# Análise de Algoritmos

## Slides de Paulo Feofiloff

**[com erros do coelho e agora também da cris]**



# Heap

Um vetor $A[1 \ldots m]$ é um (max-)heap se

$$A[\lfloor i/2 \rfloor] \geq A[i]$$

para todo $i = 2, 3, \ldots, m$.

De uma forma mais geral, $A[j \ldots m]$ é um heap se

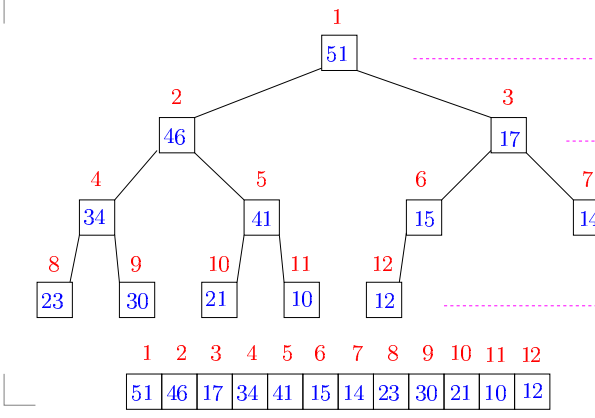
$$A[\lfloor i/2 \rfloor] \geq A[i]$$

para todo $i = 2j, 2j+1, 4j, \ldots, 4j+3, 8j, \ldots, 8j+7, \ldots$.

Neste caso também diremos que a subárvore com raiz $j$ é um heap.



# Exemplo



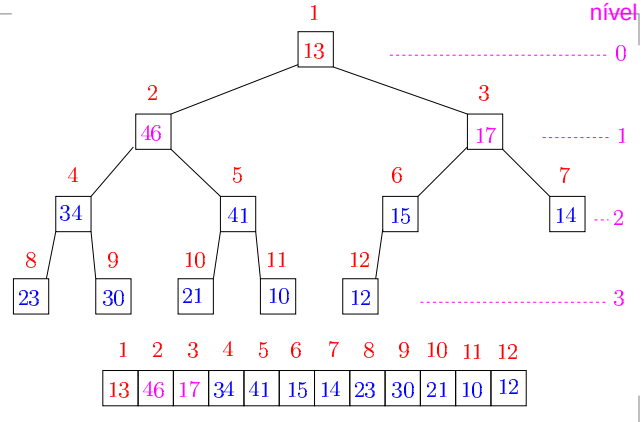

# Desce-Heap

cebe $A[1..m]$ e $i \geq 1$ tais que subárvores com raiz $2i$ e $+1$ são heaps e rearranja $A$ de modo que subárvore com z $i$ seja heap.

```
DESCE-HEAP (A, m, i)
1   e ← 2i
2   d ← 2i + 1
3   se e ≤ m e A[e] > A[i]
4         então maior ← e
5         senão maior ← i
6   se d ≤ m e A[d] > A[maior]
7         então maior ← d
8   se maior ≠ i
9         então A[i] ↔ A[maior]
10              DESCE-HEAP (A, m, maior)
```

