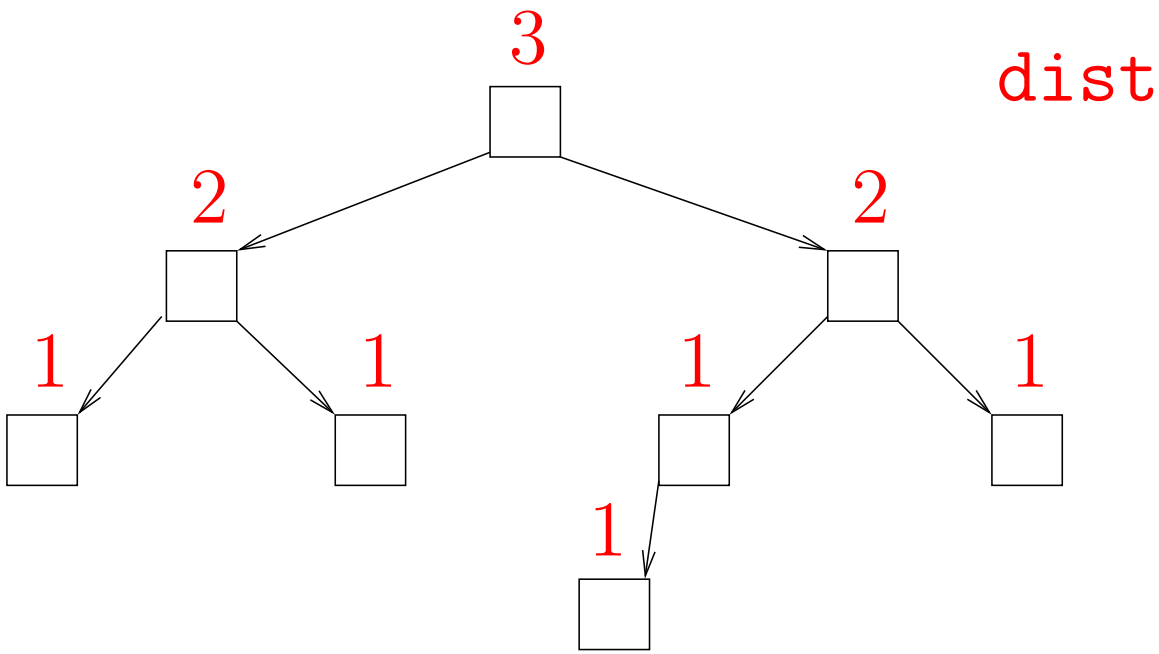
Árvores esquerdistas

Uma árvore é esquerdista se

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

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

Exemplo 2:



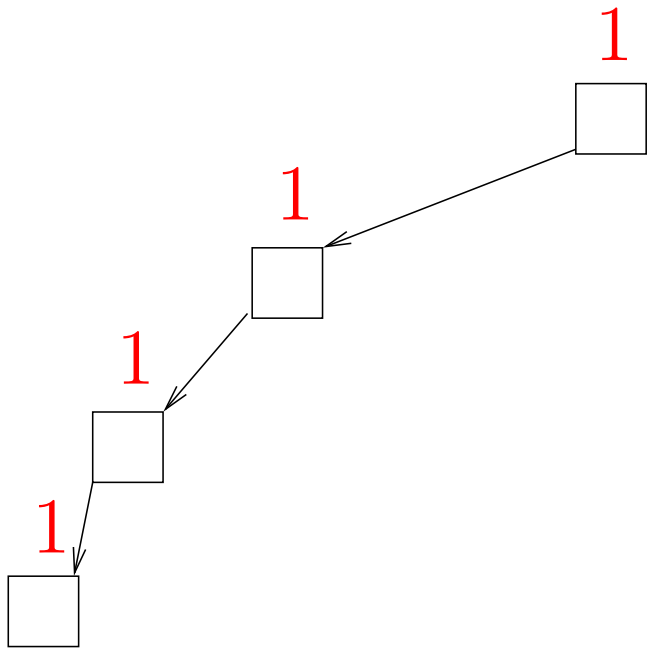
Árvores esquerdistas

Uma árvore é esquerdista se

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

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

Exemplo 3:



