

Análise de Algoritmos

**Parte destes slides são adaptações de slides
do Prof. Paulo Feofiloff e do Prof. José Coelho de Pina.**

Heapsort

CLRS 6

Heap

Um vetor $A[1..m]$ é um (max-)heap se

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

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

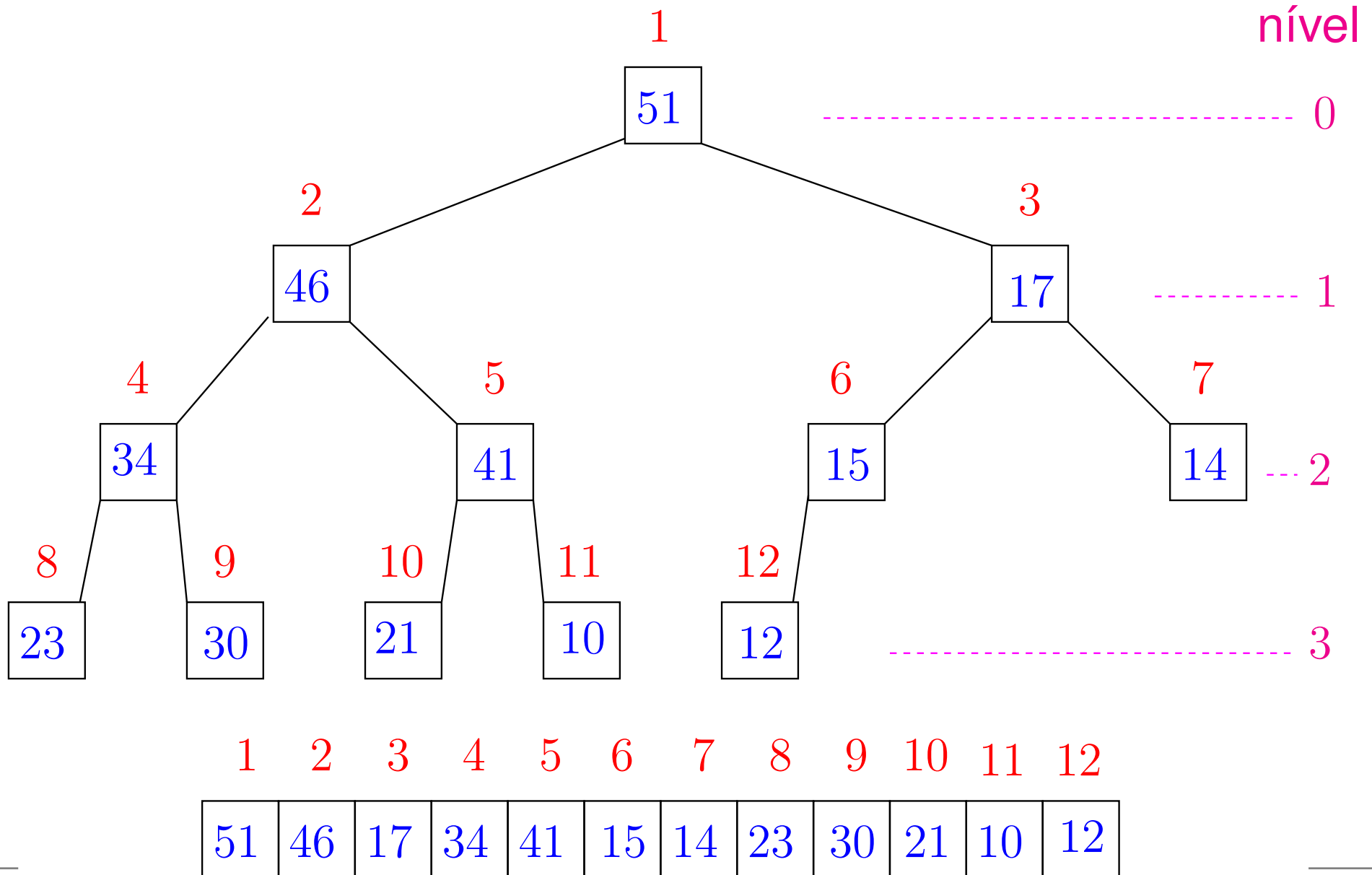
De uma forma mais geral, $A[j..m]$ é um heap se

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

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

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

Exemplo



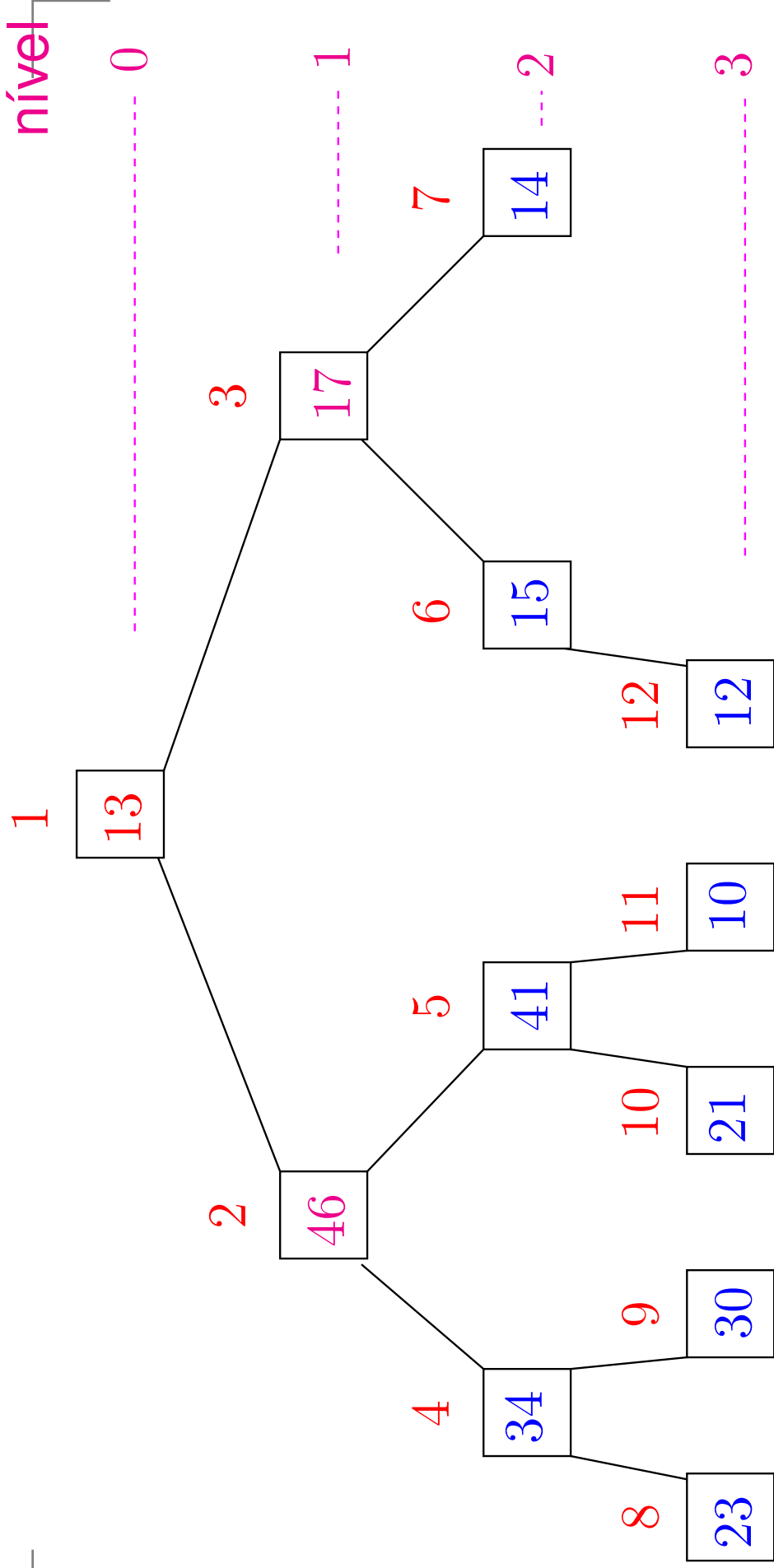
Desce-Heap

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

DESCE-HEAP (A, m, i)

```
1   $e \leftarrow 2i$ 
2   $d \leftarrow 2i + 1$ 
3  se  $e \leq m$  e  $A[e] > A[i]$ 
4      então  $maior \leftarrow e$ 
5      senão  $maior \leftarrow i$ 
6  se  $d \leq m$  e  $A[d] > A[maior]$ 
7      então  $maior \leftarrow d$ 
8  se  $maior \neq i$ 
9      então  $A[i] \leftrightarrow A[maior]$ 
10     DESCE-HEAP ( $A, m, maior$ )
```

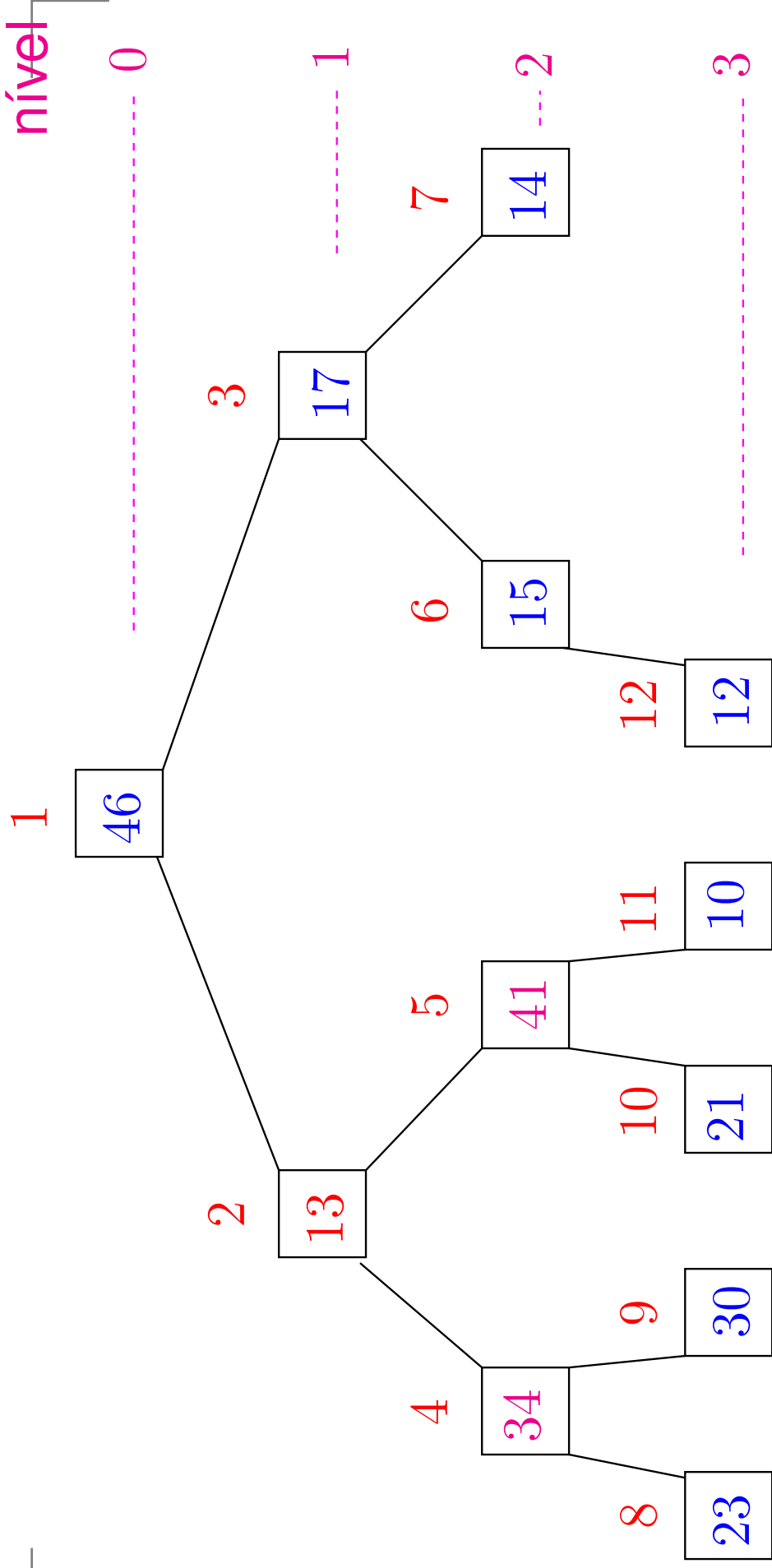
Simulação



1 2 3 4 5 6 7 8 9 10 11 12

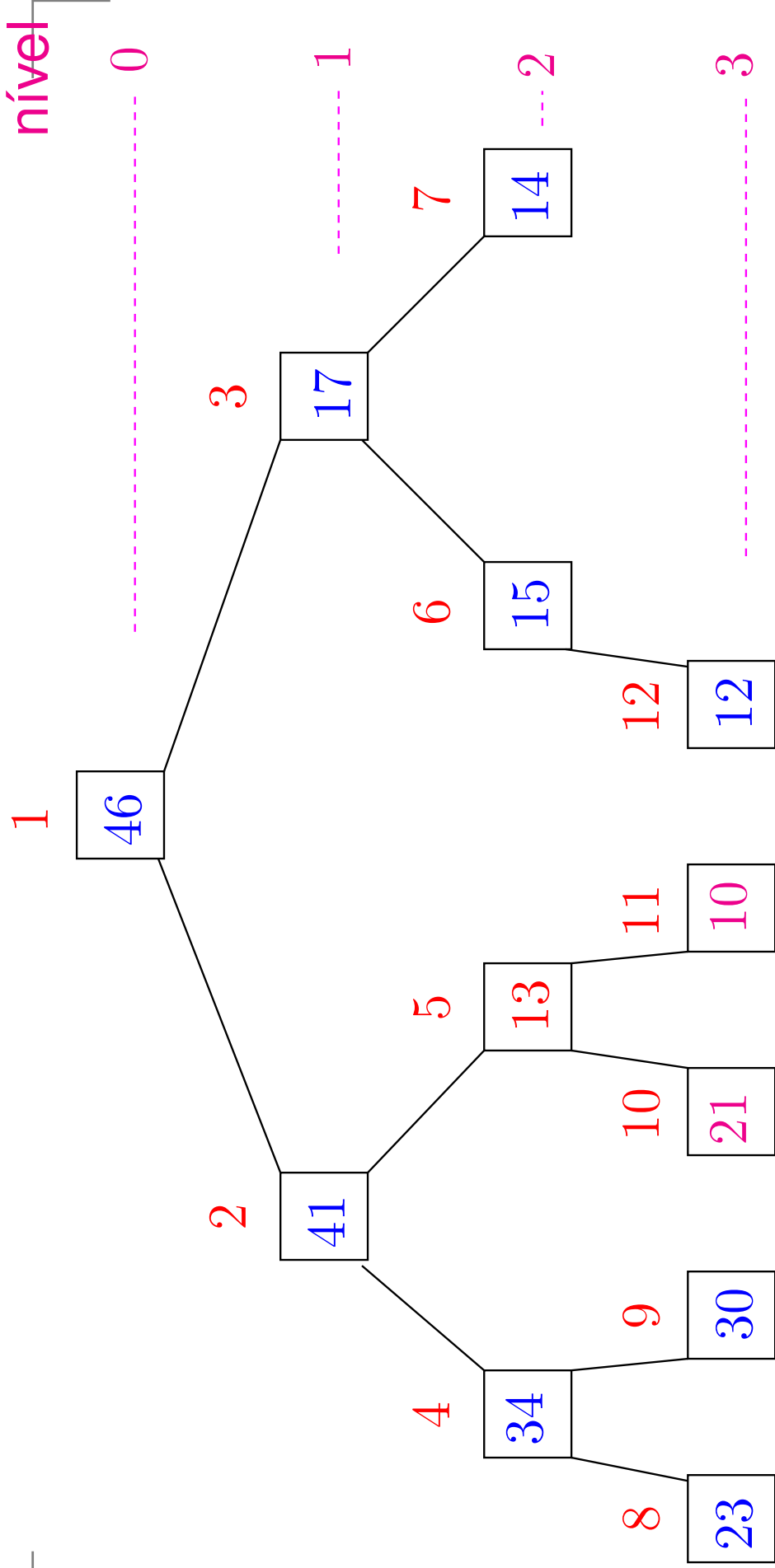
13	46	17	34	41	15	14	23	30	21	10	12
----	----	----	----	----	----	----	----	----	----	----	----