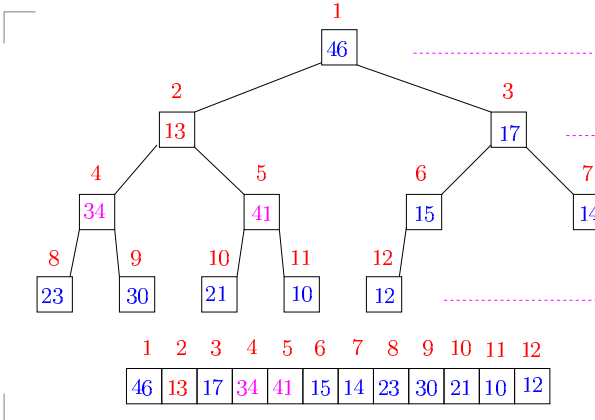


# Simulação

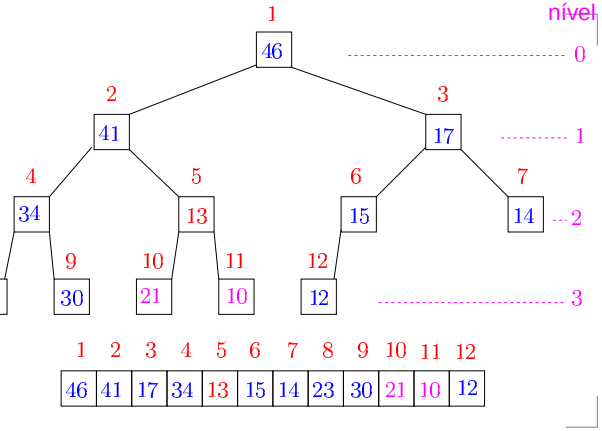





# Simulação

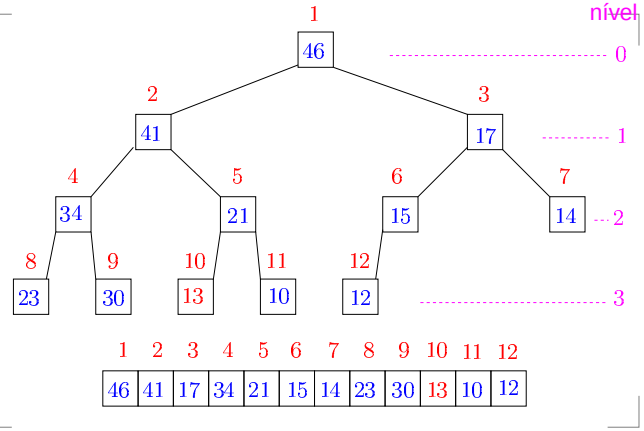


# Simulação



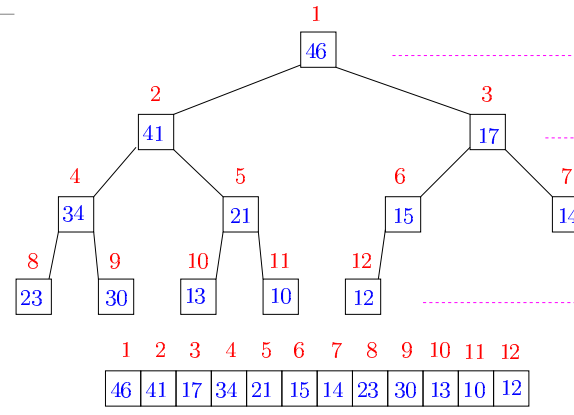



# Simulação






# Simulação




# Consumo de tempo

= altura de $i$ = $\lfloor \lg \frac{m+1}{i+1} \rfloor$

$h)$ := consumo de tempo no pior caso

| linha | | todas as execuções da linha |
|---|---|---|
| 1-3 | = | $3\,\Theta(1)$ |
| 4-5 | = | $2\,\mathrm{O}(1)$ |
| 6 | = | $\Theta(1)$ |
| 7 | = | $\mathrm{O}(1)$ |
| 8 | = | $\Theta(1)$ |
| 9 | = | $\mathrm{O}(1)$ |
| 10 | $\leq$ | $T(h-1)$ |
| total | $\leq$ | $T(h-1)+\Theta(5)+\mathrm{O}(2)$ |



# Consumo de tempo

$h$ := altura de $i$ = $\lfloor \lg \frac{m+1}{i+1} \rfloor$

$T(h)$ := consumo de tempo no pior caso

Recorrência associada:

$$T(h) \leq T(h-1) + \Theta(1),$$

pois altura de $maior$ é $h-1$.



# Consumo de tempo

$h$ := altura de $i$ = $\lfloor \lg \frac{m+1}{i+1} \rfloor$

$T(h)$ := consumo de tempo no pior caso

Recorrência associada:

$$T(h) \leq T(h-1) + \Theta(1),$$

pois altura de $maior$ é $h-1$.

Solução assintótica: $T(n)$ é ???.

# Consumo de tempo

= altura de $i$ = $\lfloor \lg \frac{m+1}{i+1} \rfloor$

$h)$ := consumo de tempo no pior caso

corrência associada:

$$T(h) \leq T(h-1) + \Theta(1),$$

s altura de *maior* é $h - 1$.

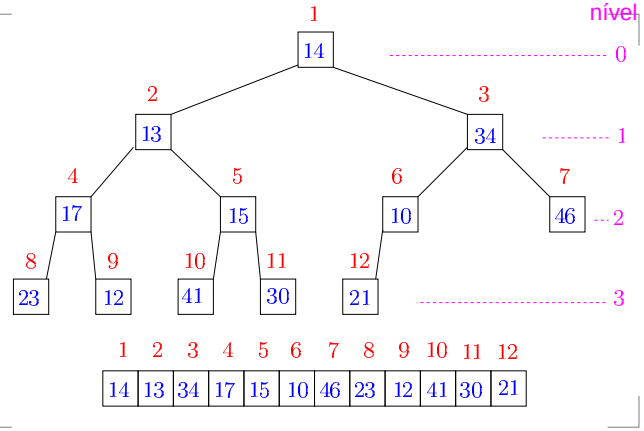
lução assintótica: $T(n)$ é $O(h)$.

mo $h \leq \lg m$, podemos dizer que:

O consumo de tempo do algoritmo DESCE-HEAP é $O(\lg m)$ (ou melhor ainda, $O(\lg \frac{m+1}{i+1})$).