dist

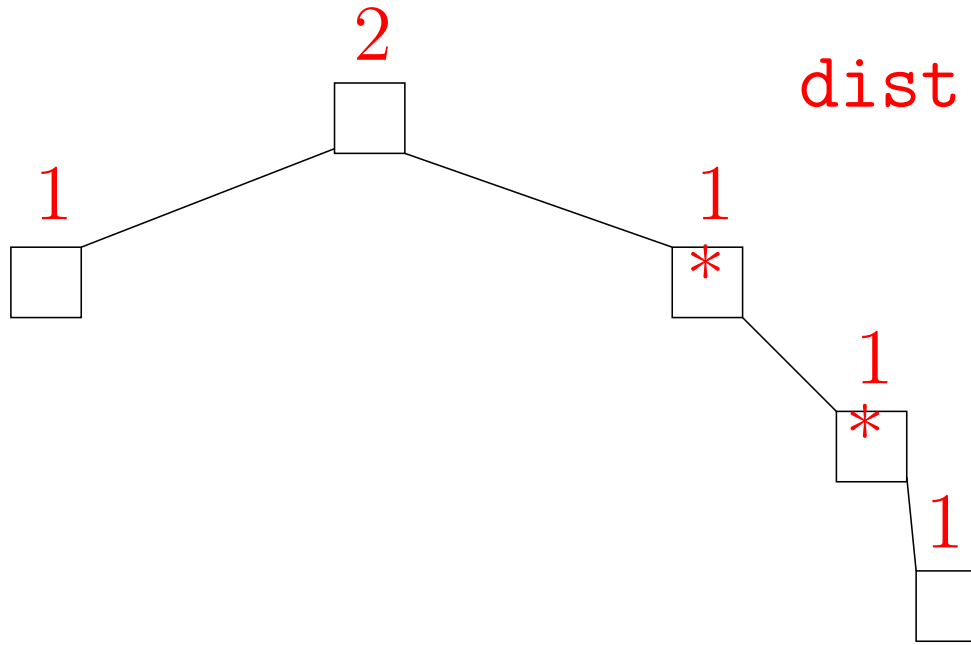
Árvores esquerdistas

Uma árvore é esquerdista se

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

para todo nó x ($\texttt{dist}[\textsc{nil}] = 0$).

Exemplo 4: árvore não-esquerdista



Caminho direitista

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

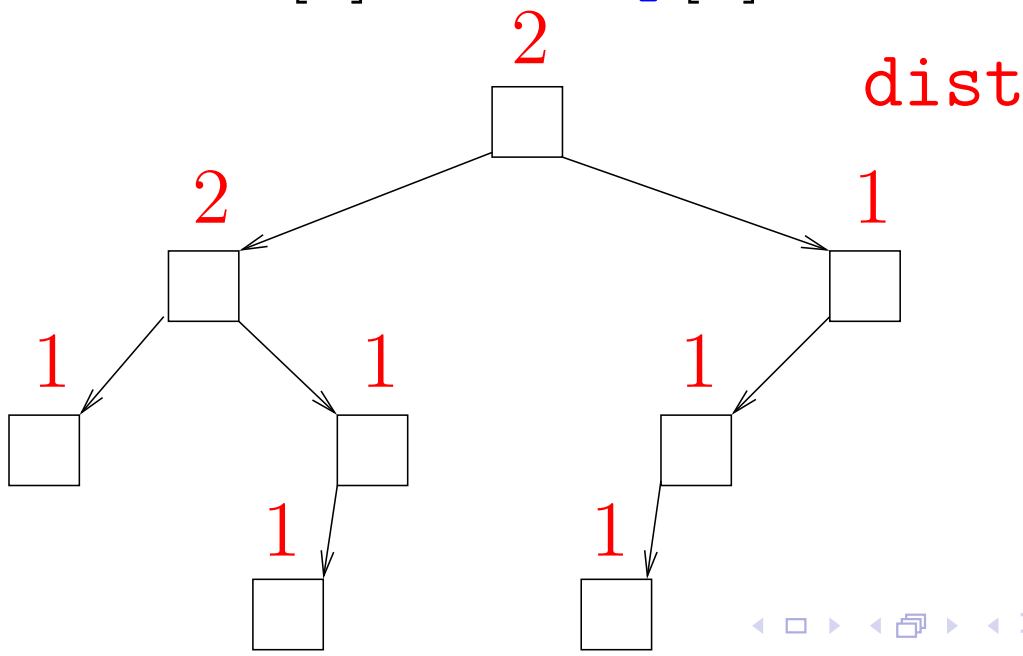
$$\langle x, \mathtt{dir}[x], \mathtt{dir}[\mathtt{dir}[x]], \ldots, \text{NIL} \rangle.$$

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

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

Se x é um nó de uma árvore esquerdista, então

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



Fato estrutural

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

Se $\texttt{x}$ é um nó de uma árvore esquerdista, então

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

Fato estrutural

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

Se $\texttt{x}$ é um nó de uma árvore esquerdista, então

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

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

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

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

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

Fato estrutural

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

Fato 2. Se $\texttt{x}$ é um nó de uma árvore esquerdista, então

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

Prova: (continuação)
Logo,

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

Consequência

Se x é um nó de uma árvore esquerdista, então

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

Em particular:

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

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

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

Heap esquerdista

$\texttt{H} :=$ árvore

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

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

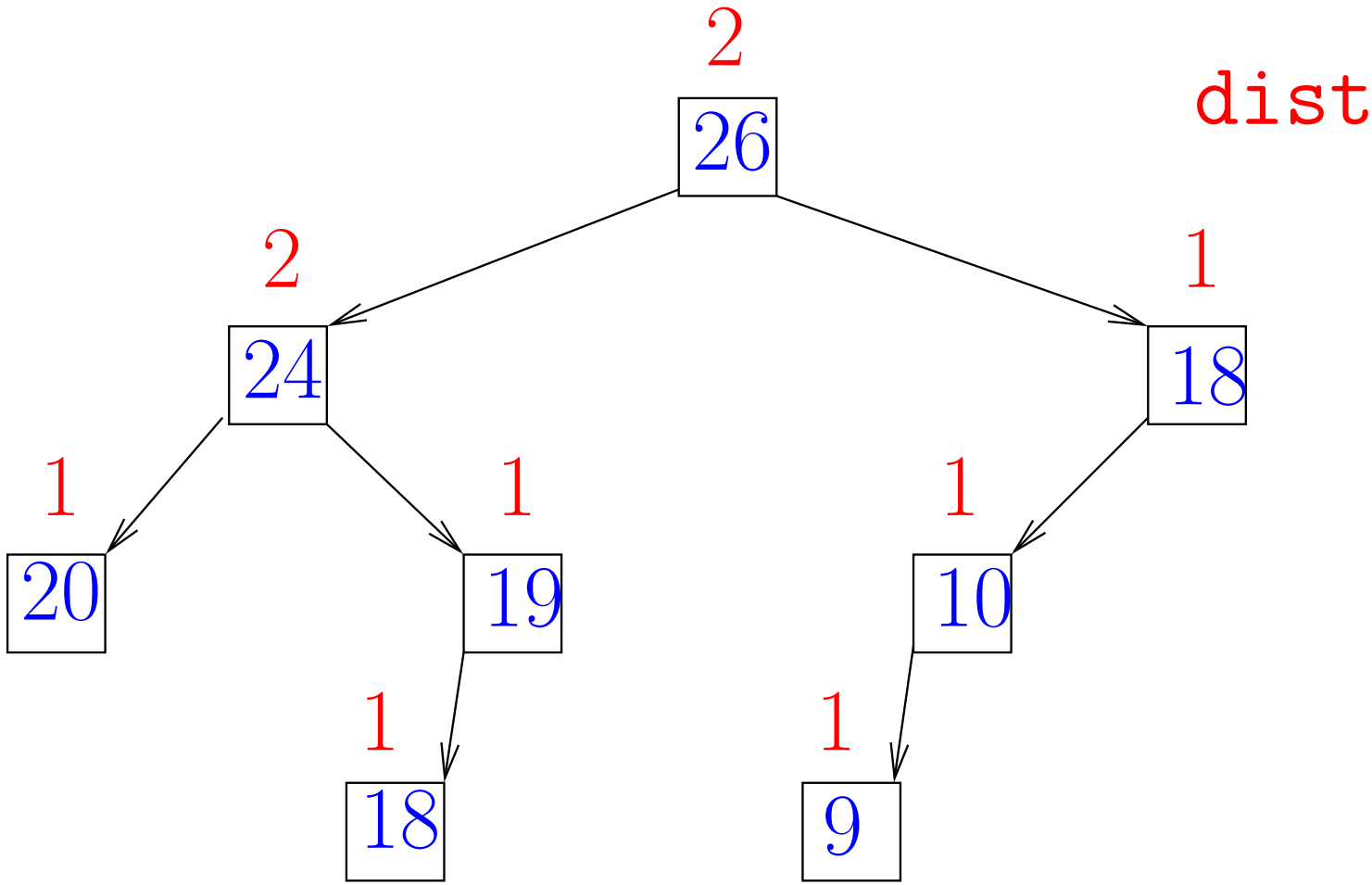
$\texttt{pai}[\texttt{x}] :=$ pai do nó $\texttt{x}$