Simulação



1	2	3	4	5	6	7	8	9	10	11	12
46	13	17	34	41	15	14	23	30	21	10	12

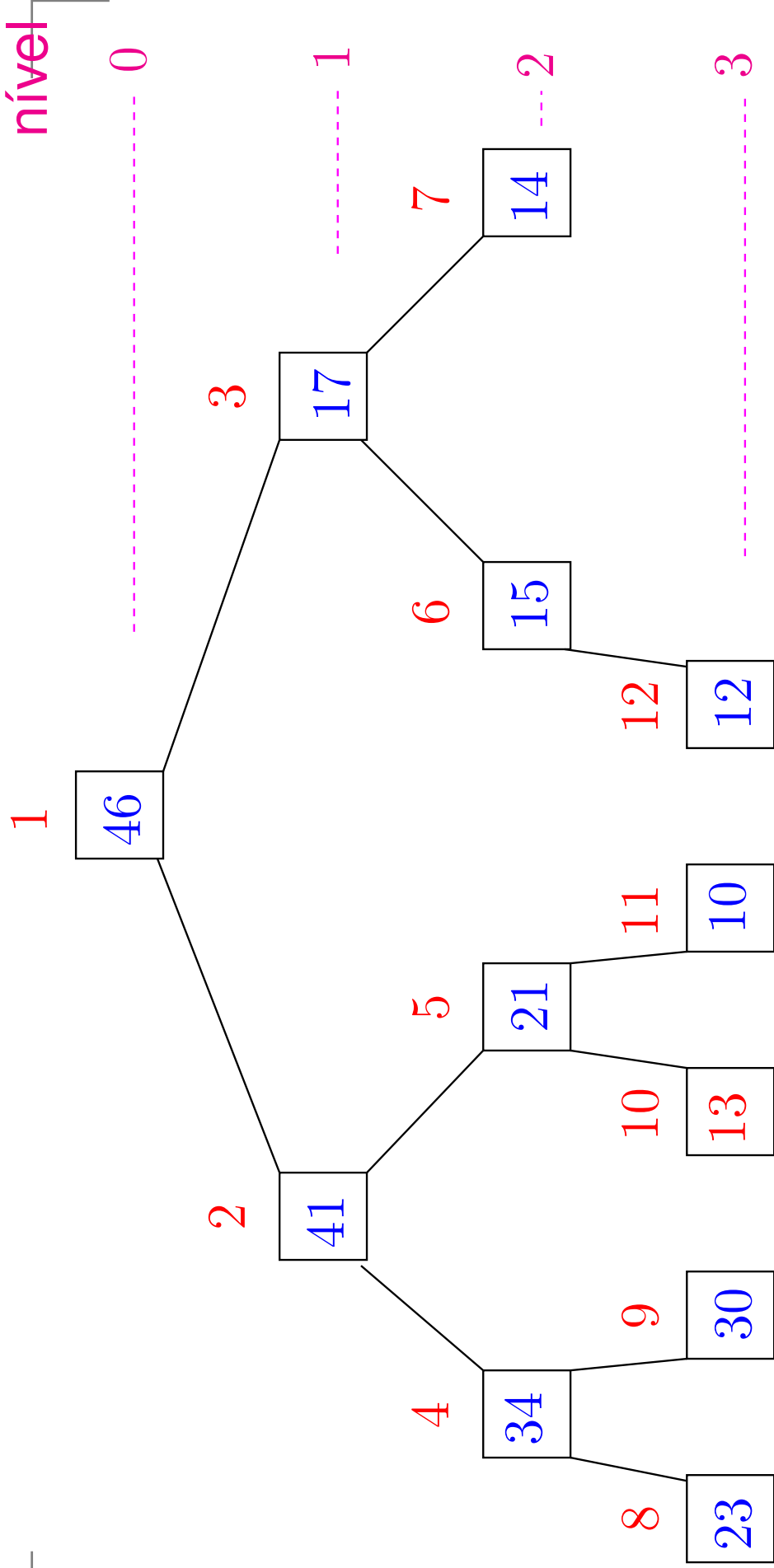
Simulação



1 2 3 4 5 6 7 8 9 10 11 12

46	41	17	34	13	15	14	23	30	21	10	12
----	----	----	----	----	----	----	----	----	----	----	----

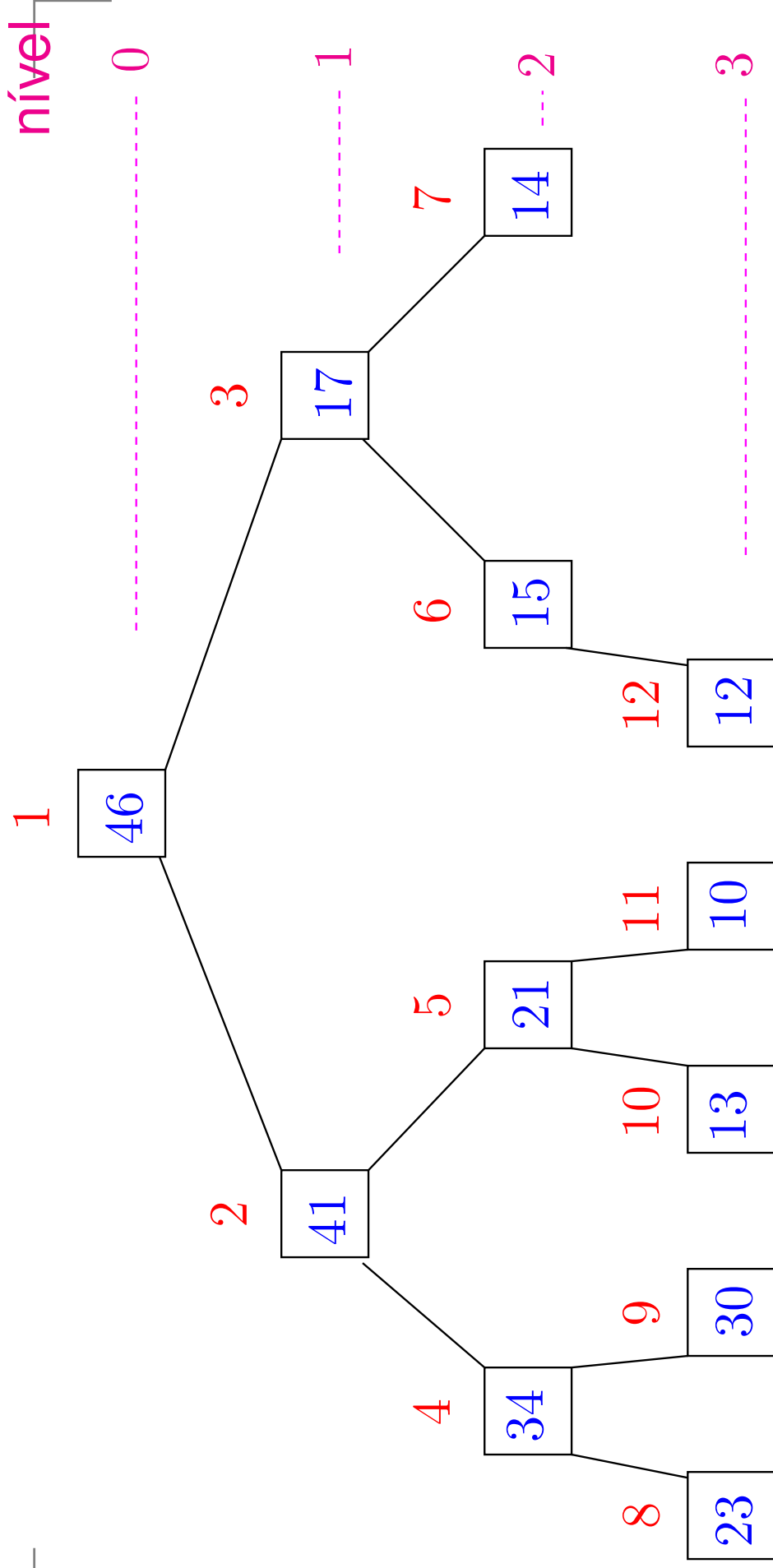
Simulação



1 2 3 4 5 6 7 8 9 10 11 12

46	41	17	34	21	15	14	23	30	13	10	12
----	----	----	----	----	----	----	----	----	----	----	----

Simulação



1 2 3 4 5 6 7 8 9 10 11 12

46	41	17	34	21	15	14	23	30	13	10	12
----	----	----	----	----	----	----	----	----	----	----	----

Consumo de tempo

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

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

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

Consumo de tempo

$T(h)$:= 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

$T(h)$:= 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: $T(h)$ é ???.

Consumo de tempo

$T(h) :=$ 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: $T(h)$ é $O(h)$.

Consumo de tempo

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

Recorrência associada:

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

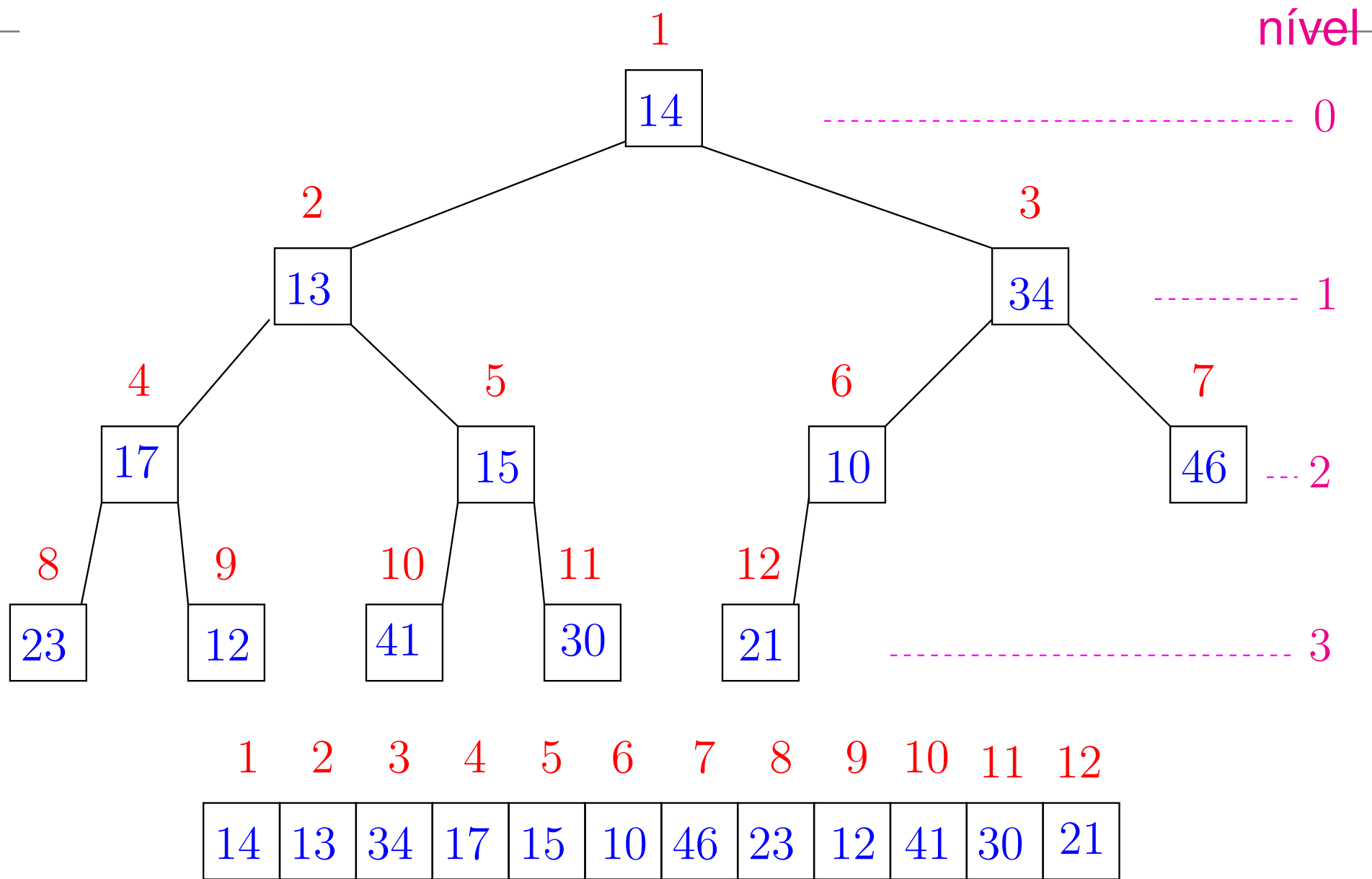
pois altura de *maior* é $h - 1$.

Solução assintótica: $T(h)$ é $O(h)$.

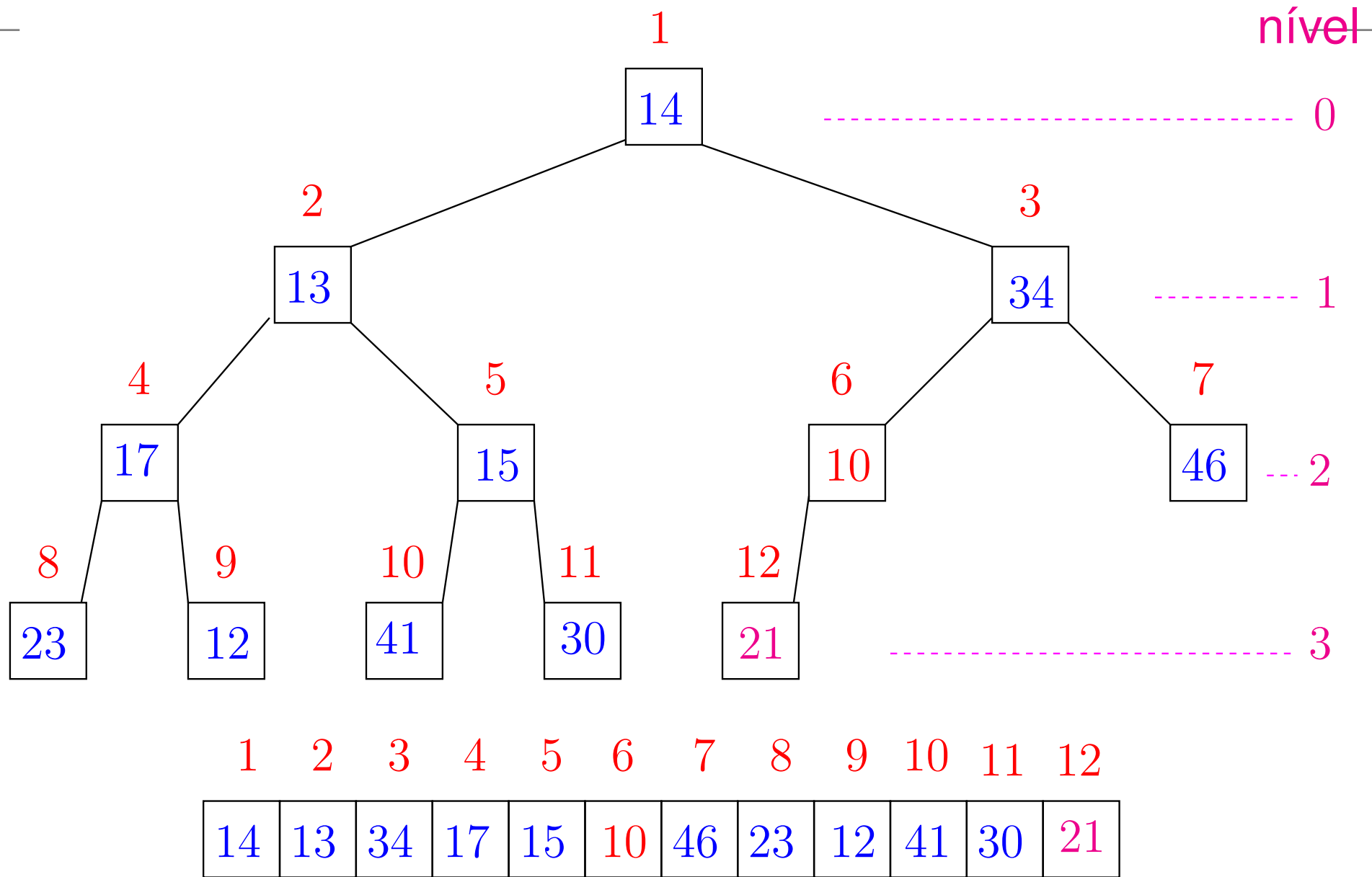
Como $h \leq \lg m$, podemos dizer que:

O consumo de tempo do algoritmo **DESCE-HEAP** é $O(\lg m)$ (ou melhor ainda, $O(h)$).