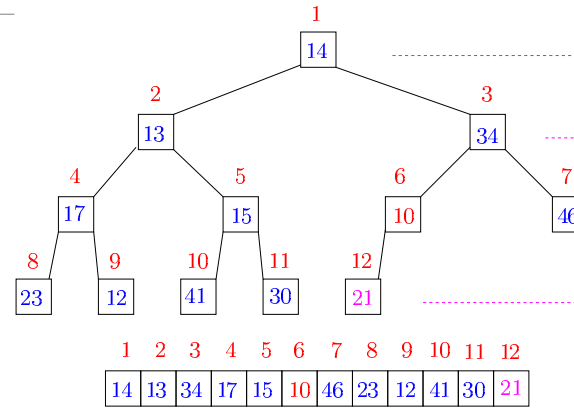


# Construção de um heap

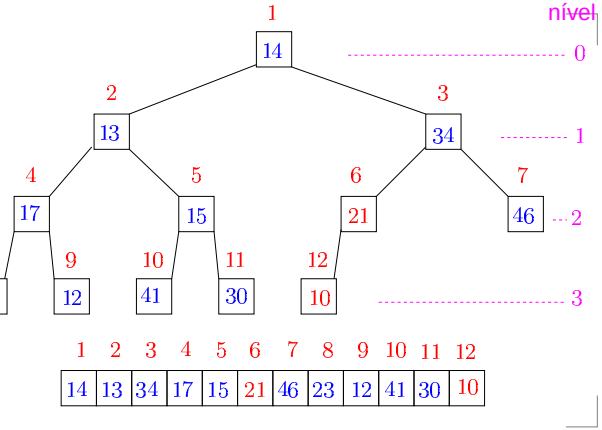





# Construção de um heap

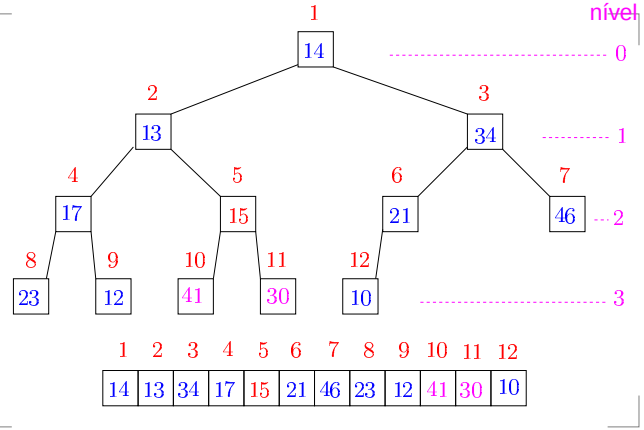



# Construção de um heap



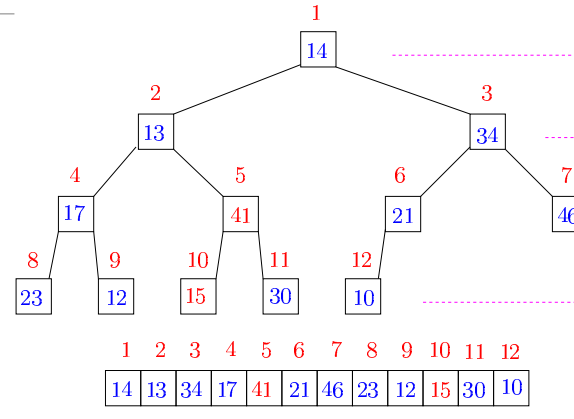



# Construção de um heap

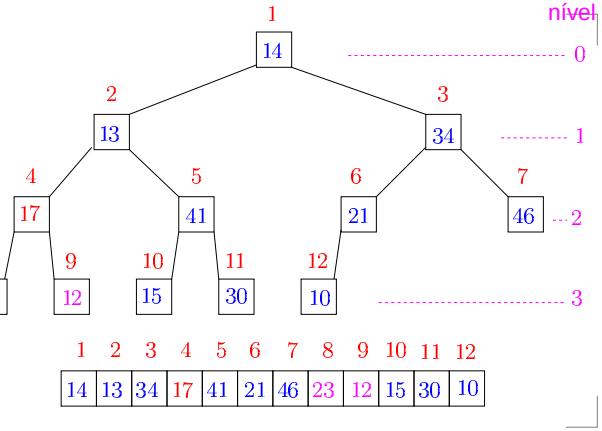





# Construção de um heap

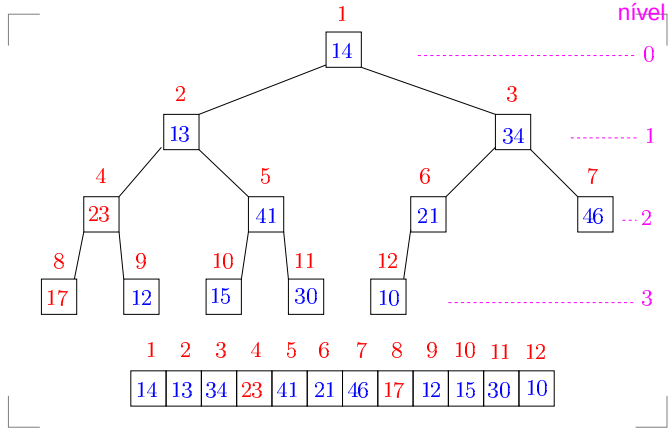


## Construção de um heap



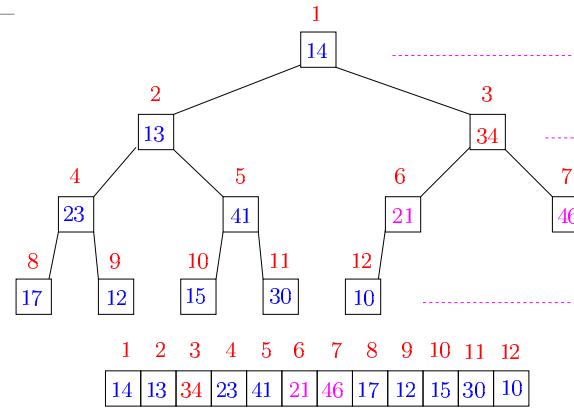



## Construção de um heap

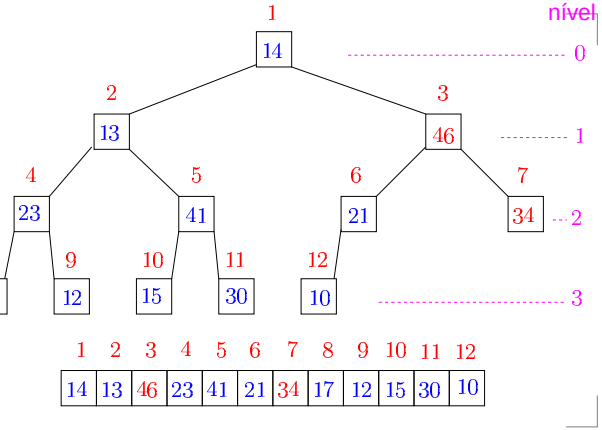





## Construção de um heap

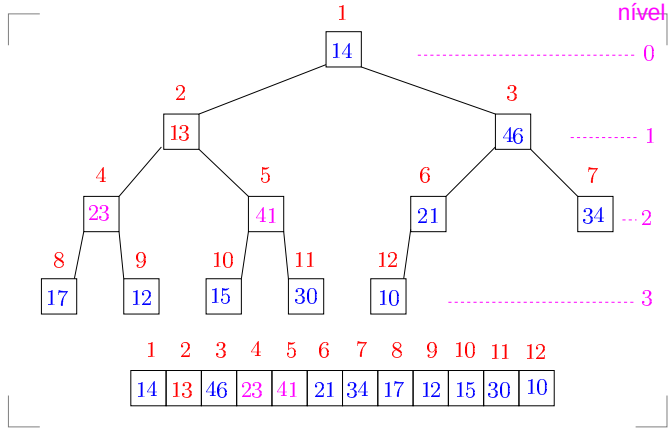



## Construção de um heap



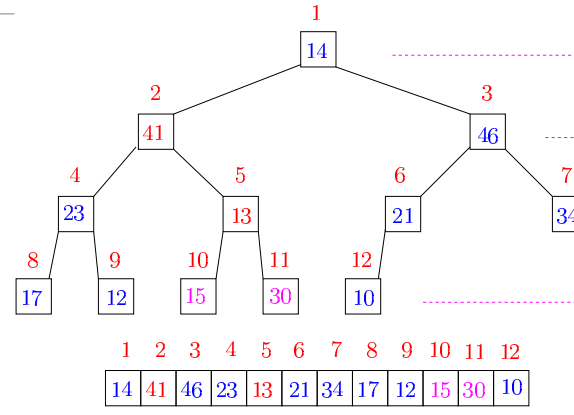