Um **heap esquerdista** $\texttt{H}$ é uma árvore esquerdista que satisfaz

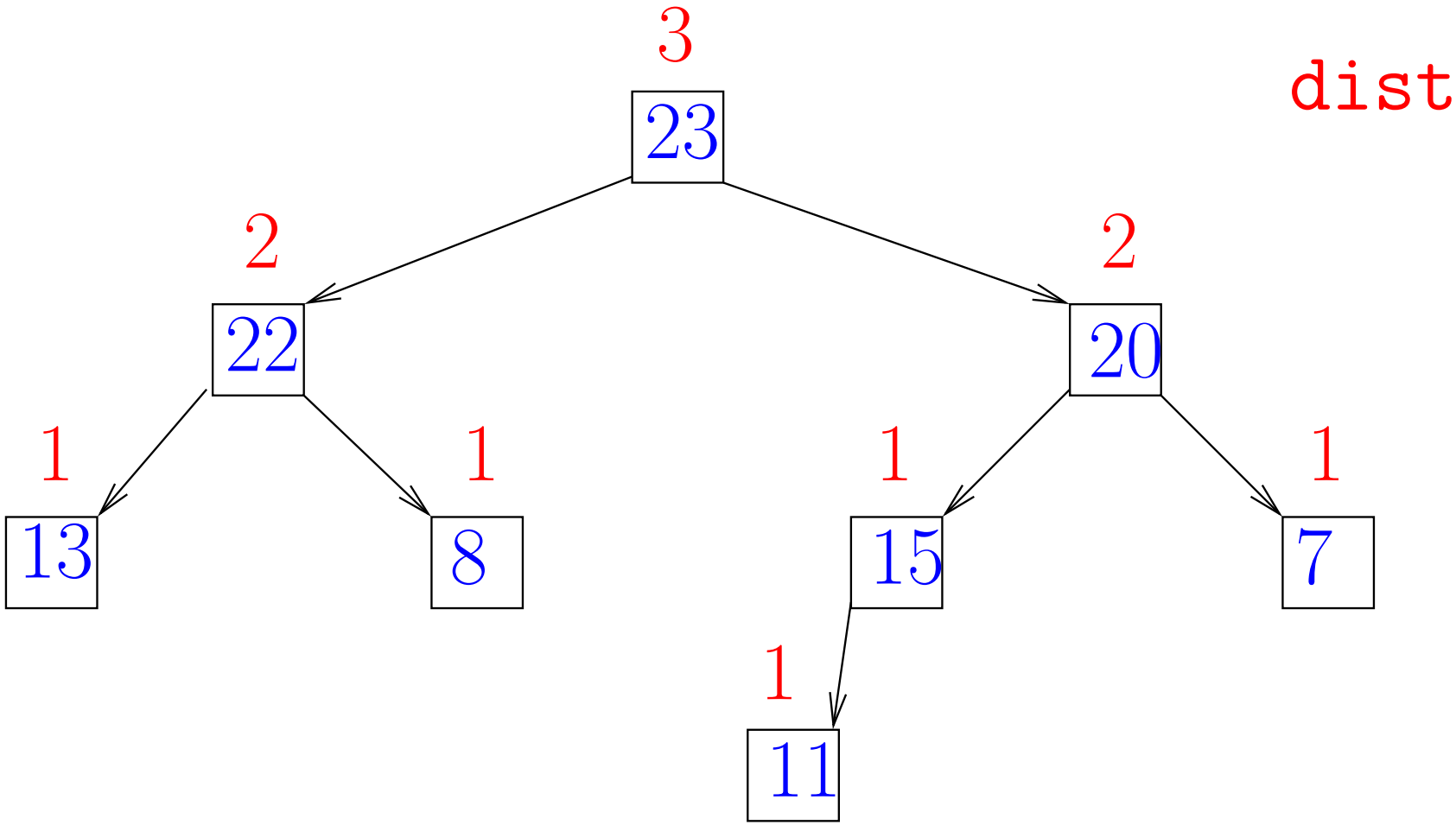
$$\texttt{prior}[\texttt{pai}[\texttt{x}]] \geq \texttt{prior}[\texttt{x}]$$

para todo nó $\texttt{x} \neq \texttt{raiz}[\texttt{H}]$.