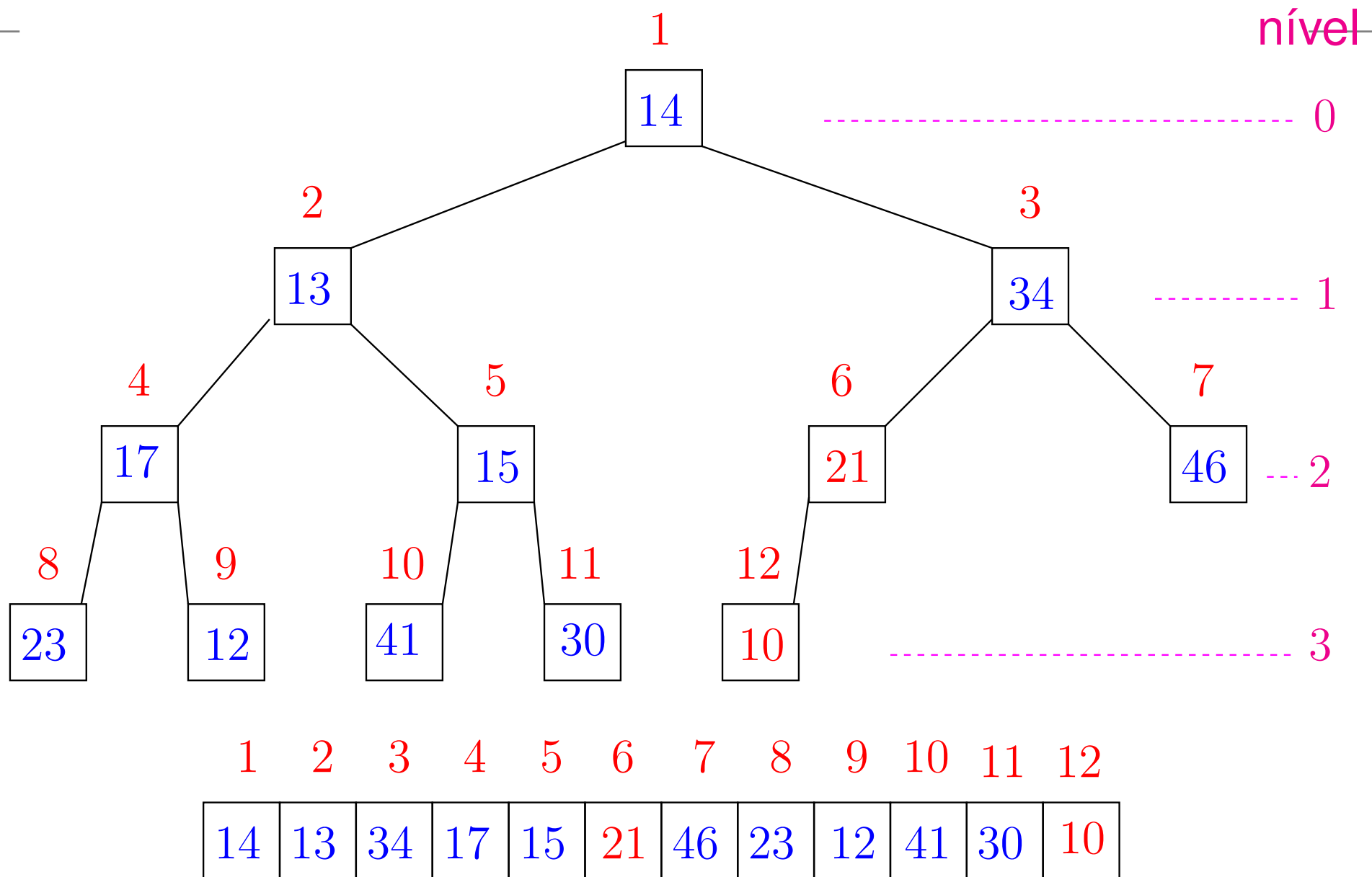
Construção de um heap



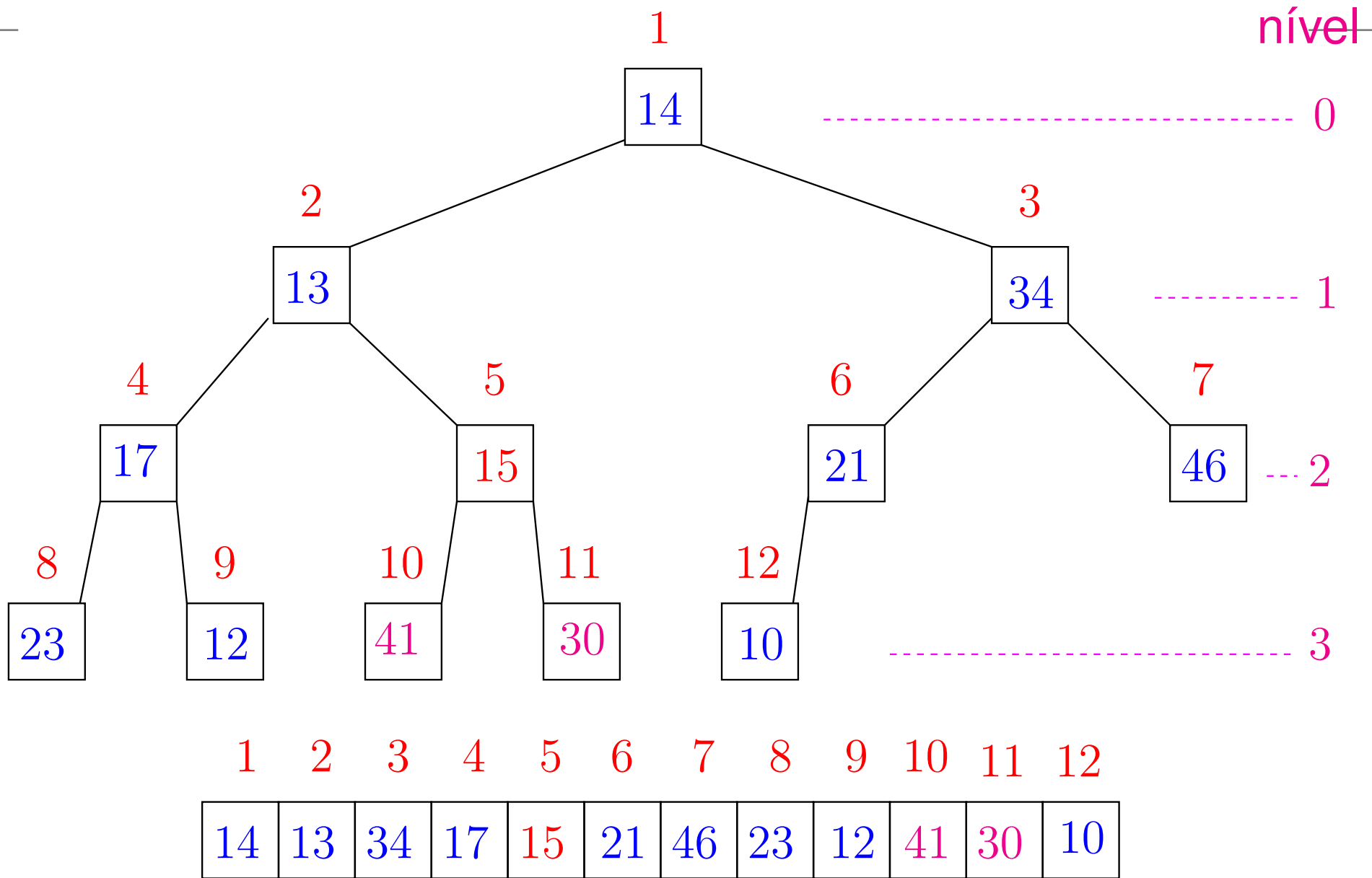
Construção de um heap



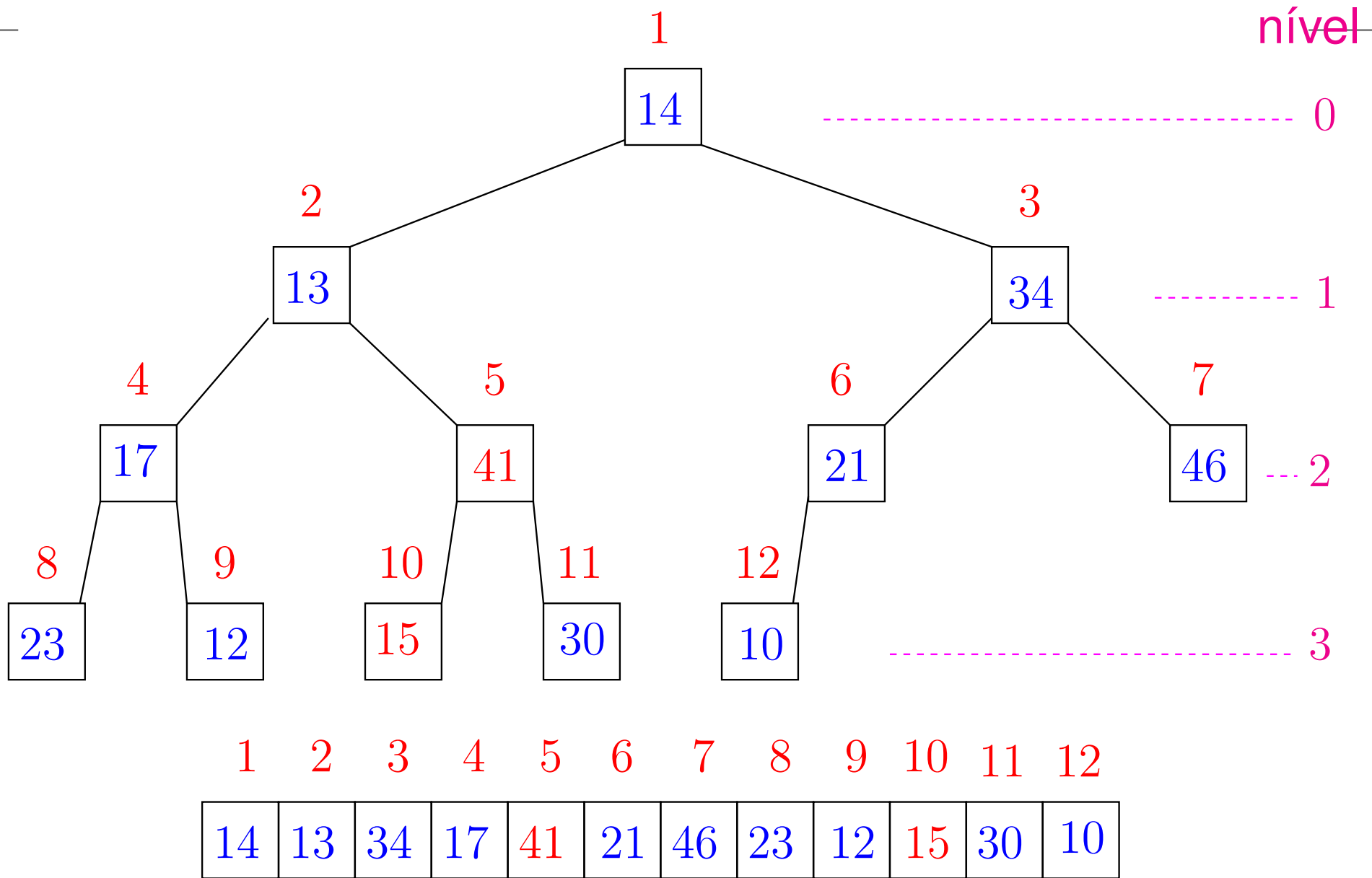
Construção de um heap



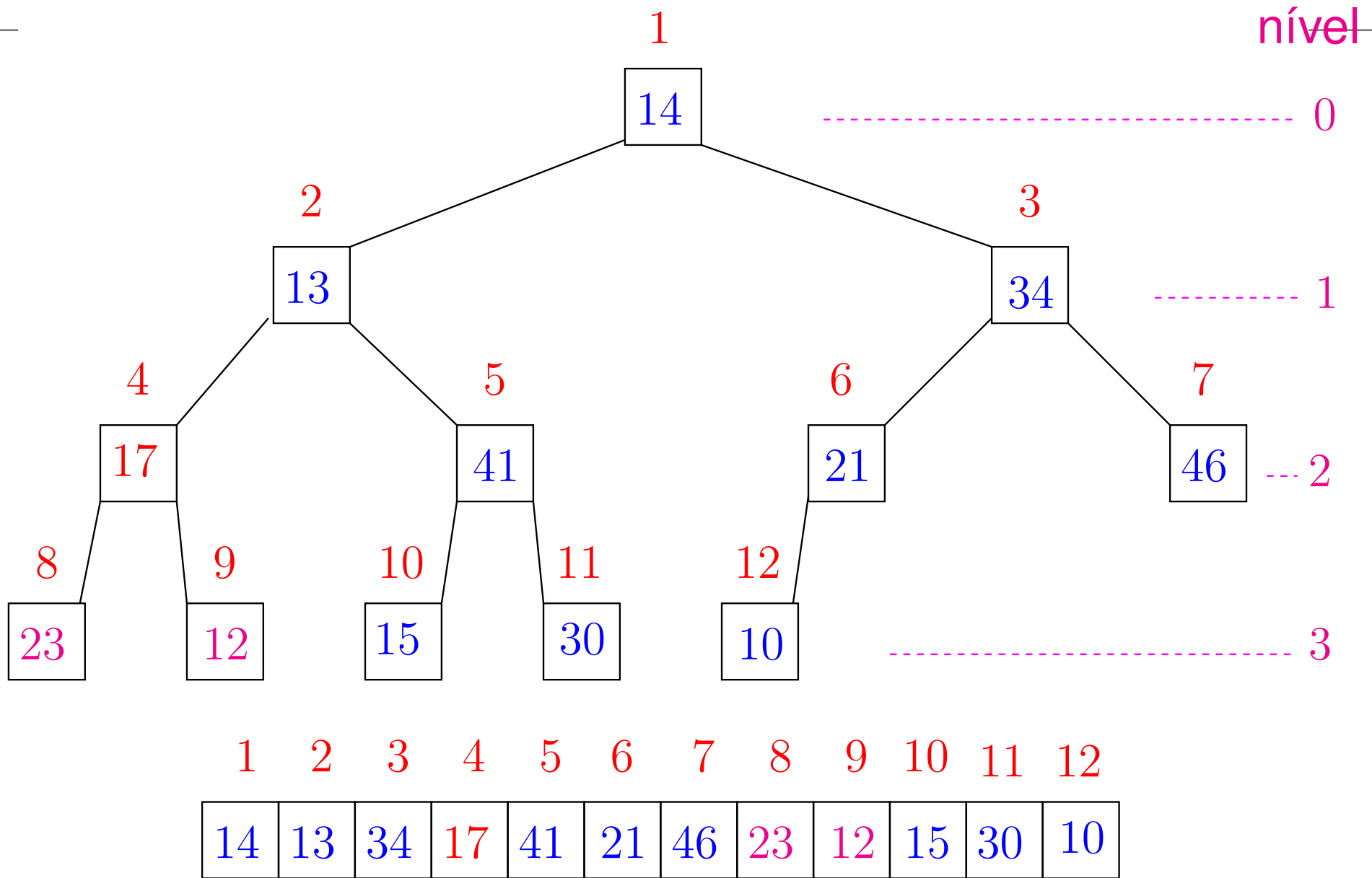
Construção de um heap



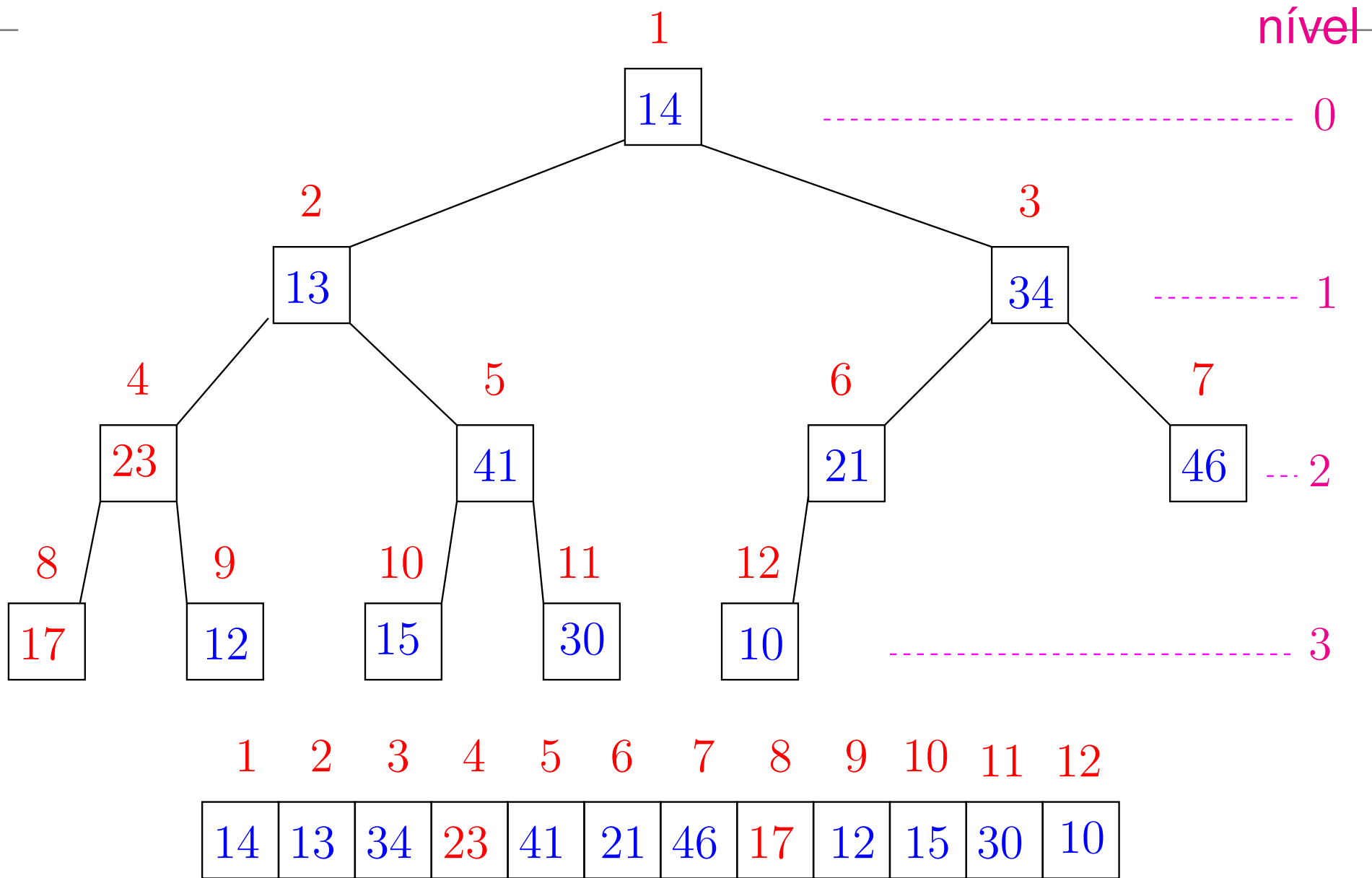
Construção de um heap



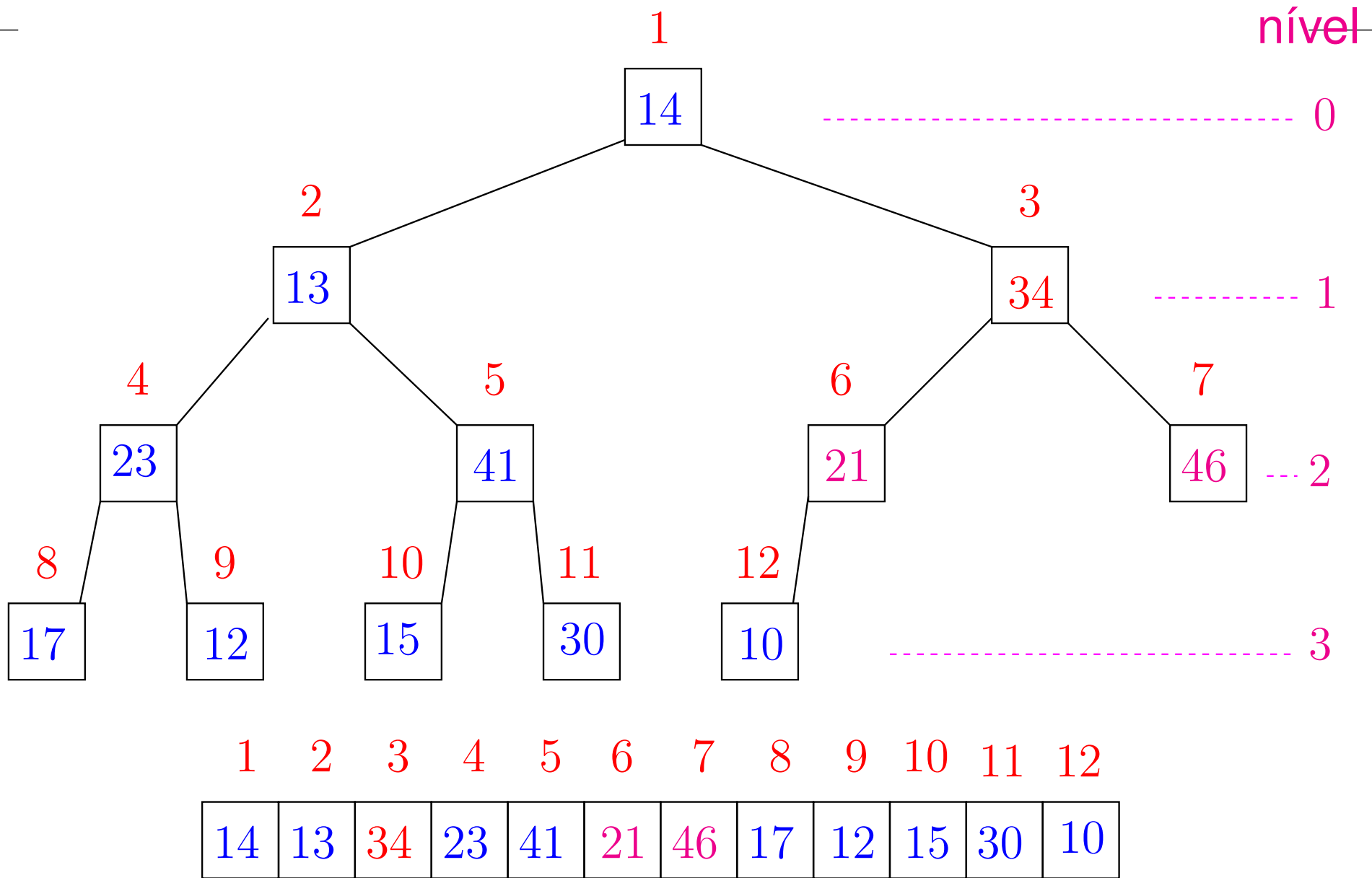