## Construção de um heap

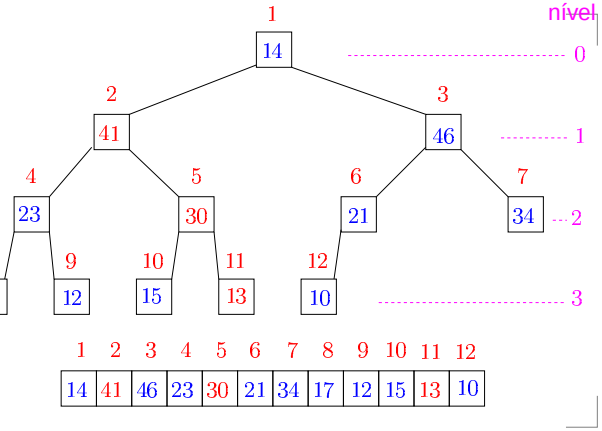





## Construção de um heap

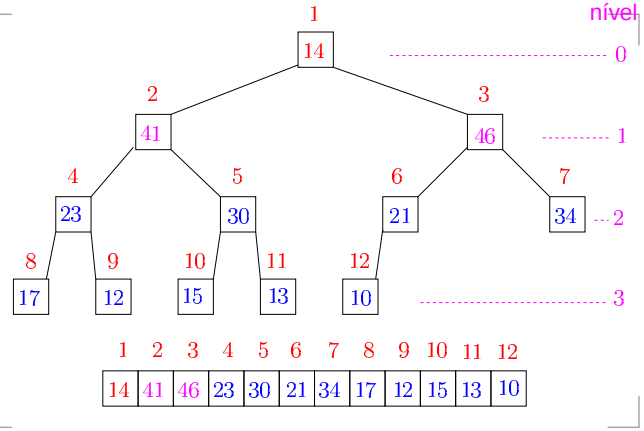


# Construção de um heap



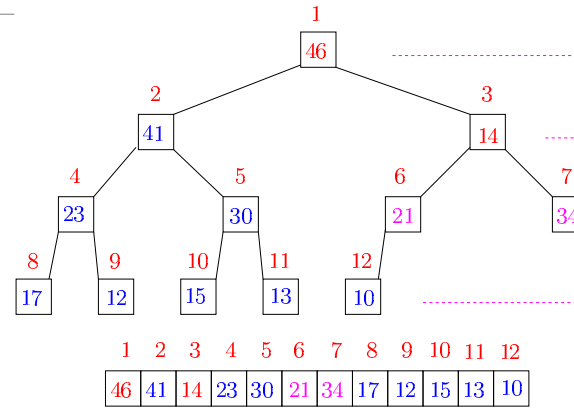



# Construção de um heap

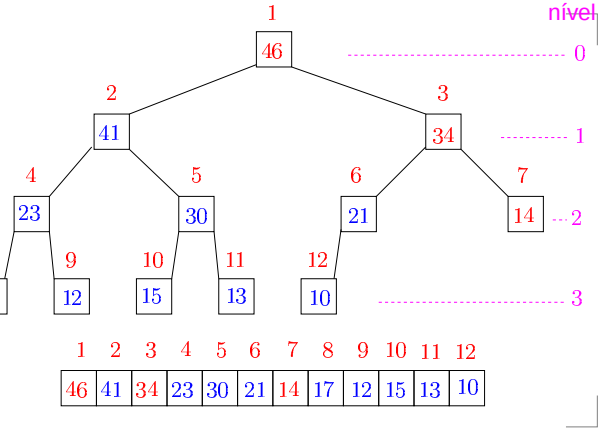





# Construção de um heap

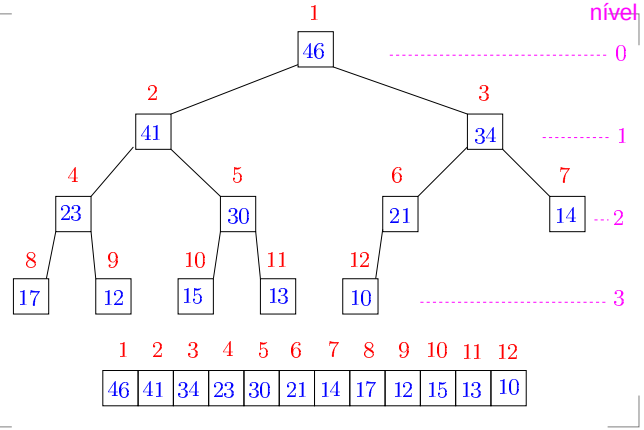



# Construção de um heap






# Construção de um heap






# Construção de um heap

Recebe um vetor $A[1..n]$ e rearranja $A$ para que seja h

CONSTRÓI-HEAP $(A, n)$
2 **para** $i \leftarrow \lfloor n/2 \rfloor$ **decrescendo até** 1 **faça**
3 DESCE-HEAP $(A, n, i)$

Relação invariante:

(i0) no início de cada iteração, $i+1, \ldots, n$ são raízes de heaps.