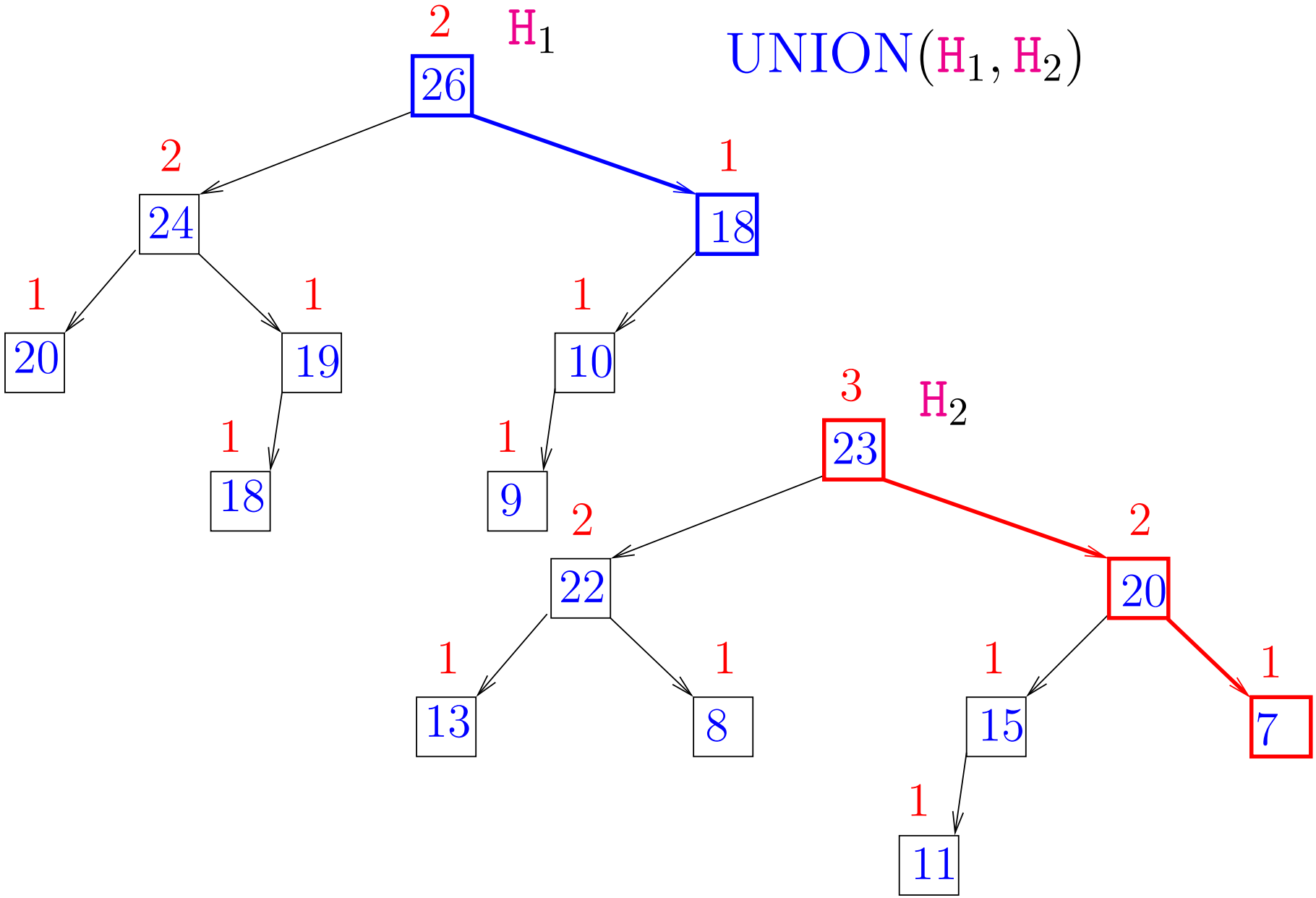
Heap esquerdista



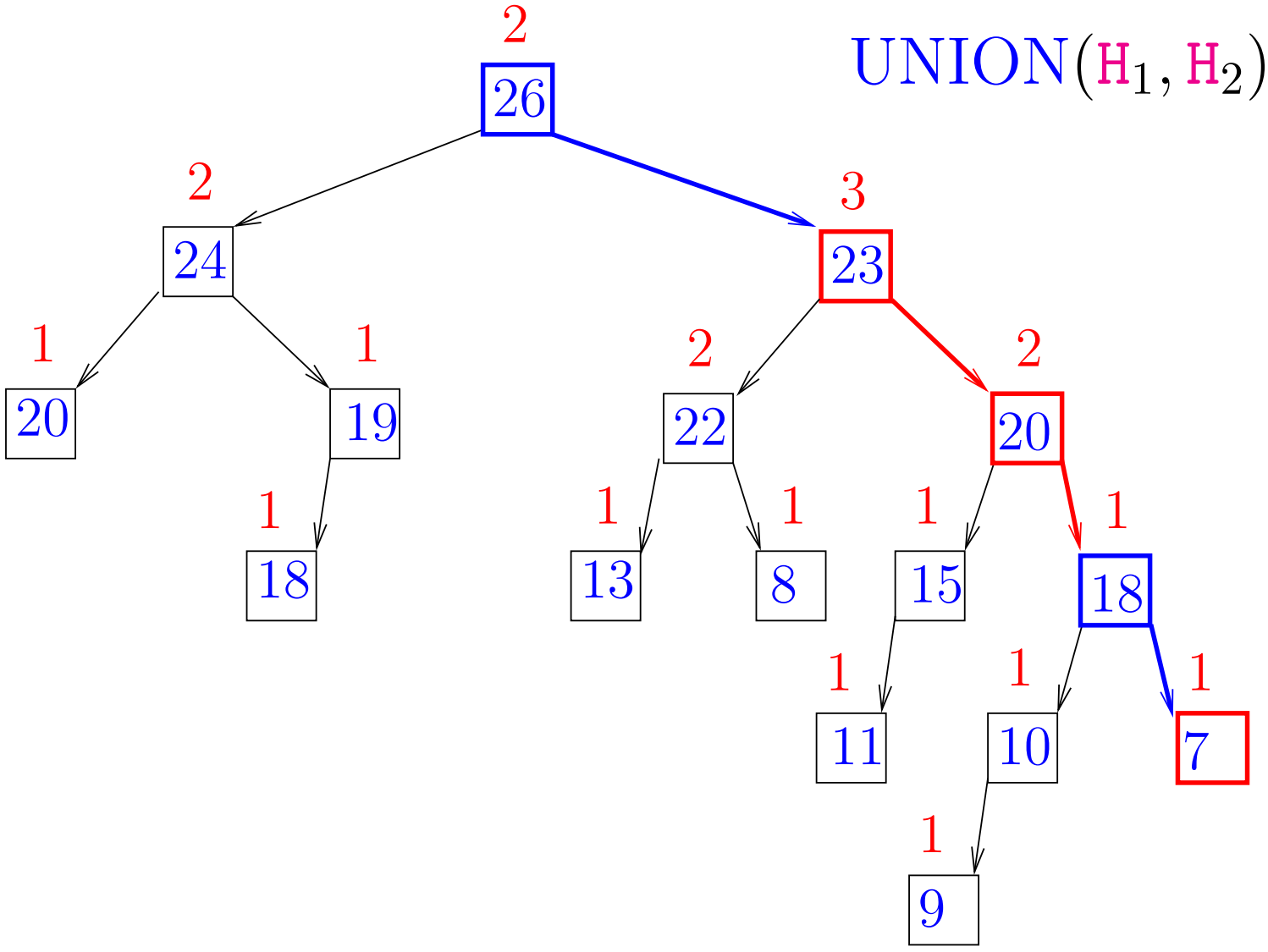
Heap esquerdista



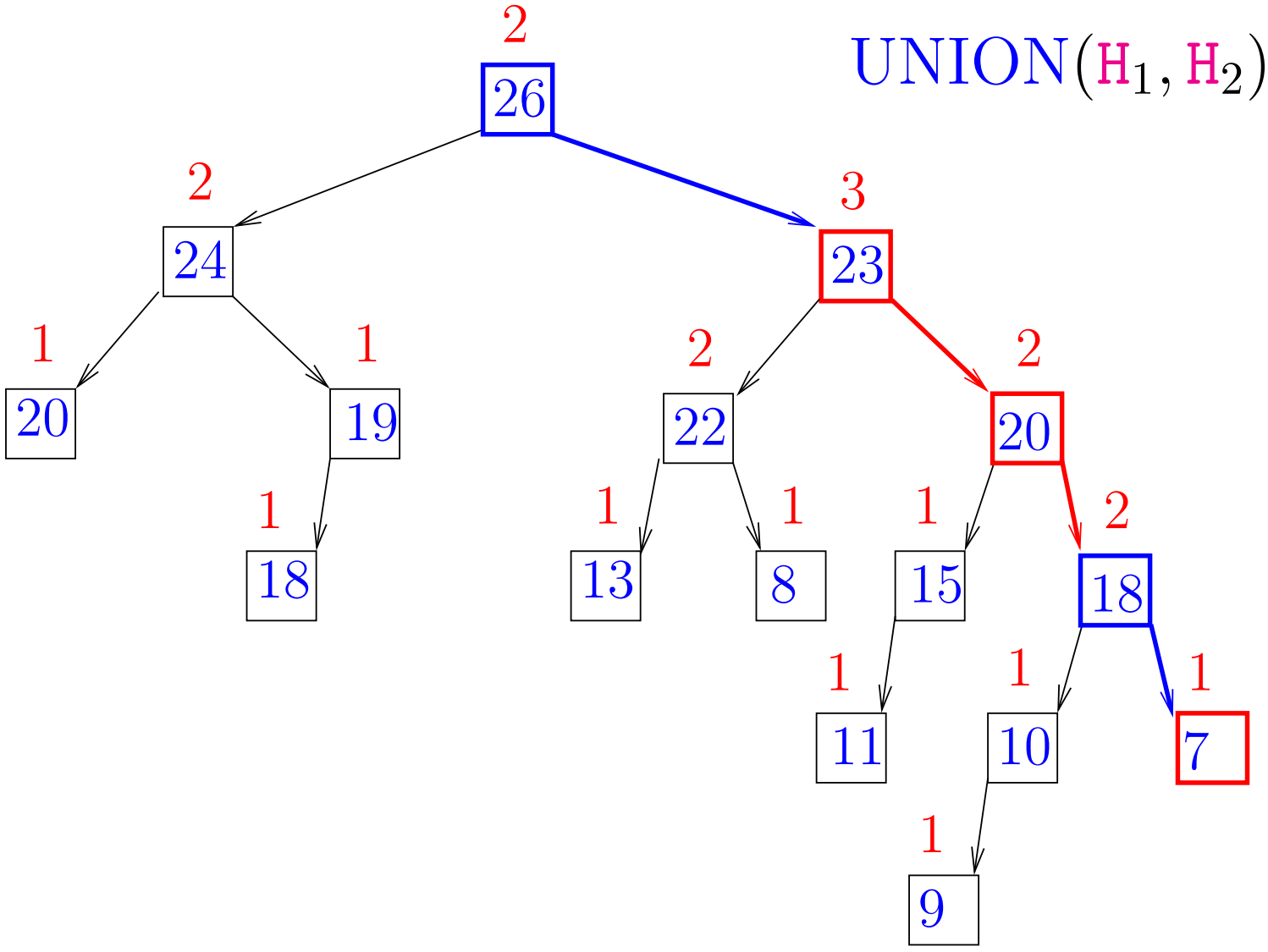
Rotina básica de manipulação



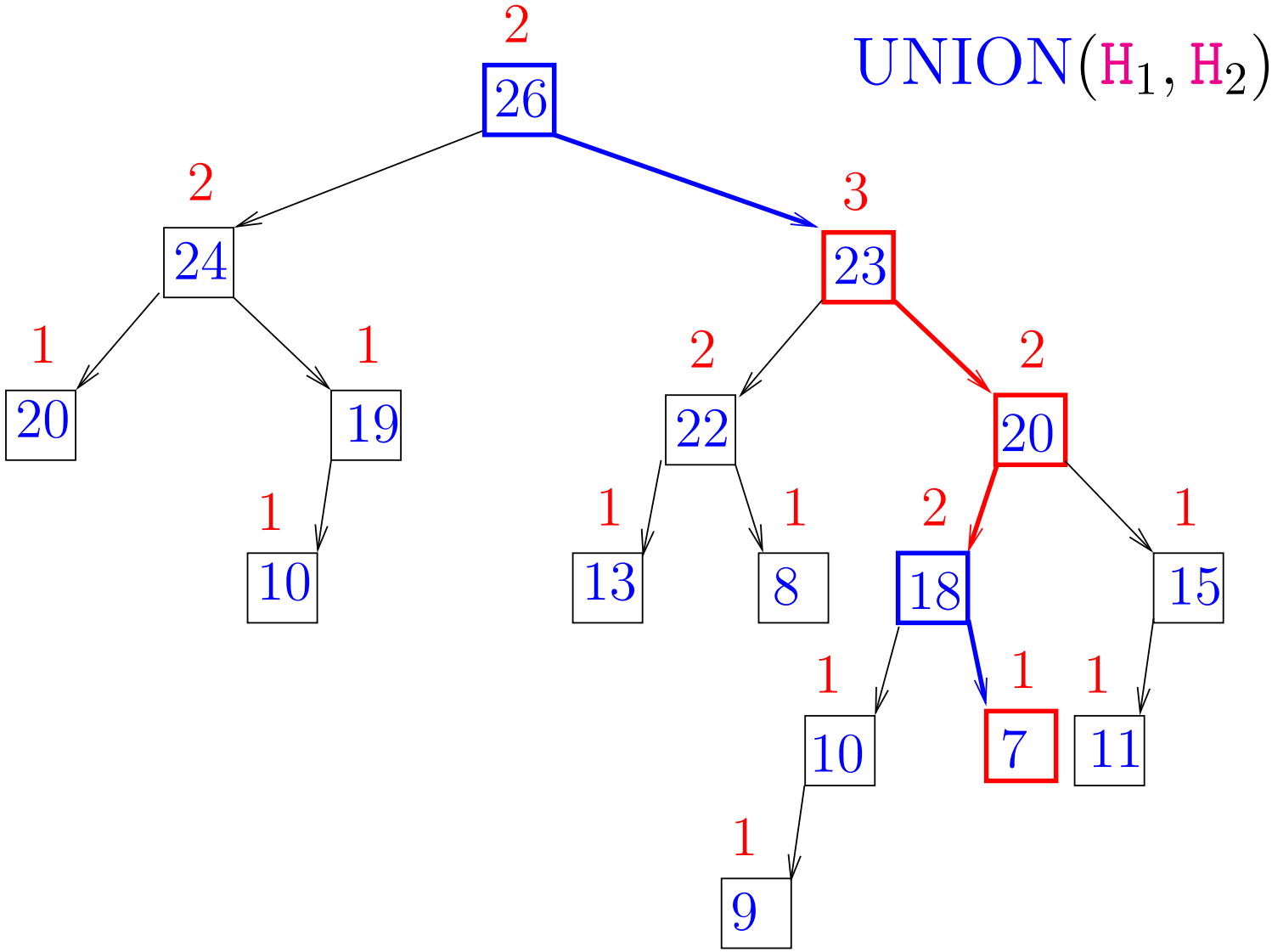
Rotina básica de manipulação



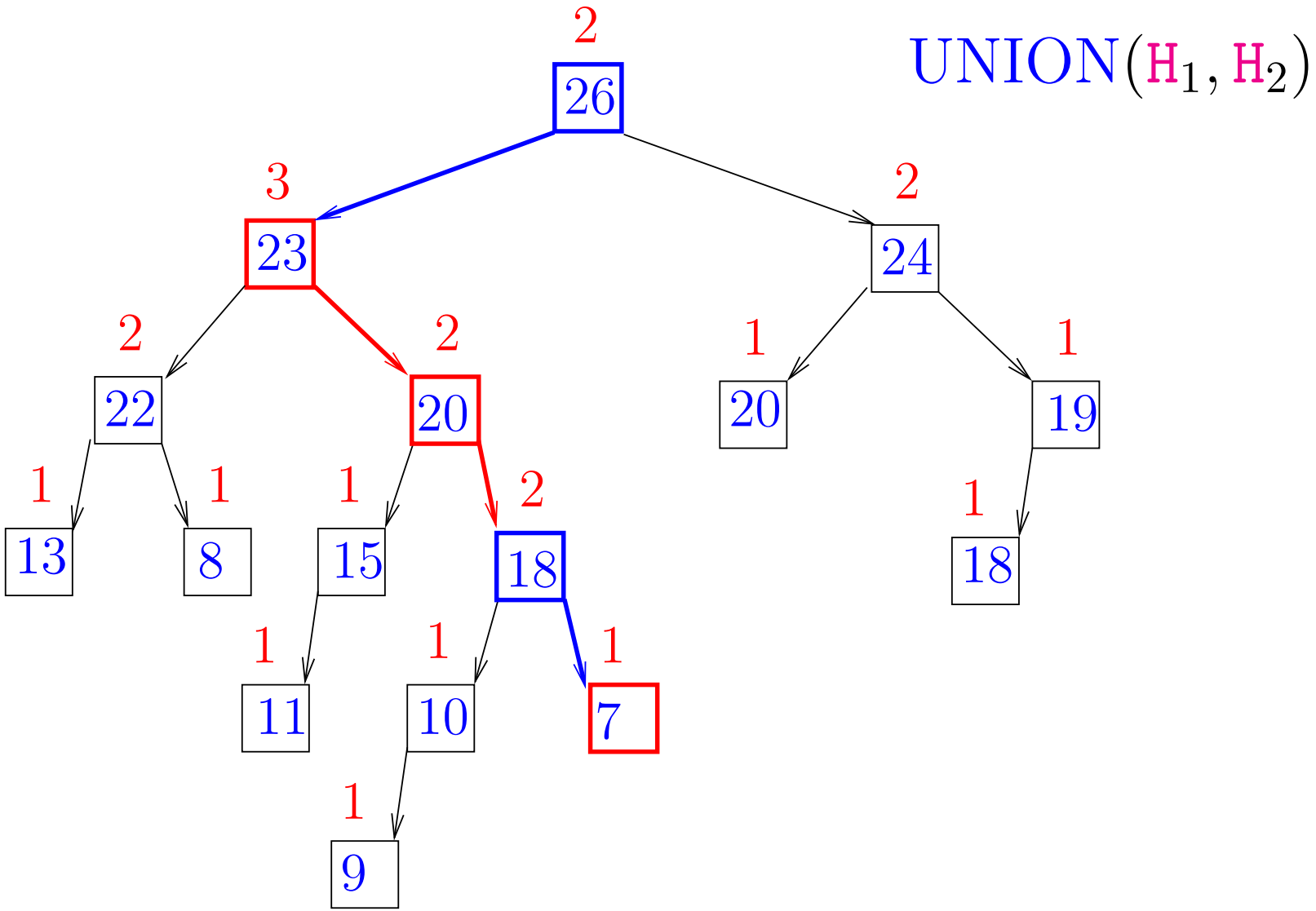
Rotina básica de manipulação