Construção de um heap



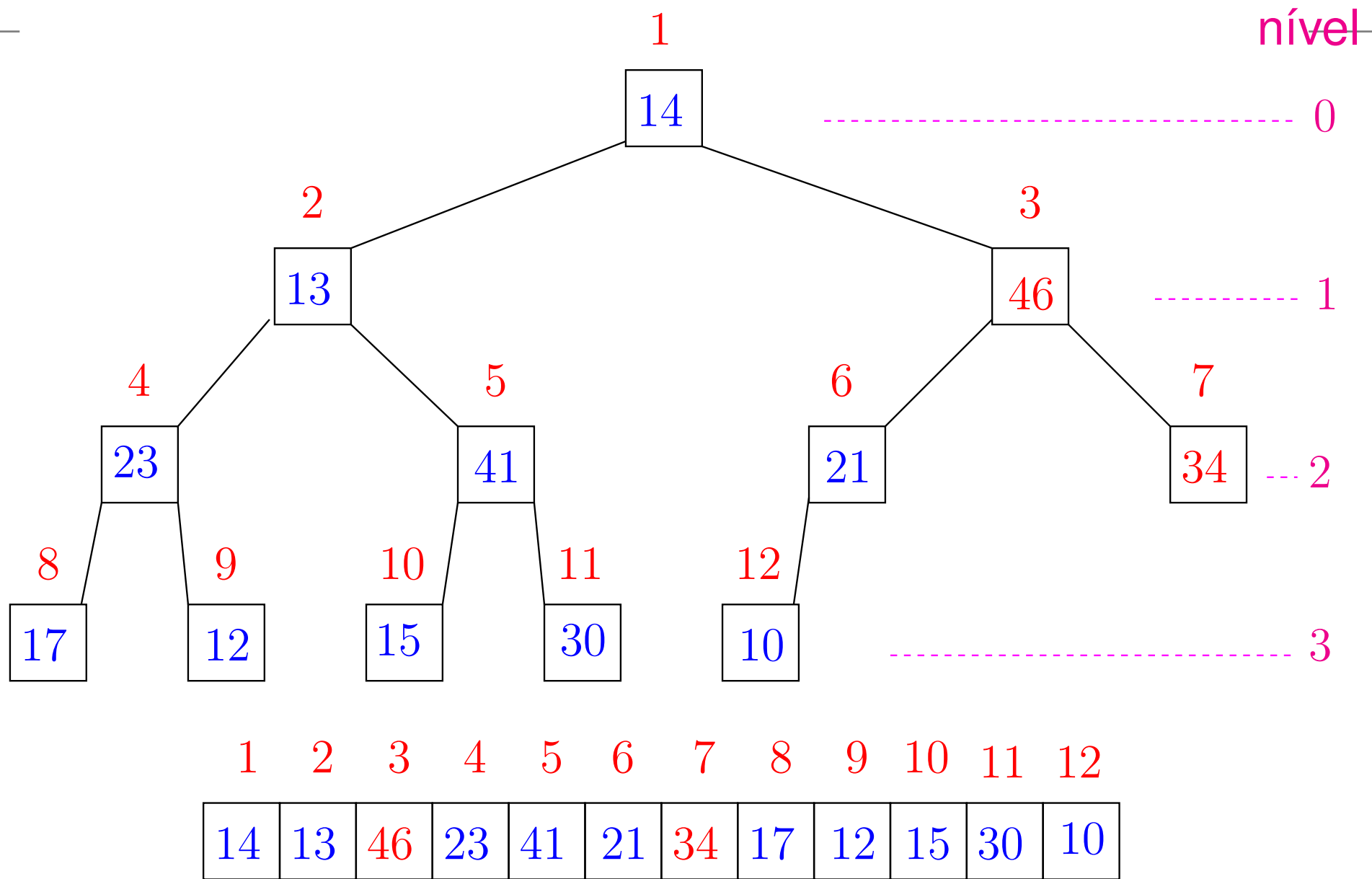
Construção de um heap



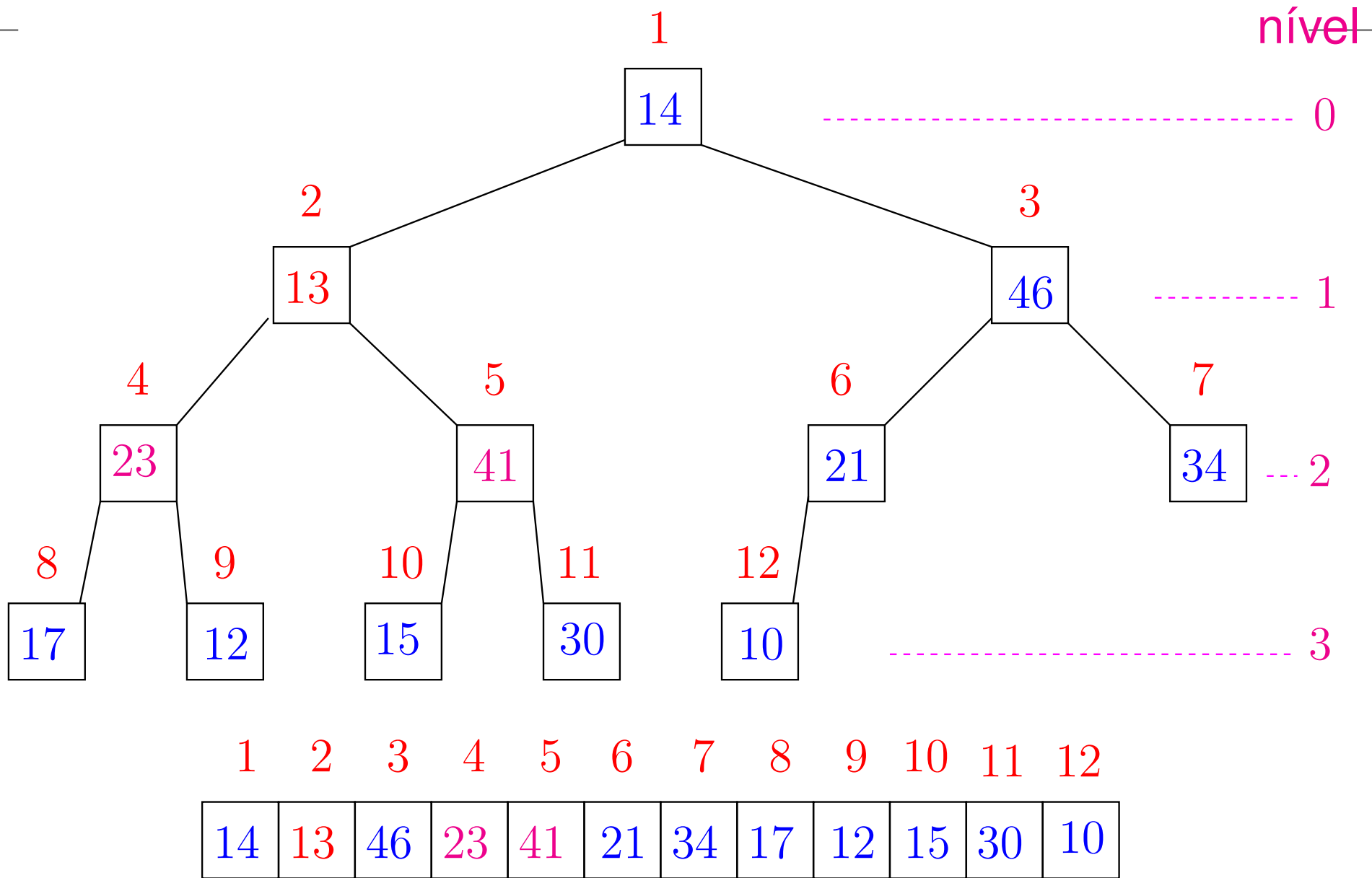
Construção de um heap



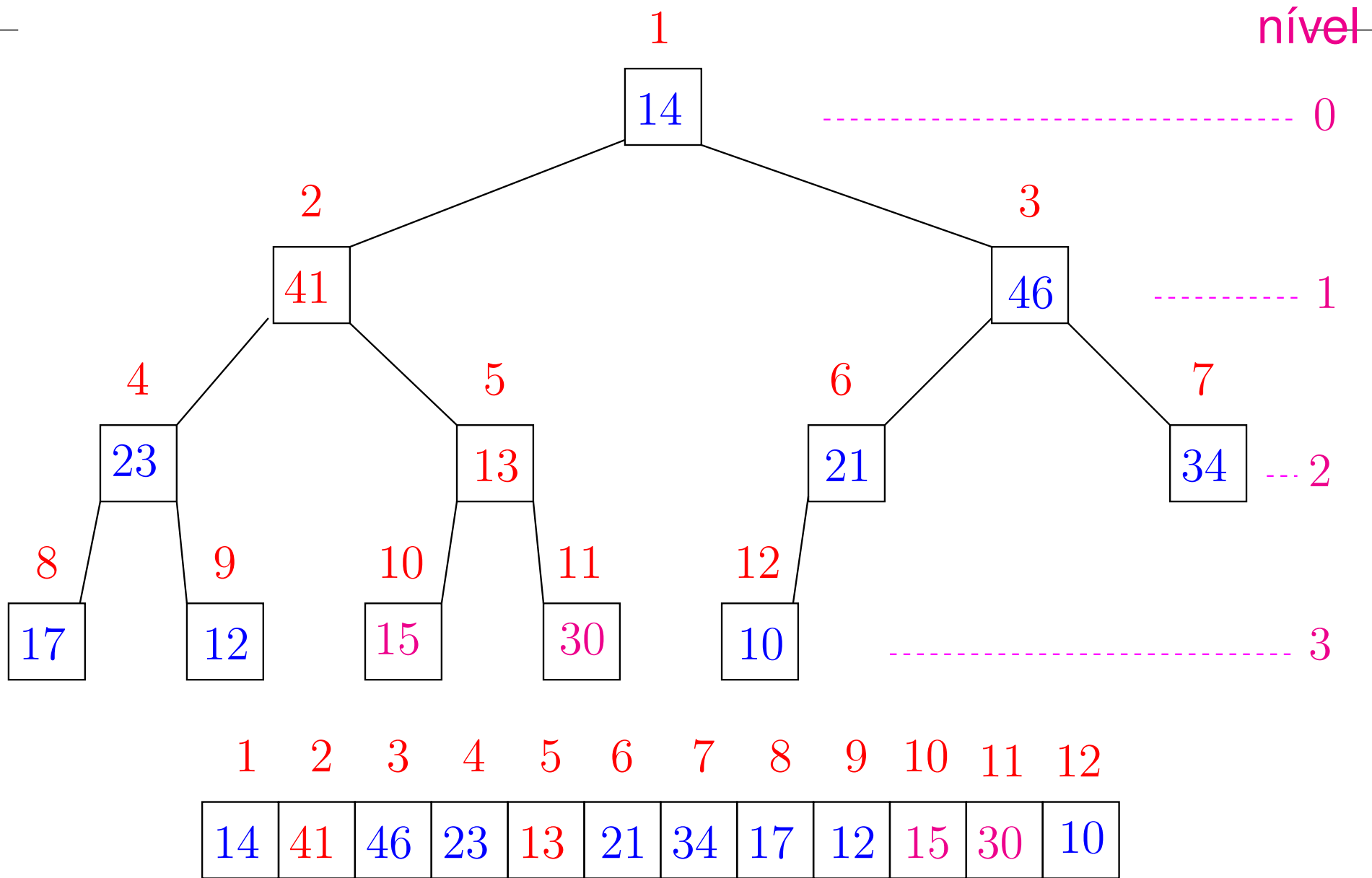
Construção de um heap



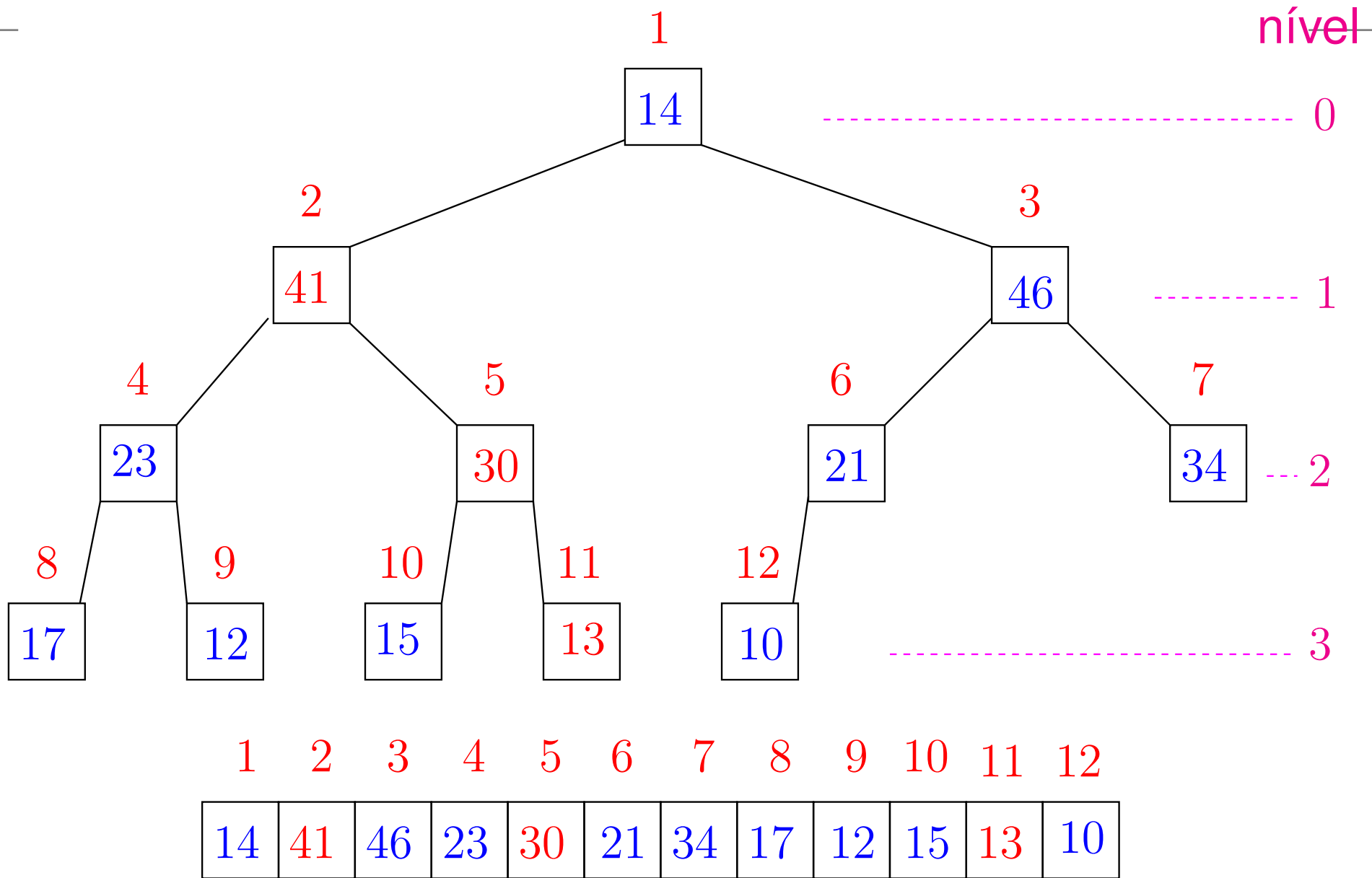
Construção de um heap



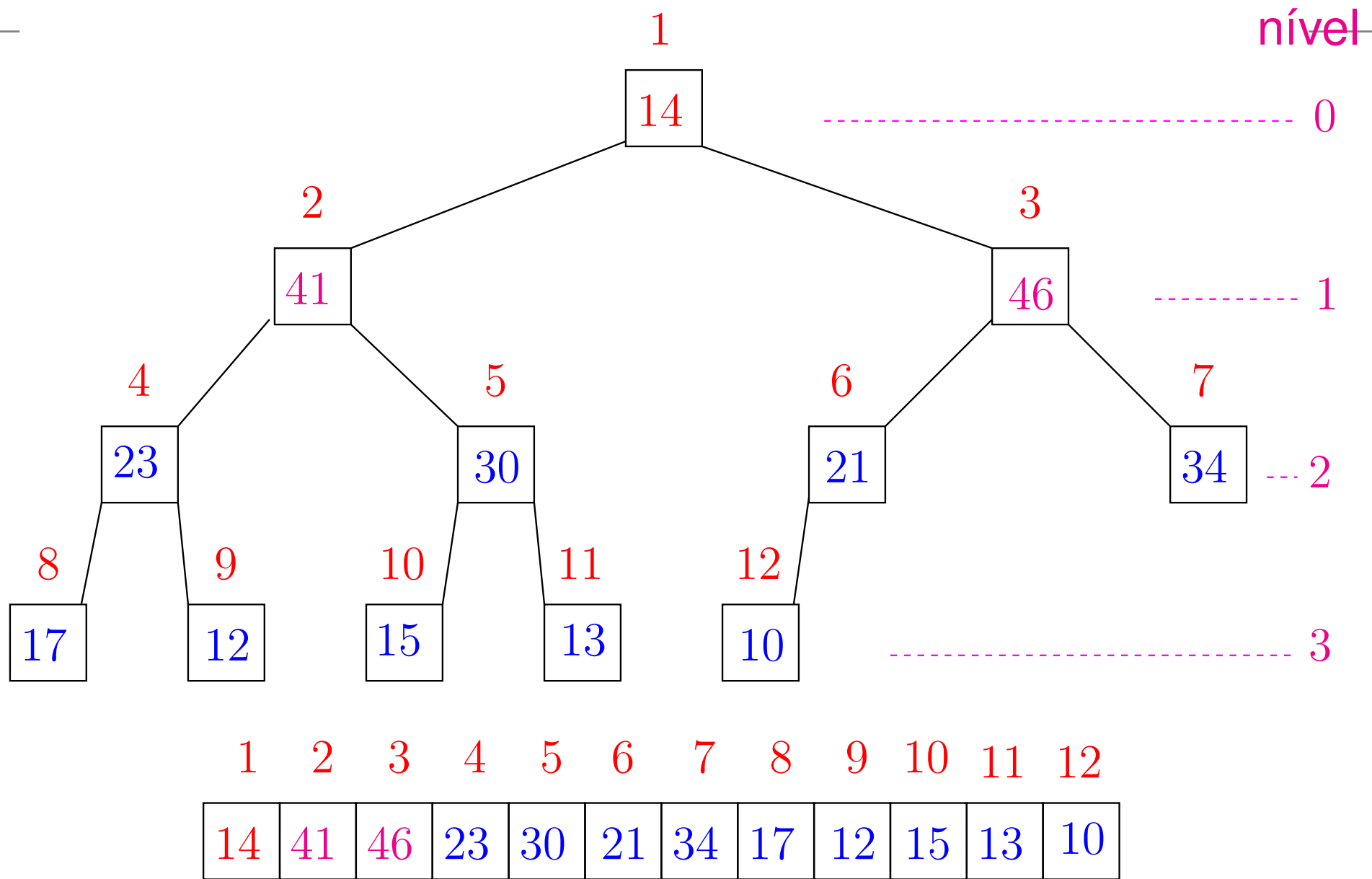