$T(n)$ := consumo de tempo no pior caso

# Construção de um heap

cebe um vetor $A[1..n]$ e rearranja $A$ para que seja heap.

CONSTRÓI-HEAP $(A, n)$
2 **para** $i \leftarrow \lfloor n/2 \rfloor$ **decrescendo até** 1 **faça**
3 DESCE-HEAP $(A, n, i)$

lação invariante:

no início de cada iteração, $i+1, \ldots, n$ são raízes de heaps.

$n)$ := consumo de tempo no pior caso

álise grosseira: $T(n)$ é $\frac{n}{2}\,\mathrm{O}(\lg n) = \mathrm{O}(n \lg n)$.

álise mais cuidadosa: $T(n)$ é ????.



# $T(n)$ é $\mathrm{O}(n)$

Prova: O consumo de DESCE-HEAP $(A, n, i)$ é proporcional a $h = \lfloor \lg \frac{n+1}{i+1} \rfloor$. Logo,

$$\begin{aligned} T(n) &= \sum_{h=1}^{\lfloor \lg n \rfloor} 2^{\lfloor \lg n \rfloor - h} h \\ &\leq \sum_{h=1}^{\lfloor \lg n \rfloor} \frac{n}{2^h} h \\ &\leq n \left( \frac{1}{2^1} + \frac{2}{2^2} + \frac{3}{2^3} + \cdots + \frac{\lfloor \lg n \rfloor}{2^{\lfloor \lg n \rfloor}} \right) \\ &< n \frac{1/2}{(1-1/2)^2} \\ &= 2n. \end{aligned}$$

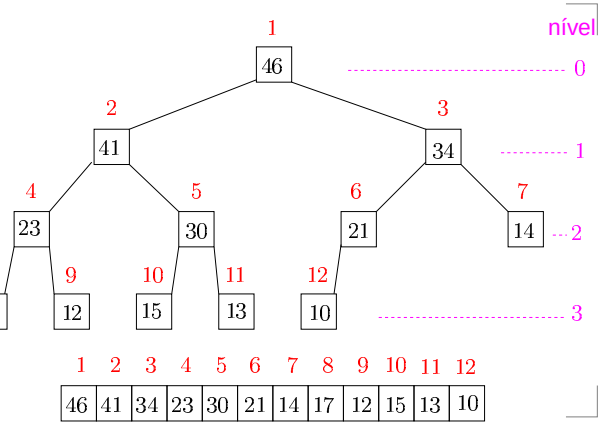


# $T(n)$ é $\mathrm{O}(n)$

Prova: O consumo de tempo de DESCE-HEAP $(A, n, i)$ é $\mathrm{O}(h) = \mathrm{O}(\lfloor \lg \frac{n+1}{i+1} \rfloor)$. Logo,

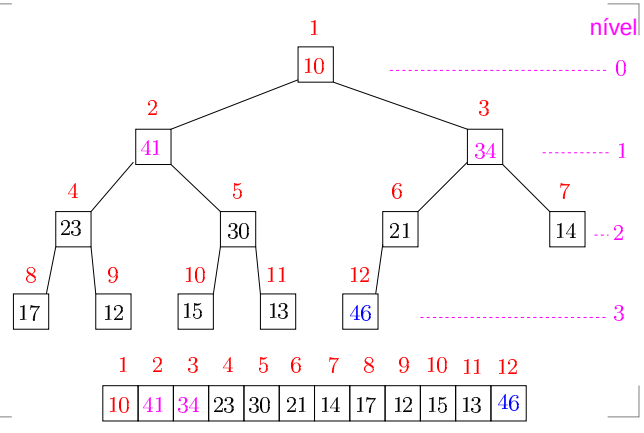
$$\begin{aligned} T(n) &= \sum_{h=0}^{\lfloor \lg n \rfloor} 2^{\lfloor \lg n \rfloor - h} \mathrm{O}(h) \\ &= \mathrm{O}\left( n \sum_{h=0}^{\lfloor \lg n \rfloor} \frac{h}{2^h} \right) \\ &= \mathrm{O}\left( n \sum_{h=0}^{\infty} \frac{h}{2^h} \right) \\ &= \mathrm{O}\left( n \frac{1/2}{(1-1/2)^2} \right) \\ &= \mathrm{O}(2n) = \mathrm{O}(n) \end{aligned}$$