Rotina básica de manipulação



Rotina básica de manipulação



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

Consumo de tempo

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

$$\texttt{dcomp}(\texttt{raiz}[\texttt{H}_1]) + \texttt{dcomp}(\texttt{raiz}[\texttt{H}_2]) = \mathrm{O}(\lg \texttt{m})$$

onde $\texttt{m} = \texttt{tam}(\texttt{raiz}[\texttt{H}_1]) + \texttt{tam}(\texttt{raiz}[\texttt{H}_2])$

O consumo de tempo do algoritmo `LeftistHeapMerge` é $\mathrm{O}(\lg \texttt{m})$.

Fila com heap esquerdista

Rotina que insere um nó $\texttt{x}$ em um heap esquerdista $\texttt{H}$.

A rotina supõe que $\texttt{prior}[\texttt{x}]$ 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 $\mathrm{O}(\lg \texttt{m})$.

Fila com heap esquerdista

Rotina que remove e devolve o nó de maior prioridade de um heap esquerdista $\texttt{H}$.

```
LEFTIST-HEAP-EXTRACT (H)
1   x ← raiz[H]
2   H′ ← MakeLeftistHeap()
3   raiz[H′] ← esq[x]
4   raiz[H] ← dir[x]
5   H ← LeftistHeapMerge(H, H′)
6   devolva x
```

Consome tempo $\mathrm{O}(\lg \mathtt{m})$.