Construção de um heap



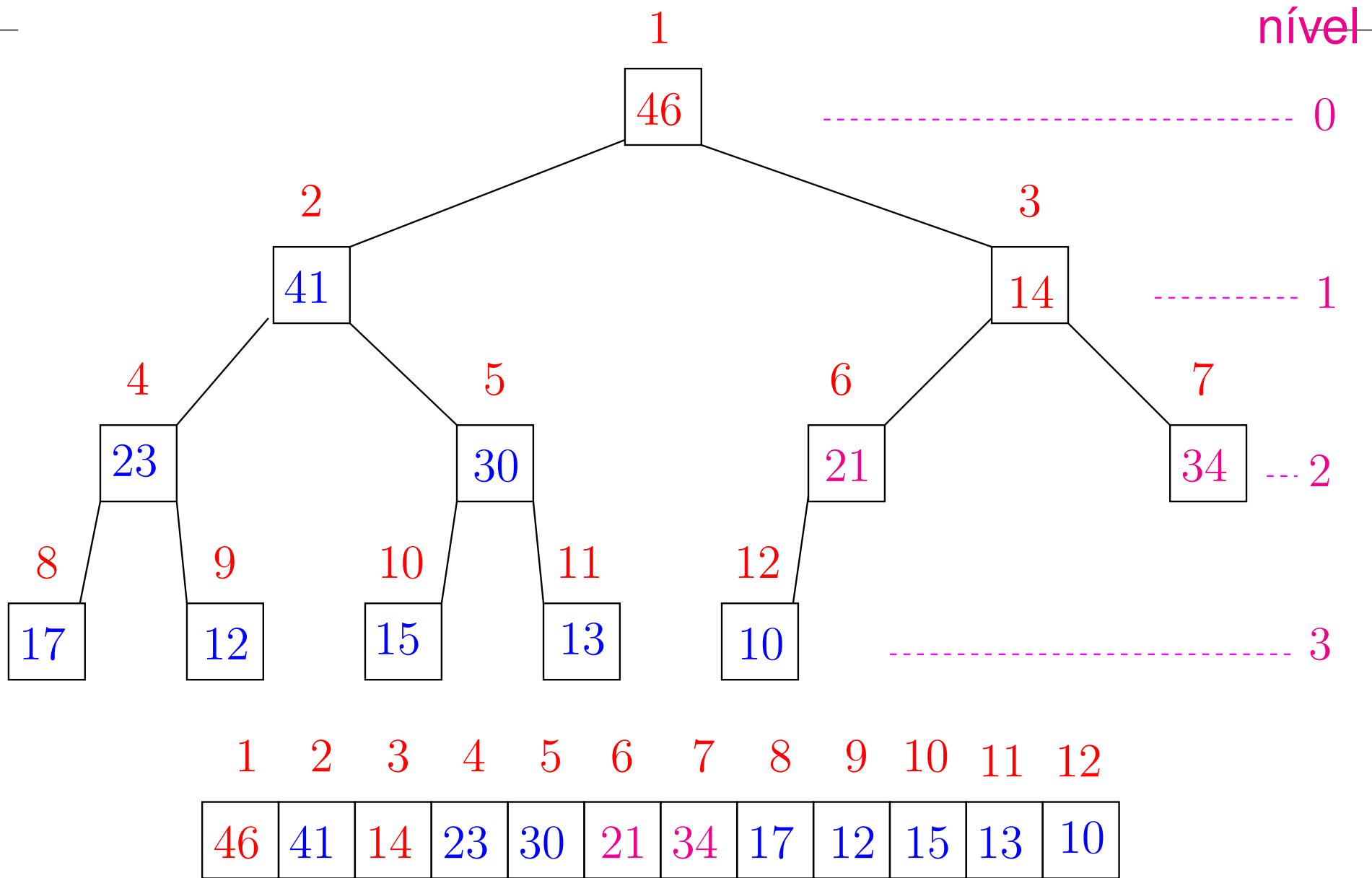
Construção de um heap



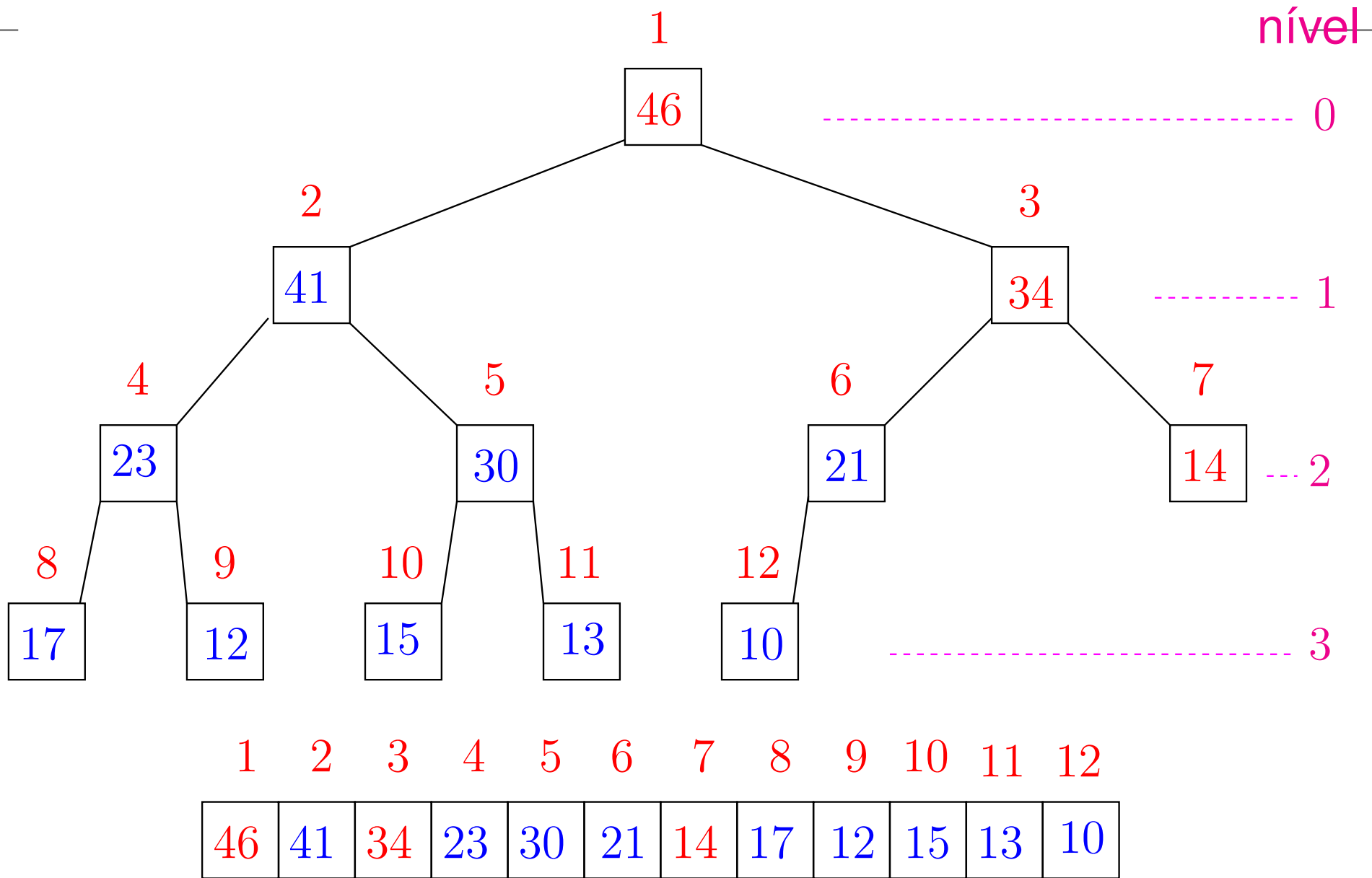
Construção de um heap



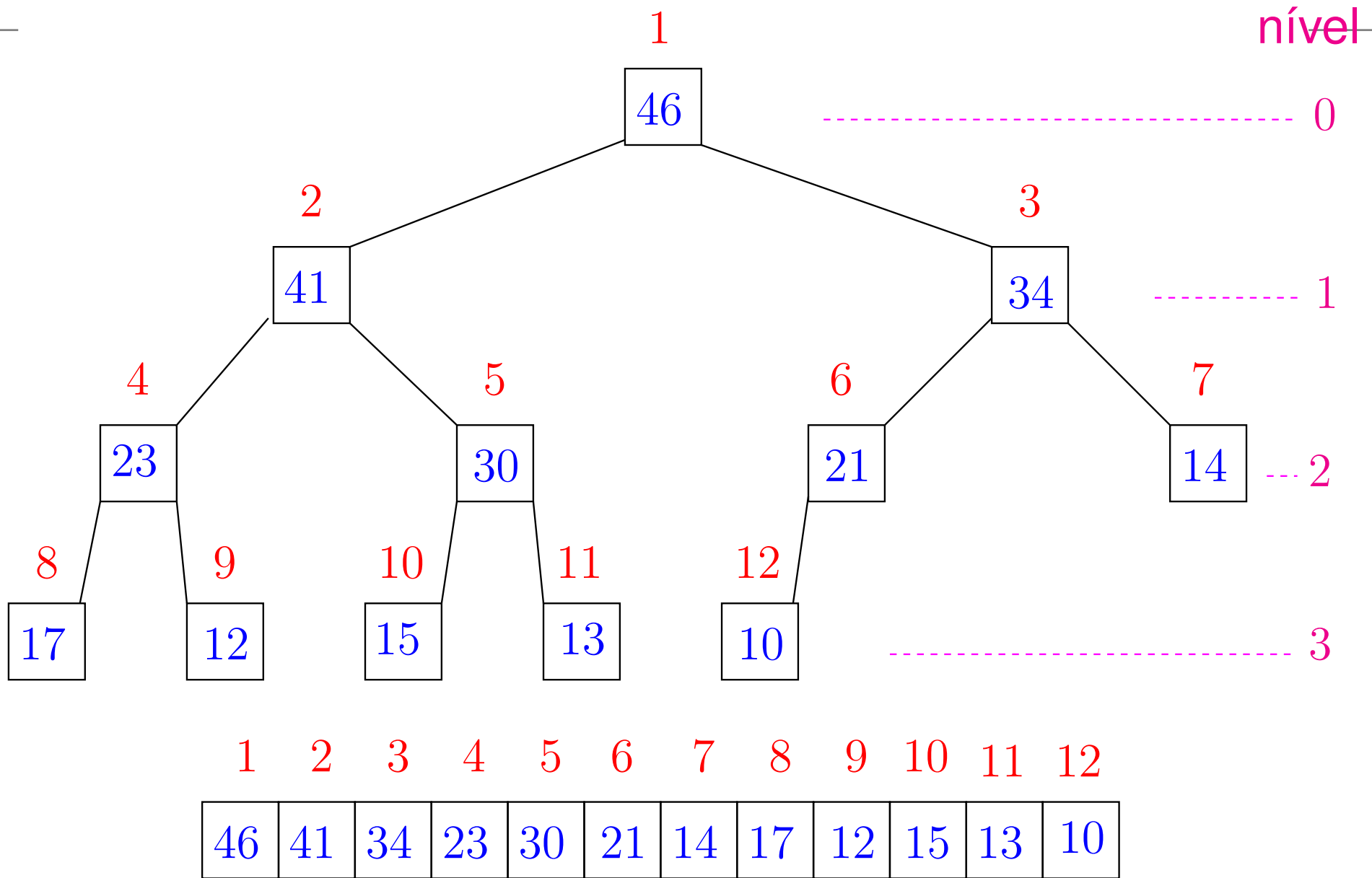
Construção de um heap



Construção de um heap



Construção de um heap



Construção de um heap

Recebe um vetor $A[1 \dots n]$ e **rearranja** A para que seja heap.