# Heap sort



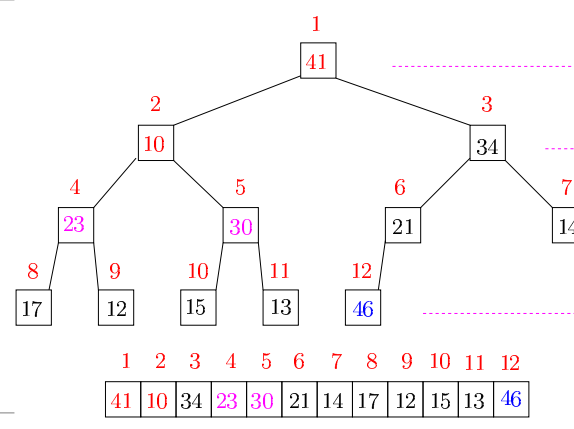



# Heap sort

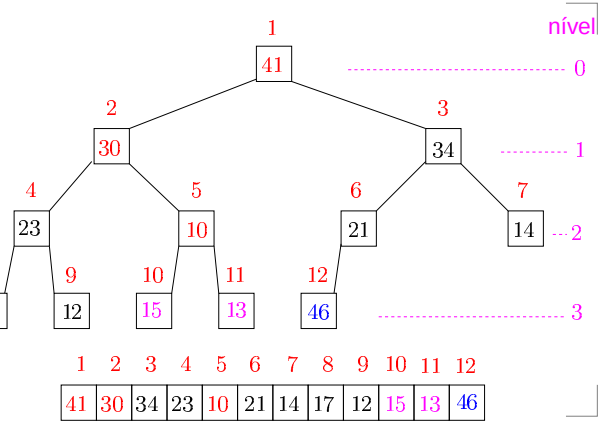





# Heap sort

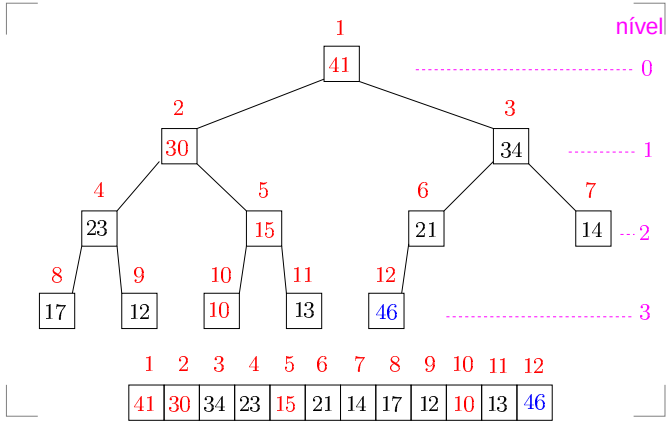


# Heap sort



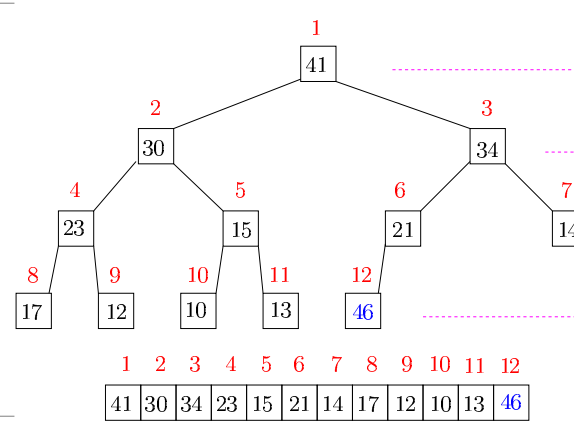



# Heap sort

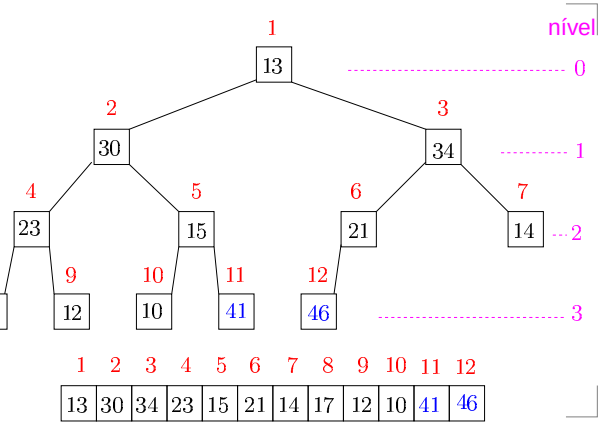





# Heap sort

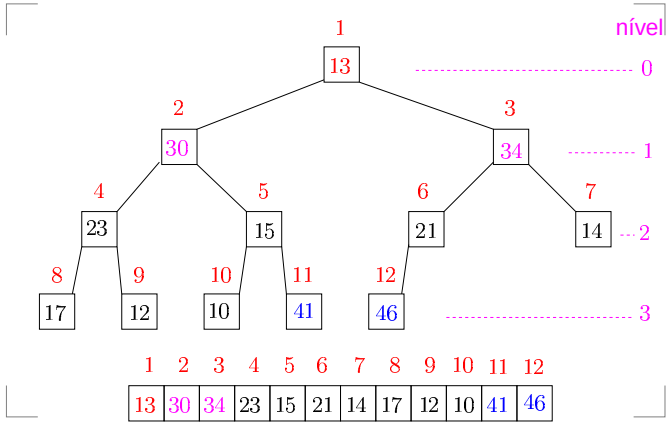



# Heap sort



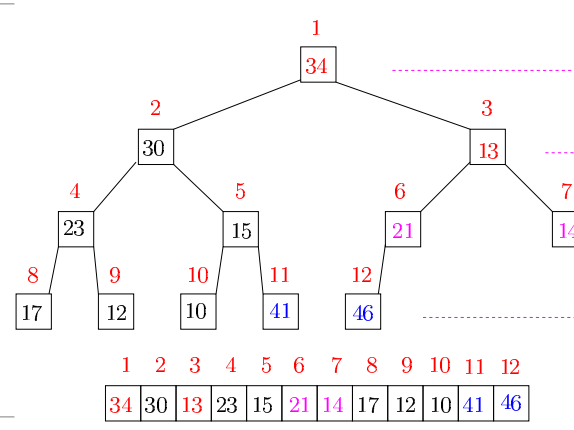



# Heap sort

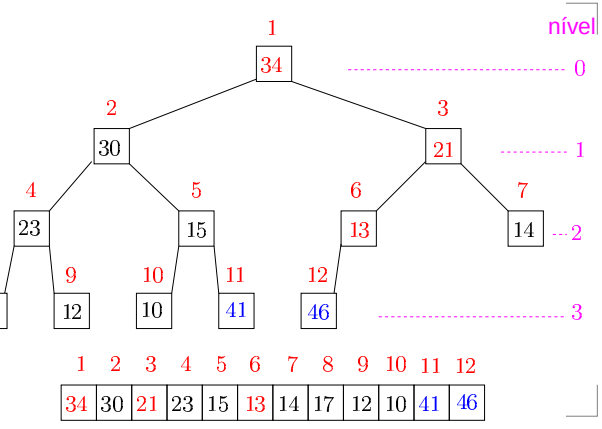





# Heap sort

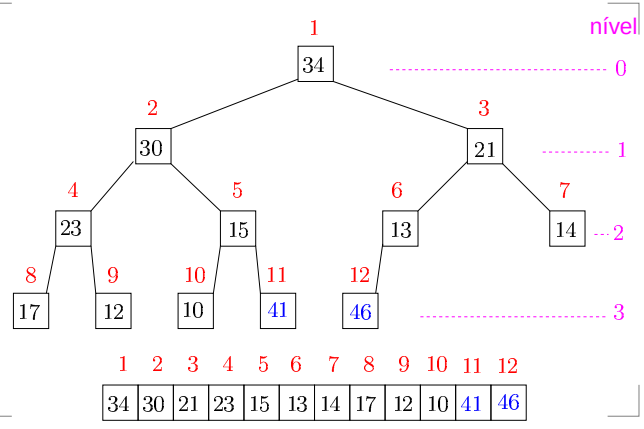


# Heap sort



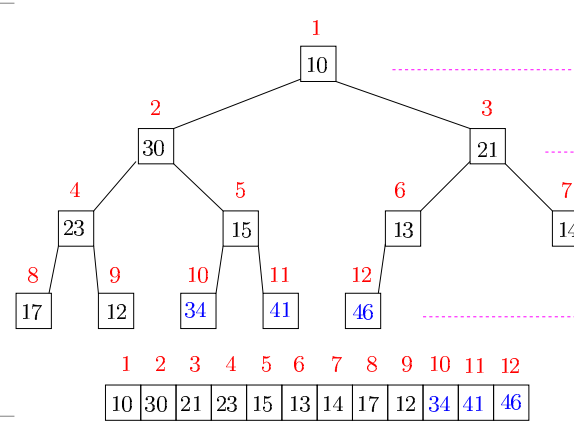



# Heap sort

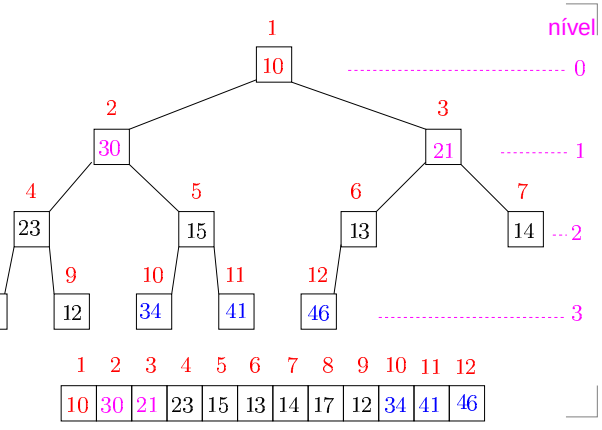





# Heap sort

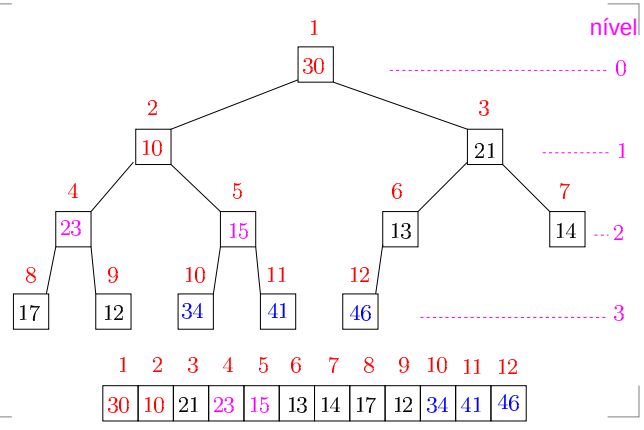



# Heap sort



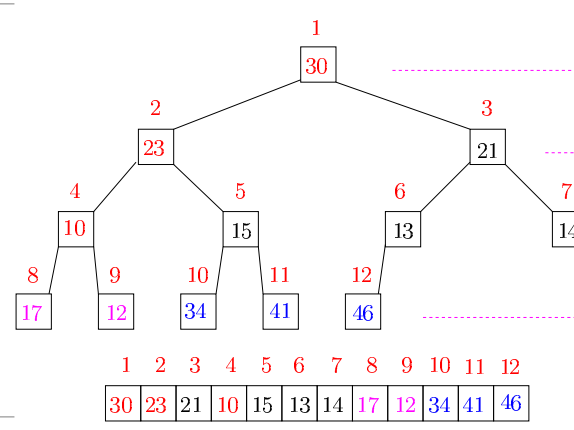



# Heap sort

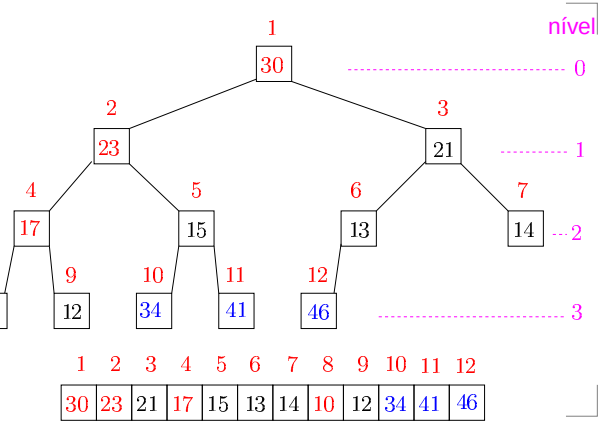





# Heap sort

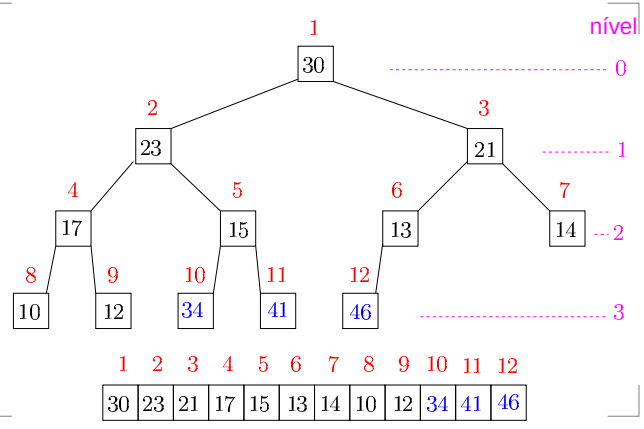


# Heap sort



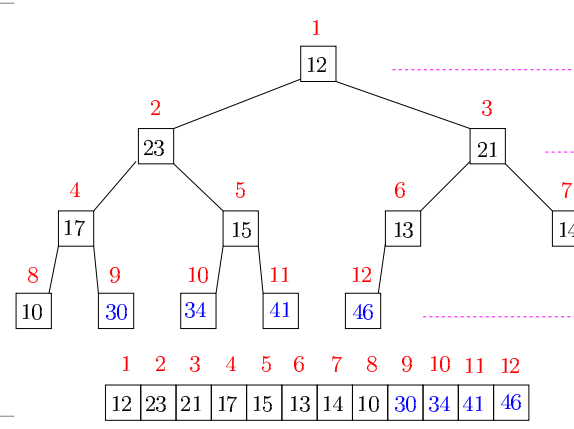



# Heap sort






# Heap sort




# Heap sort

goritmo rearranja $A[1\,..\,n]$ em ordem crescente.

HEAPSORT $(A, n)$

```
0  CONSTRÓI-HEAP (A, n)      ▷ pré-processamento
1  m ← n
2  para i ← n decrescendo até 2 faça
3      A[1] ↔ A[i]
4      m ← m − 1
5      DESCE-HEAP (A, m, 1)