CONSTRÓI-HEAP (A, n)

2 **para** $i \leftarrow \lfloor n/2 \rfloor$ **decrecendo até** 1 **faça**

3 **DESCE-HEAP** (A, n, i)

Relação invariante:

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

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

Construção de um heap

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

CONSTRÓI-HEAP (A, n)

2 **para** $i \leftarrow \lfloor n/2 \rfloor$ **decrecendo até** 1 **faça**

3 **DESCE-HEAP** (A, n, i)

Relação invariante:

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

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

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

Análise mais cuidadosa: $T(n)$ é **????**.

$$T(n) \text{ é } O(n)$$

Prova:

Consumo de **DESCE-HEAP** (A, n, i) é proporcional a h , logo

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

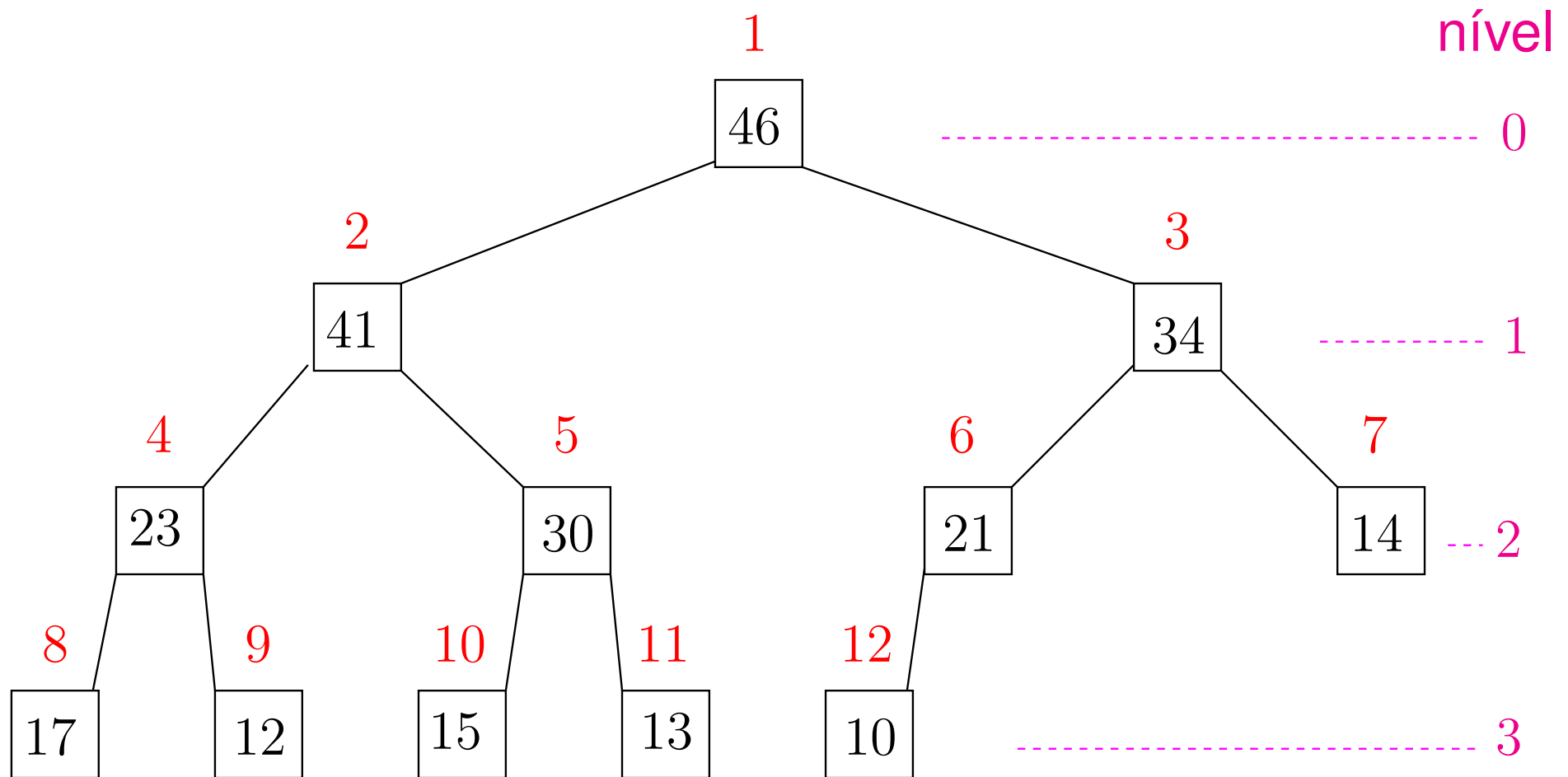
$$T(n) \text{ é } O(n)$$

Prova:

O consumo de tempo de **DESCE-HEAP** (A, n, i) é $O(h)$, onde h é a altura da árvore de raiz i . Logo,

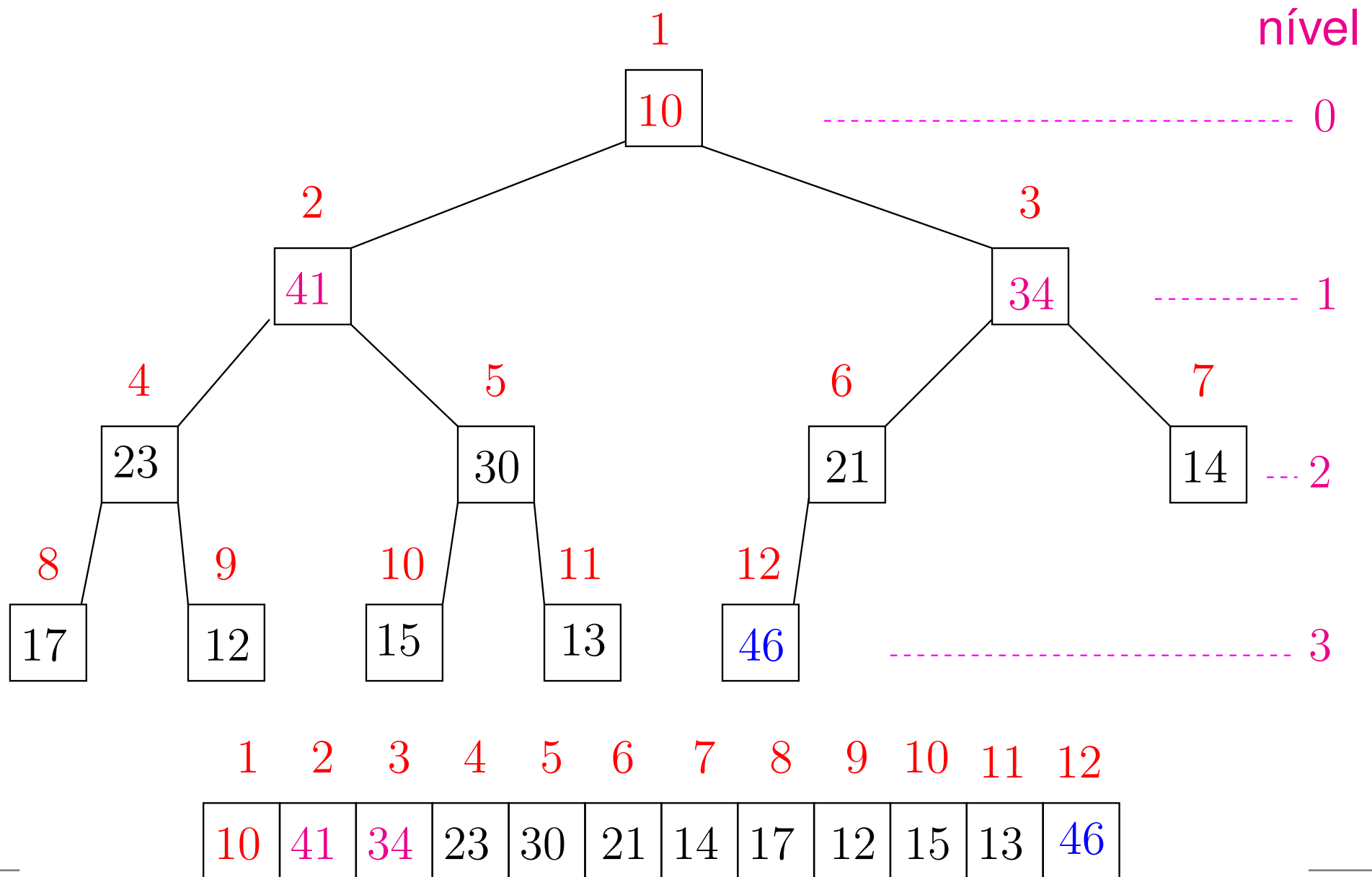
$$\begin{aligned} T(n) &= \sum_{h=0}^{\lfloor \lg n \rfloor} 2^{\lfloor \lg n \rfloor - h} O(h) \\ &= O\left(n \sum_{h=0}^{\lfloor \lg n \rfloor} \frac{h}{2^h}\right) \\ &= O\left(n \sum_{h=0}^{\infty} \frac{h}{2^h}\right) \\ &= O\left(n \frac{1/2}{(1 - 1/2)^2}\right) = O(2n) = O(n). \end{aligned}$$