```

lações invariantes: Na linha 2 vale que:

- $A[m\,..\,n]$ é crescente;
- $A[1\,..\,m] \le A[m+1]$;
- $A[1\,..\,m]$ é um heap.



# Consumo de tempo

| linha | | todas as execuções da linha |
|---|---|---|
| 0 | = | $\Theta(n)$ |
| 1 | = | $\Theta(1)$ |
| 2 | = | $\Theta(n)$ |
| 3 | = | $\Theta(n)$ |
| 4 | = | $\Theta(n)$ |
| 6 | = | $n\mathrm{O}(\lg n)$ |
| total | = | $n\mathrm{O}(\lg n) + \Theta(4n+1) = \mathrm{O}(n \lg n)$ |

O consumo de tempo do algoritmo HEAPSORT é $\mathrm{O}(n \lg n)$.



# Exercícios

**Exercício 9.A**
A altura de $i$ em $A[1\,..\,m]$ é o comprimento da mais longa seqüência da forma

$$\langle \mathrm{filho}(i), \mathrm{filho}(\mathrm{filho}(i)), \mathrm{filho}(\mathrm{filho}(\mathrm{filho}(i))), \ldots \rangle$$

onde $\mathrm{filho}(i)$ vale $2i$ ou $2i+1$. Mostre que a altura de $i$ é $\lfloor \lg \frac{m}{i} \rfloor$.
É verdade que $\lfloor \lg \frac{m}{i} \rfloor = \lfloor \lg m \rfloor - \lfloor \lg i \rfloor$?

**Exercício 9.B**
Mostre que um heap $A[1\,..\,m]$ tem no máximo $\lceil m/2^{h+1} \rceil$ nós com altura $h$.

**Exercício 9.C**
Mostre que $\lceil m/2^{h+1} \rceil \le m/2^h$ quando $h \le \lfloor \lg m \rfloor$.