Heap sort

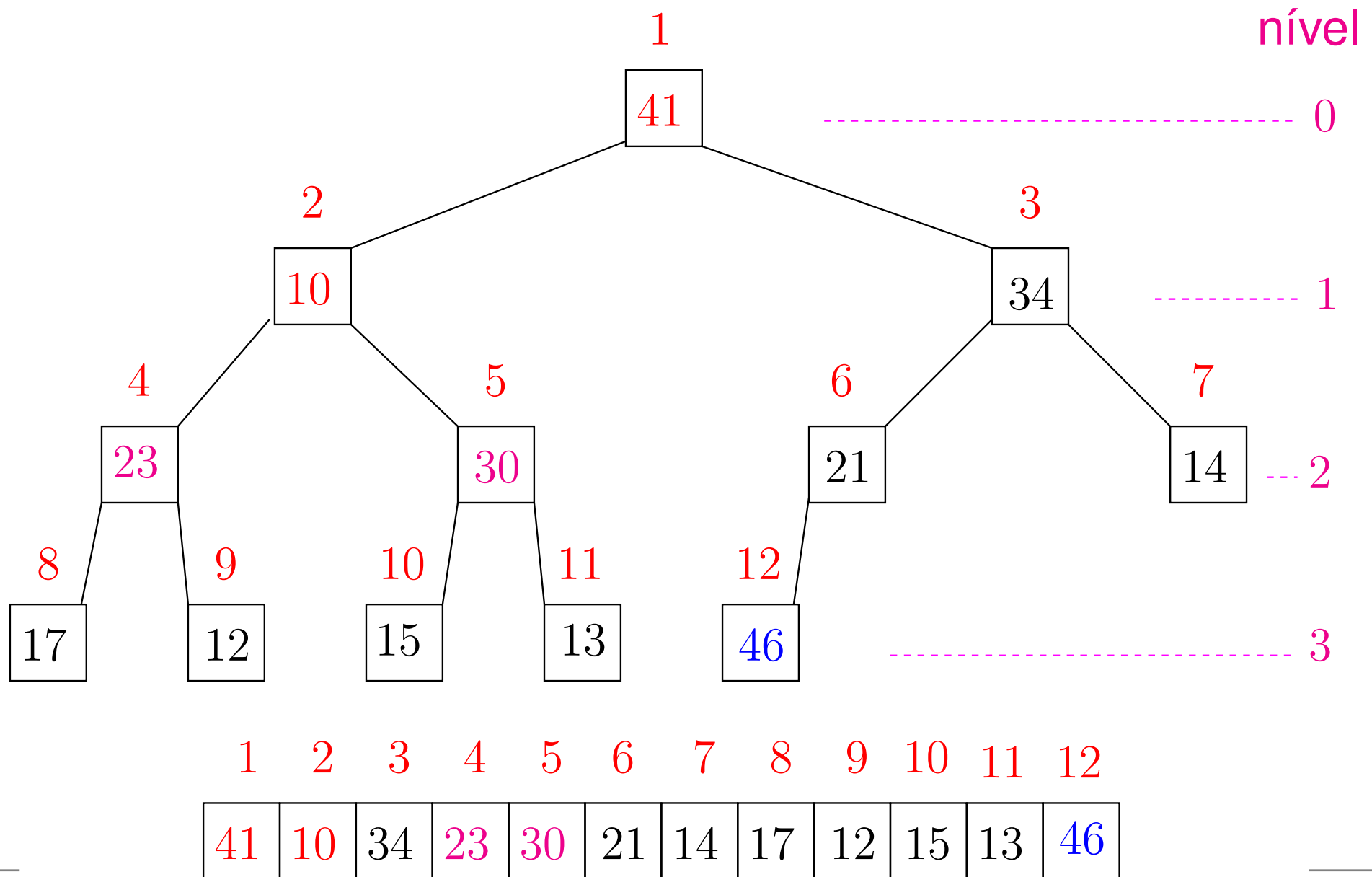


1	2	3	4	5	6	7	8	9	10	11	12
46	41	34	23	30	21	14	17	12	15	13	10

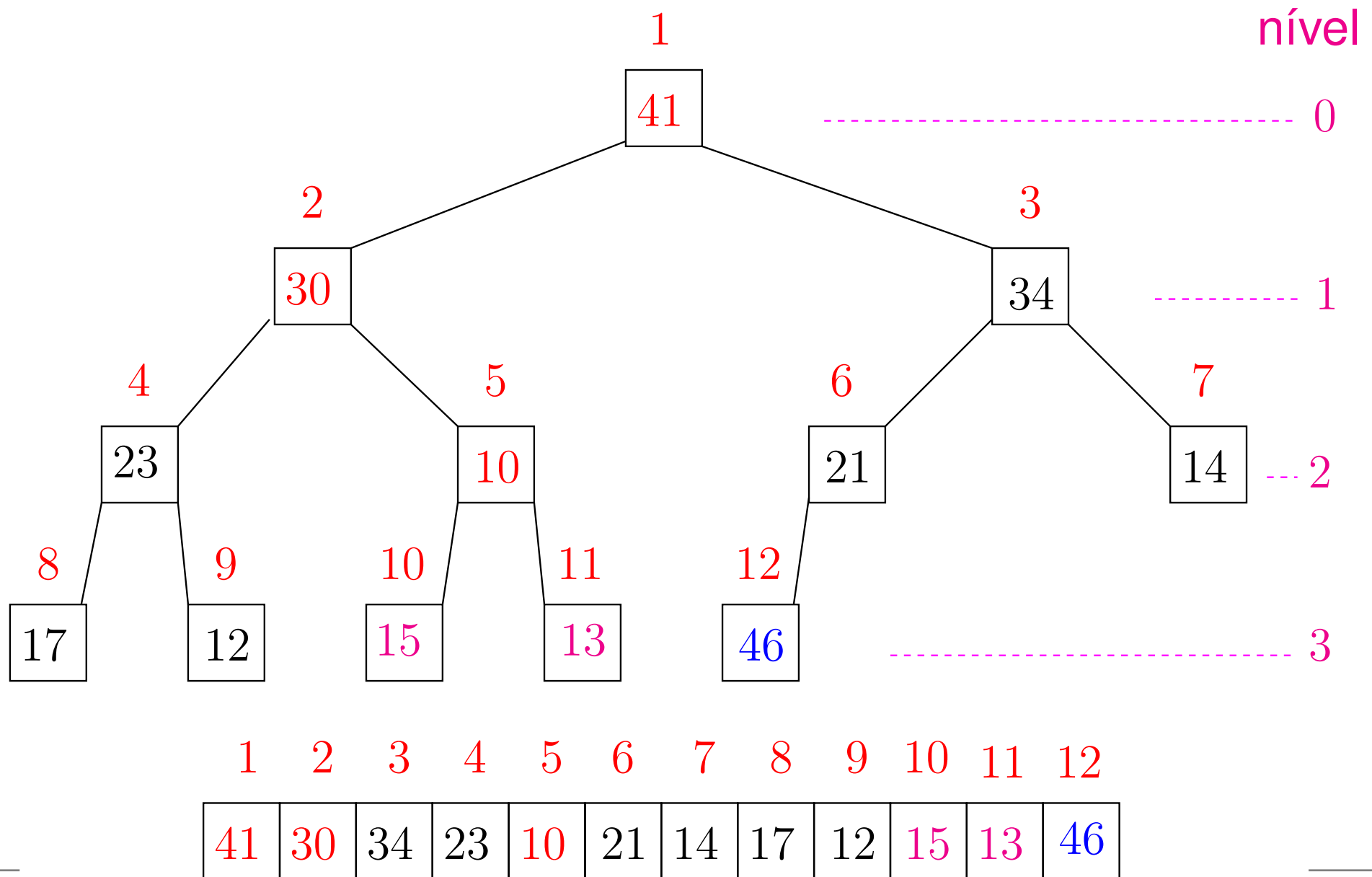
Heap sort



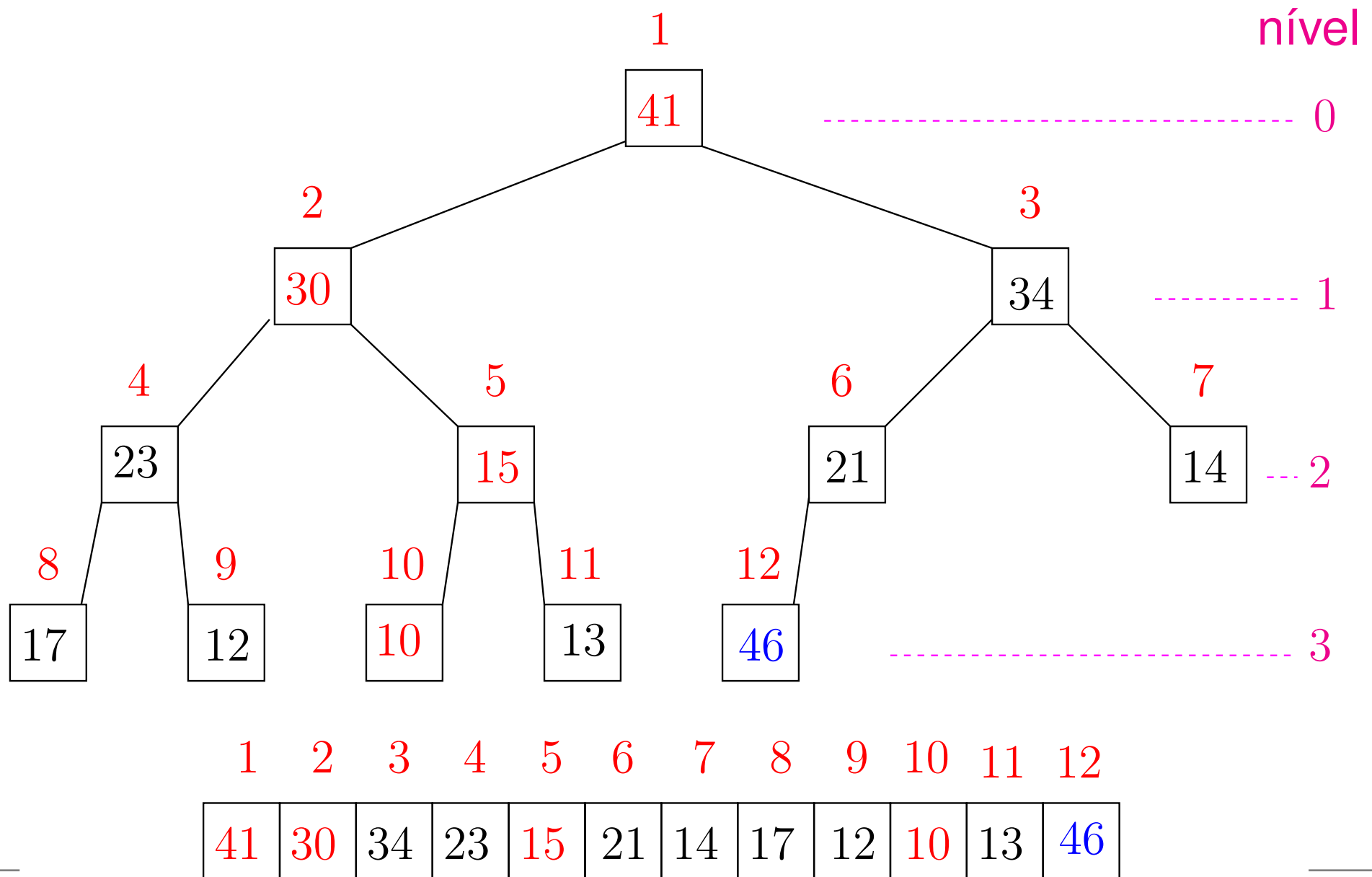
Heap sort



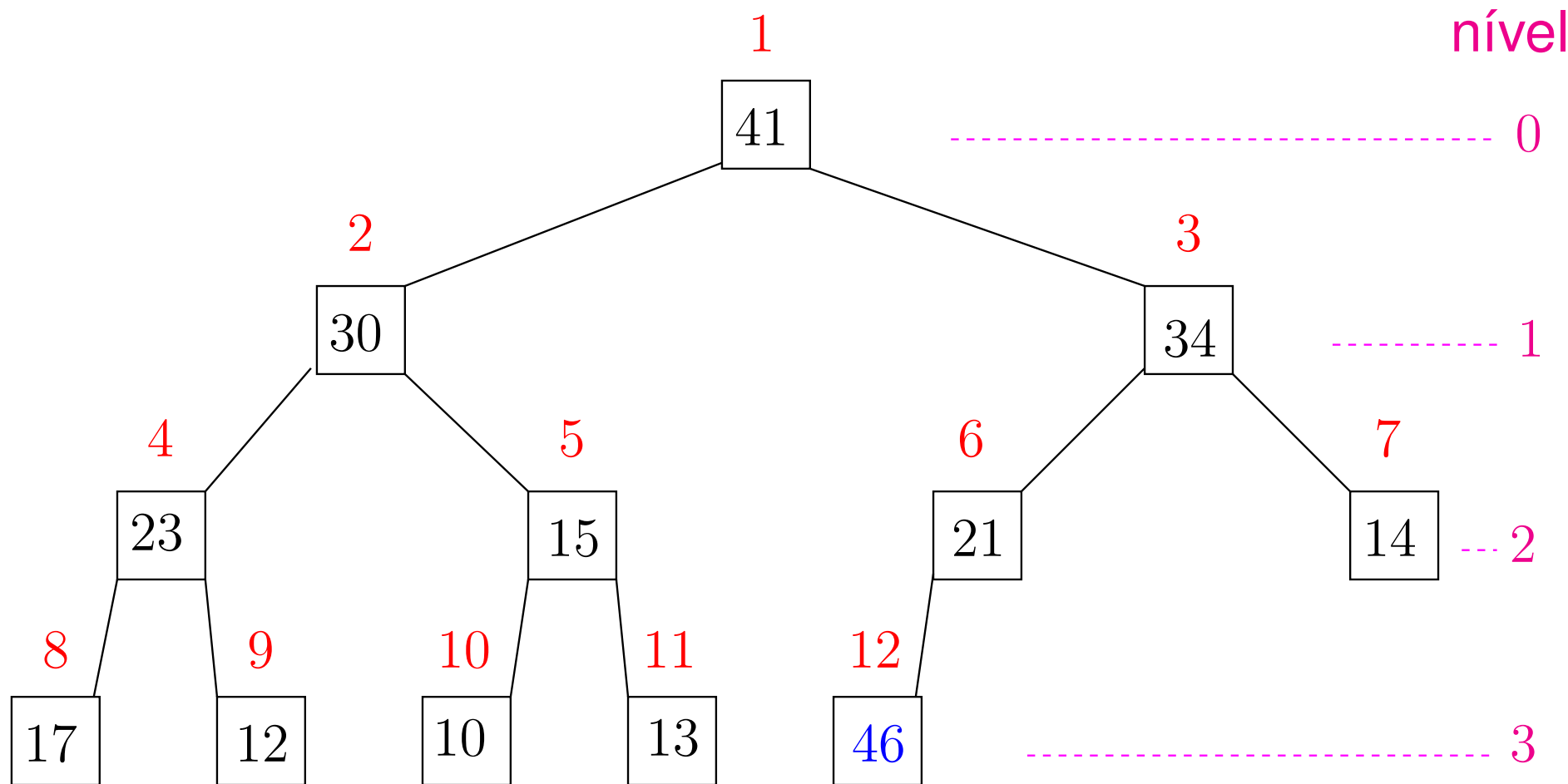
Heap sort



Heap sort

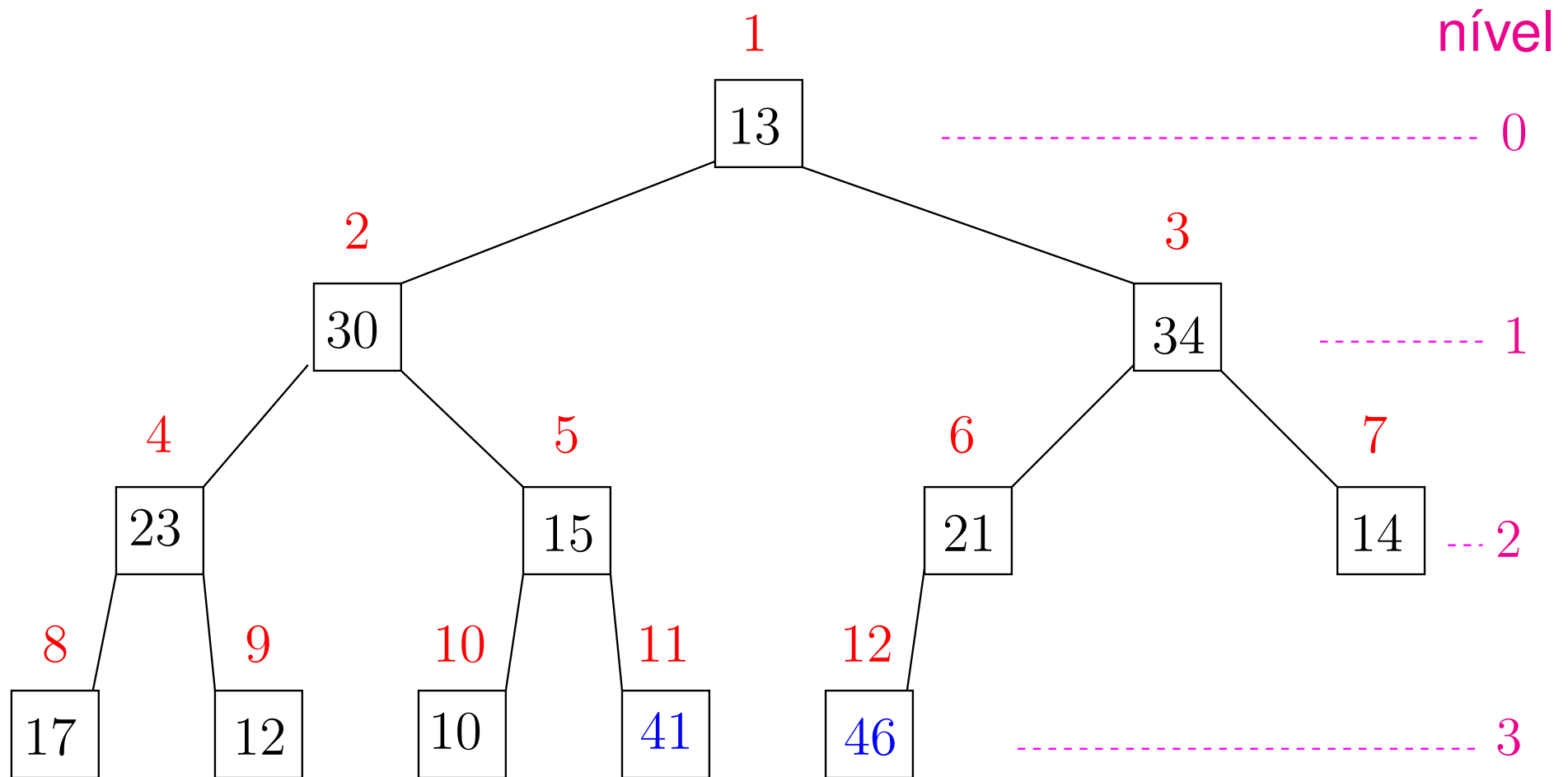


Heap sort



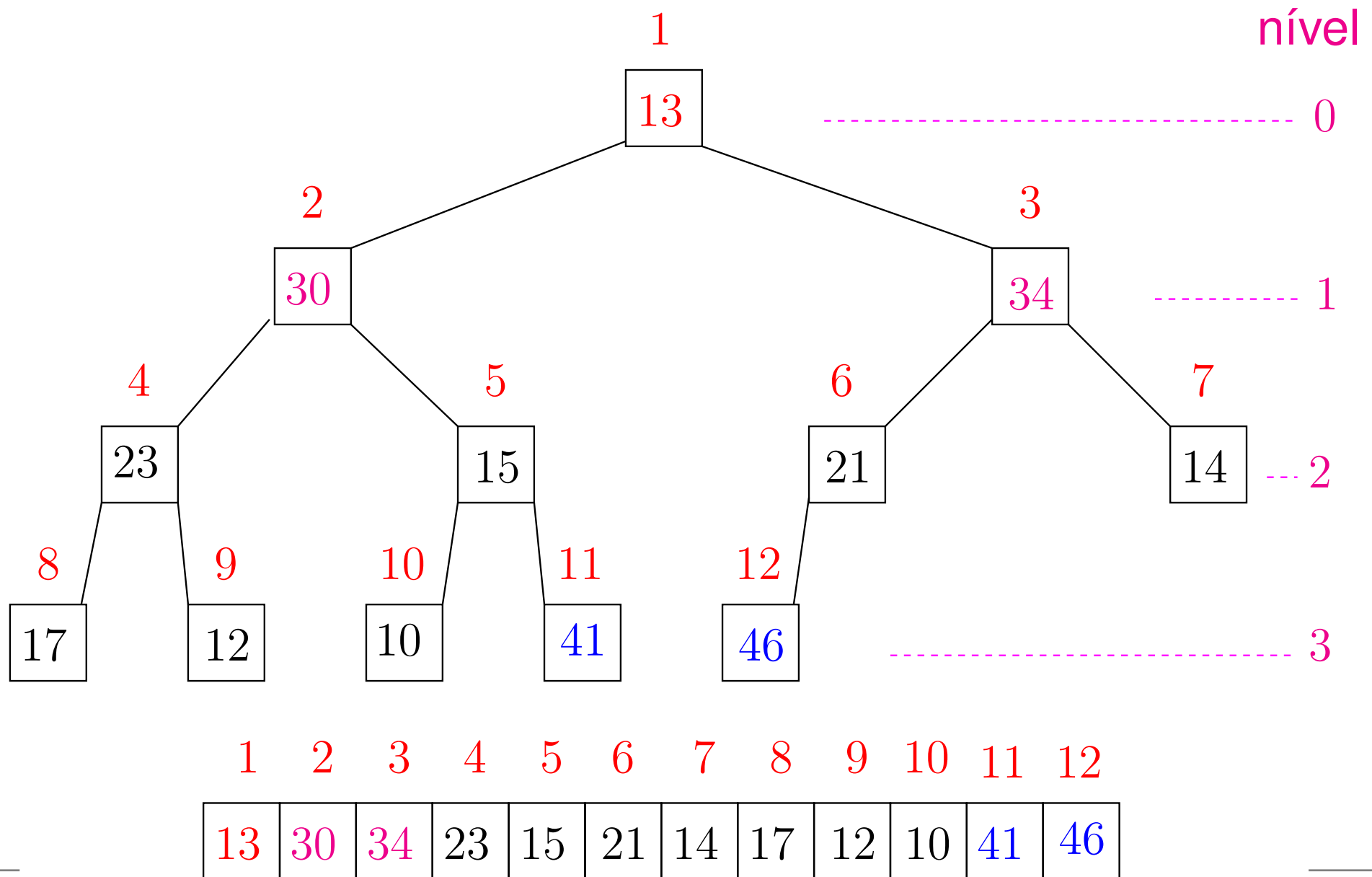
1	2	3	4	5	6	7	8	9	10	11	12
41	30	34	23	15	21	14	17	12	10	13	46