**Exercício 9.D**
Mostre que um heap $A[1\,..\,m]$ tem no mínimo $\lfloor m/2^{h+1} \rfloor$ nós com altura $h$.

**Exercício 9.E**
Considere um heap $A[1\,..\,m]$; a raiz do heap é o elemento de índice 1. Seja $m'$ o núm
de elementos do "sub-heap esquerdo", cuja raiz é o elemento de índice 2. Seja $m''$ o
número de elementos do "sub-heap direito", cuja raiz é o elemento de índice 3. Mostre

$$m'' \le m' < 2m/3\,.$$

# Mais execícios

**rcício 9.F**

tre que a solução da recorrência

$$\begin{aligned} T(1) &= 1 \\ T(k) &\leq T(2k/3) + 5 \quad \text{para } k \geq 2 \end{aligned}$$

$\log k)$. Mais geral: mostre que se $T(k) = T(2k/3) + \mathrm{O}(1)$ então $\mathrm{O}(\log k)$.
iosidade: Essa é a recorrência do DESCE-HEAP $(A, m, i)$ se interpretarmos $k$
o sendo o número de nós na subárvore com raiz $i$).

**rcício 9.G**

eva uma versão iterativa do algoritmo DESCE-HEAP. Faça uma análise do
umo de tempo do algoritmo.



# Mais exercícios ainda

**Exercício 9.H**

Discuta a seguinte variante do algoritmo DESCE-HEAP:

```
D-H (A, m, i)
1   e ← 2i
2   d ← 2i + 1
3   se e ≤ m e A[e] > A[i]
4         então A[i] ↔ A[e]
5               D-H (A, m, e)
6   se d ≤ m e A[d] > A[i]
7         então A[i] ↔ A[d]
8               D-H (A, m, d)
```