Heap sort

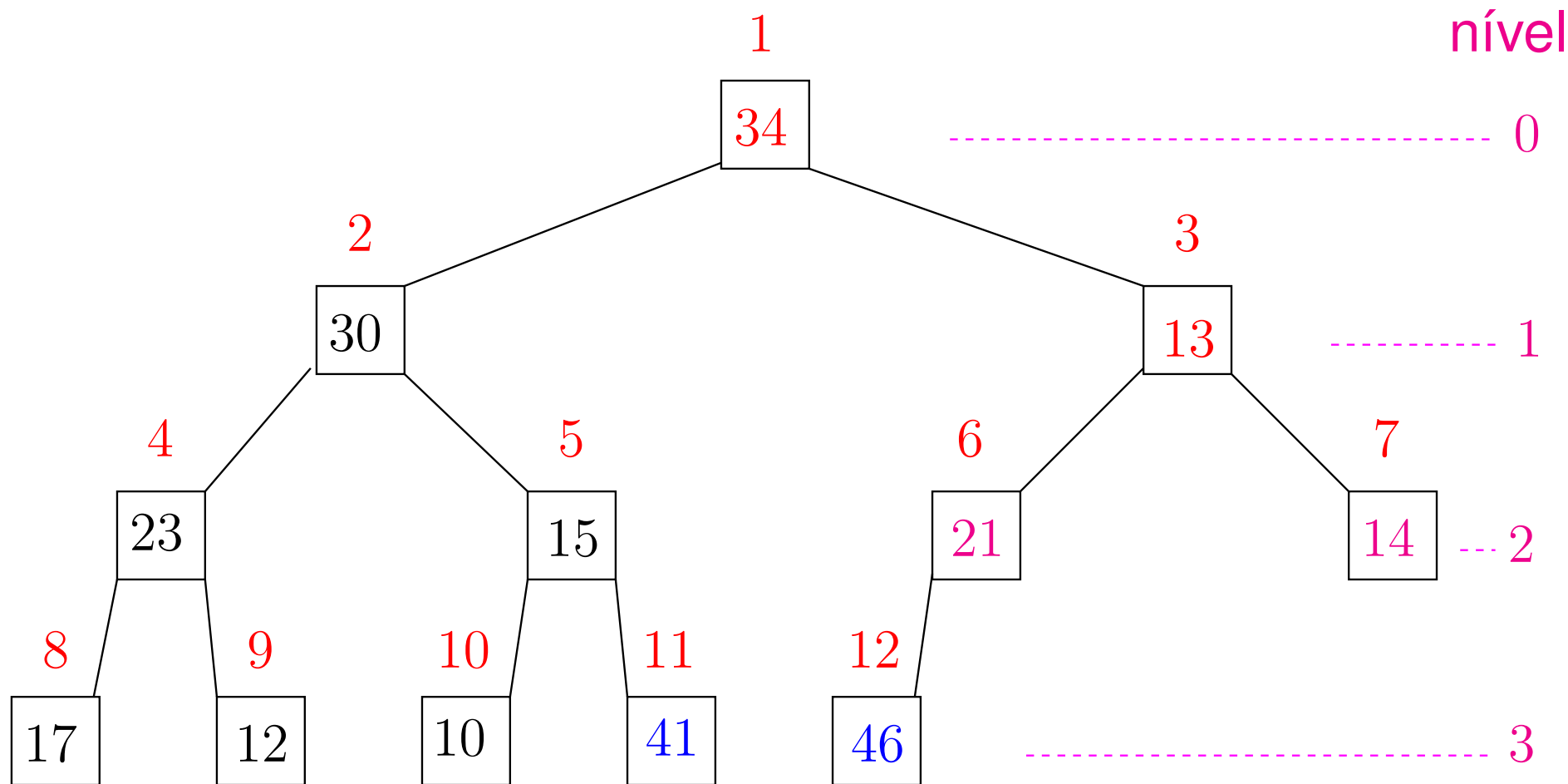


1	2	3	4	5	6	7	8	9	10	11	12
13	30	34	23	15	21	14	17	12	10	41	46

Heap sort

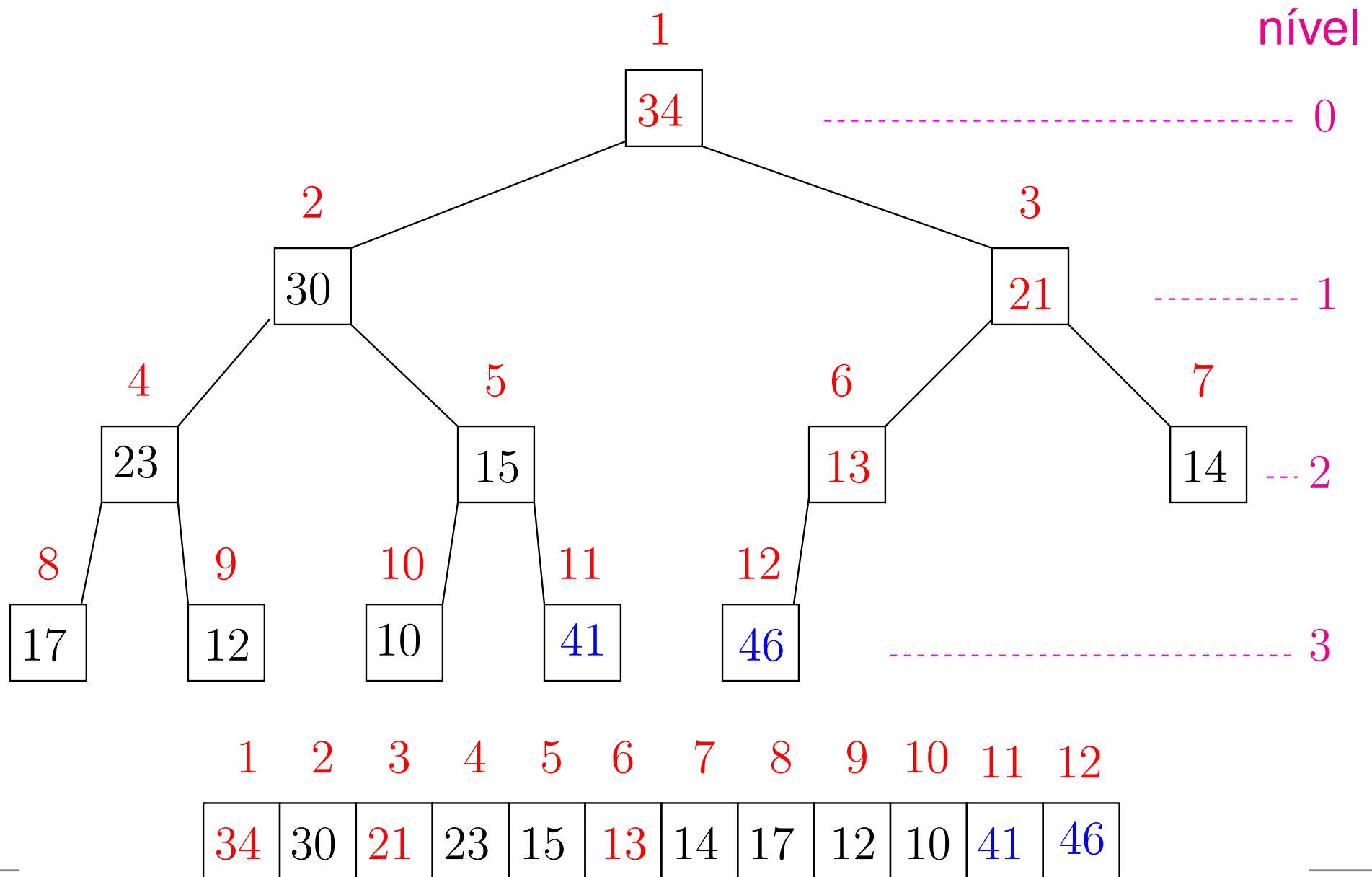


Heap sort

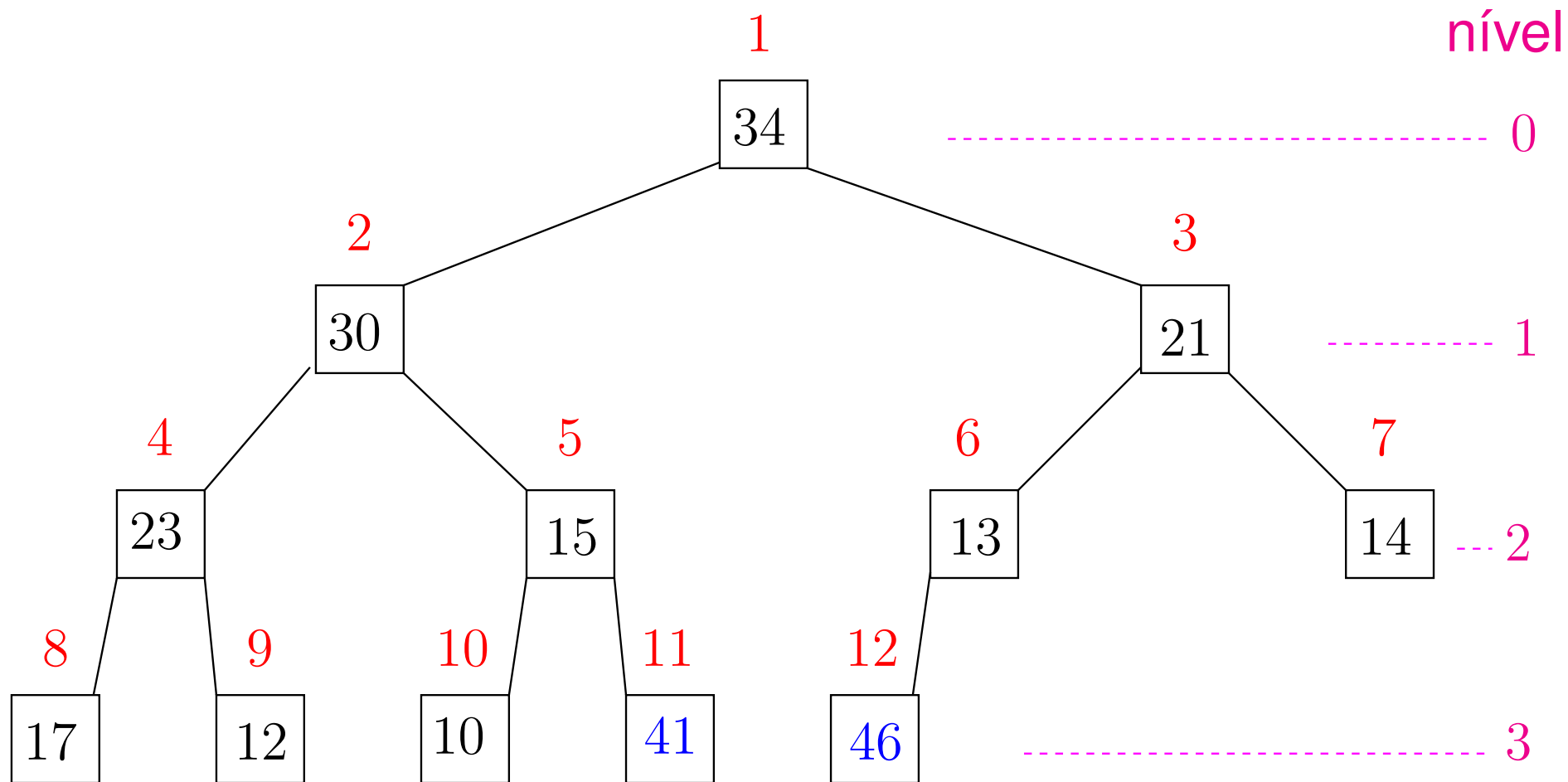


1	2	3	4	5	6	7	8	9	10	11	12
34	30	13	23	15	21	14	17	12	10	41	46

Heap sort

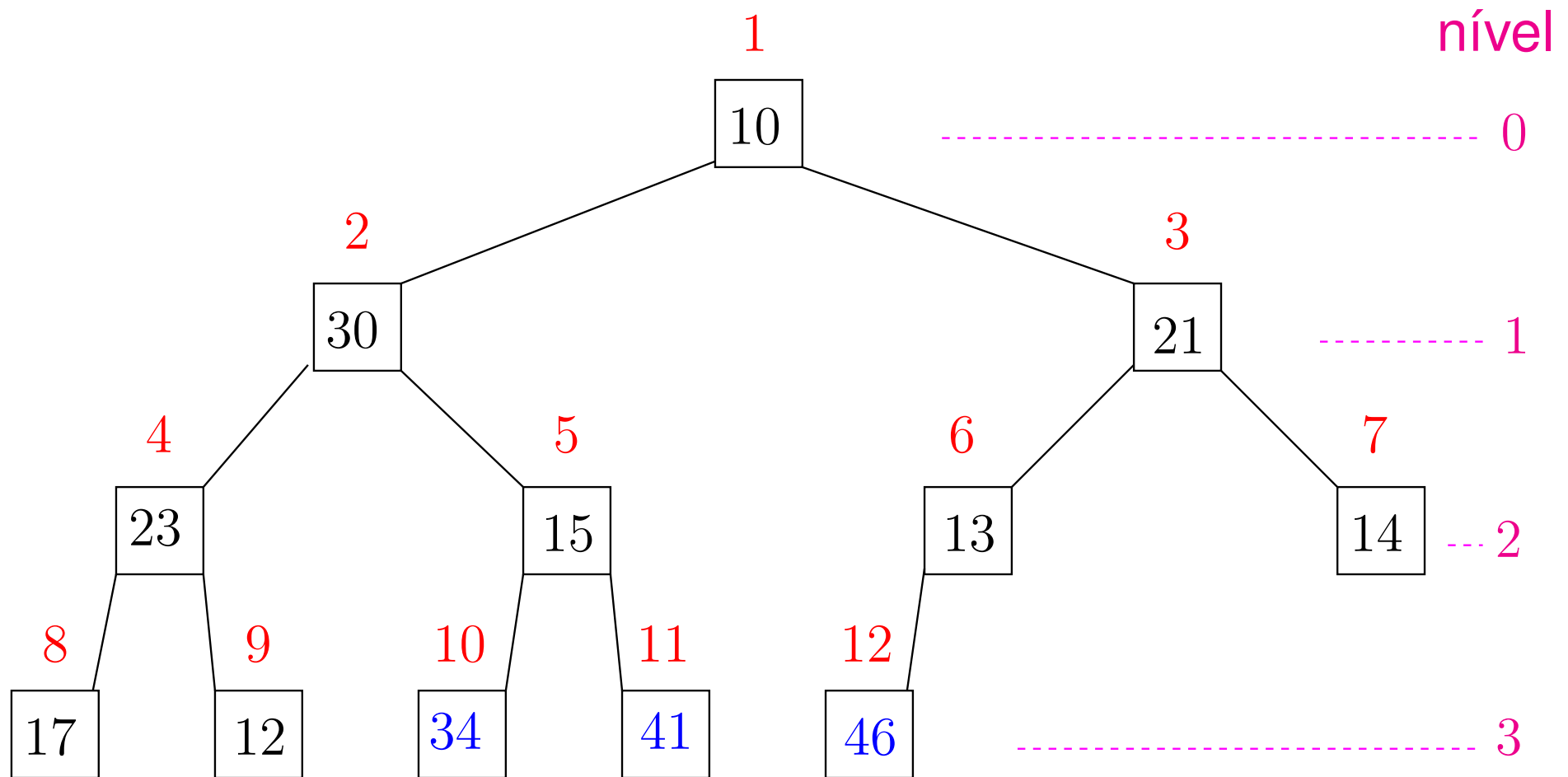


Heap sort



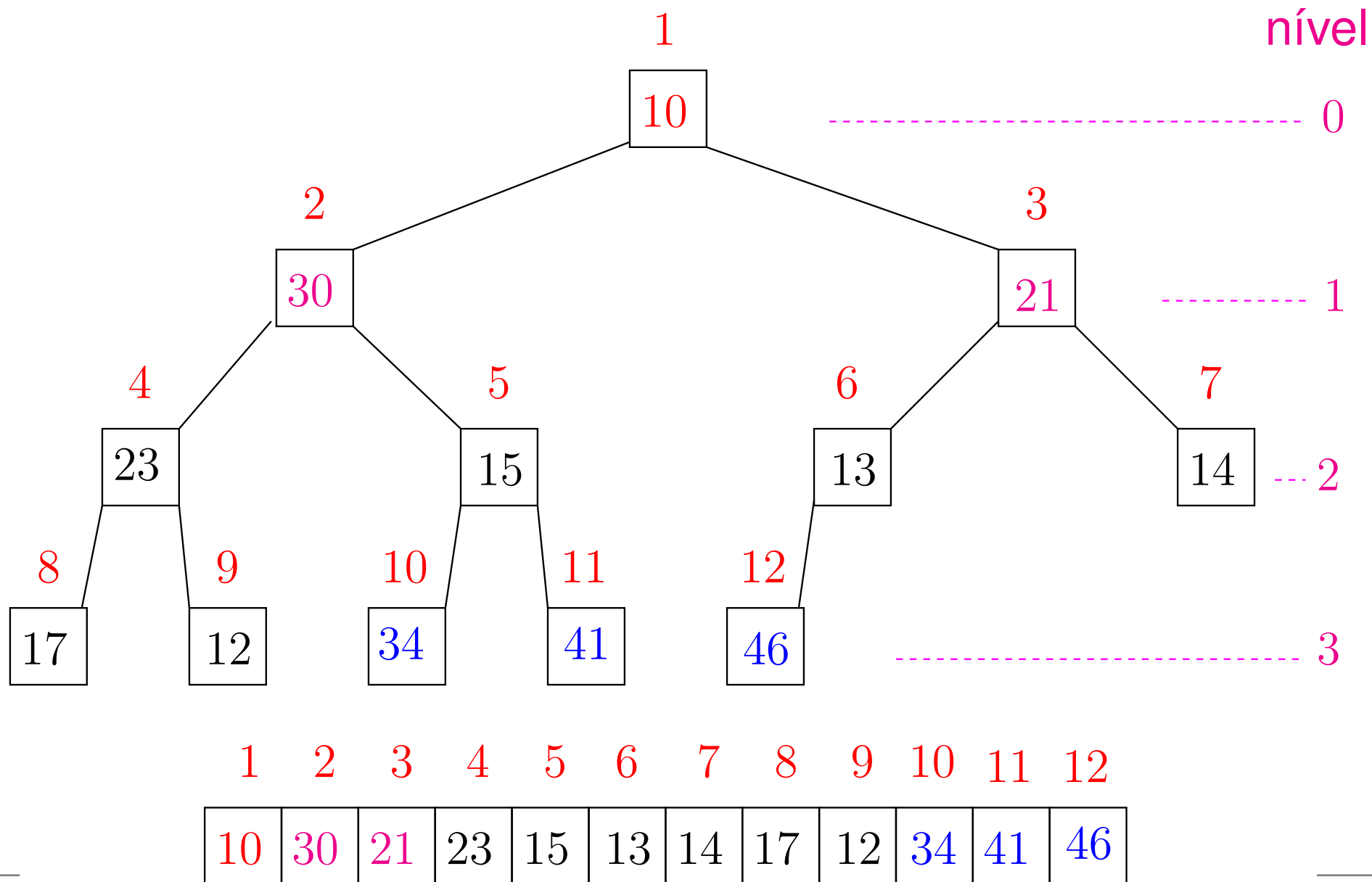
1	2	3	4	5	6	7	8	9	10	11	12
34	30	21	23	15	13	14	17	12	10	41	46

Heap sort

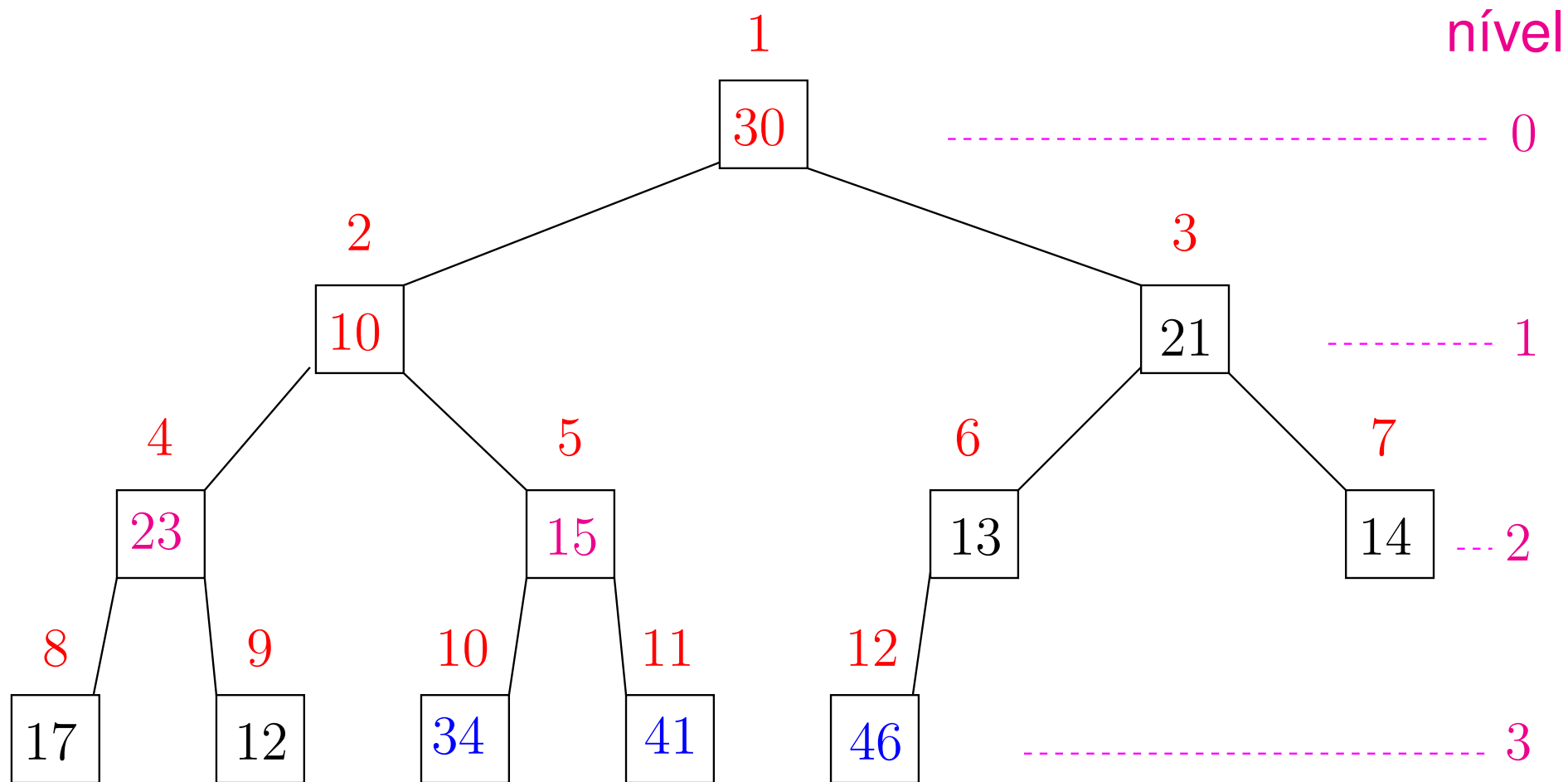


1	2	3	4	5	6	7	8	9	10	11	12
10	30	21	23	15	13	14	17	12	34	41	46

Heap sort

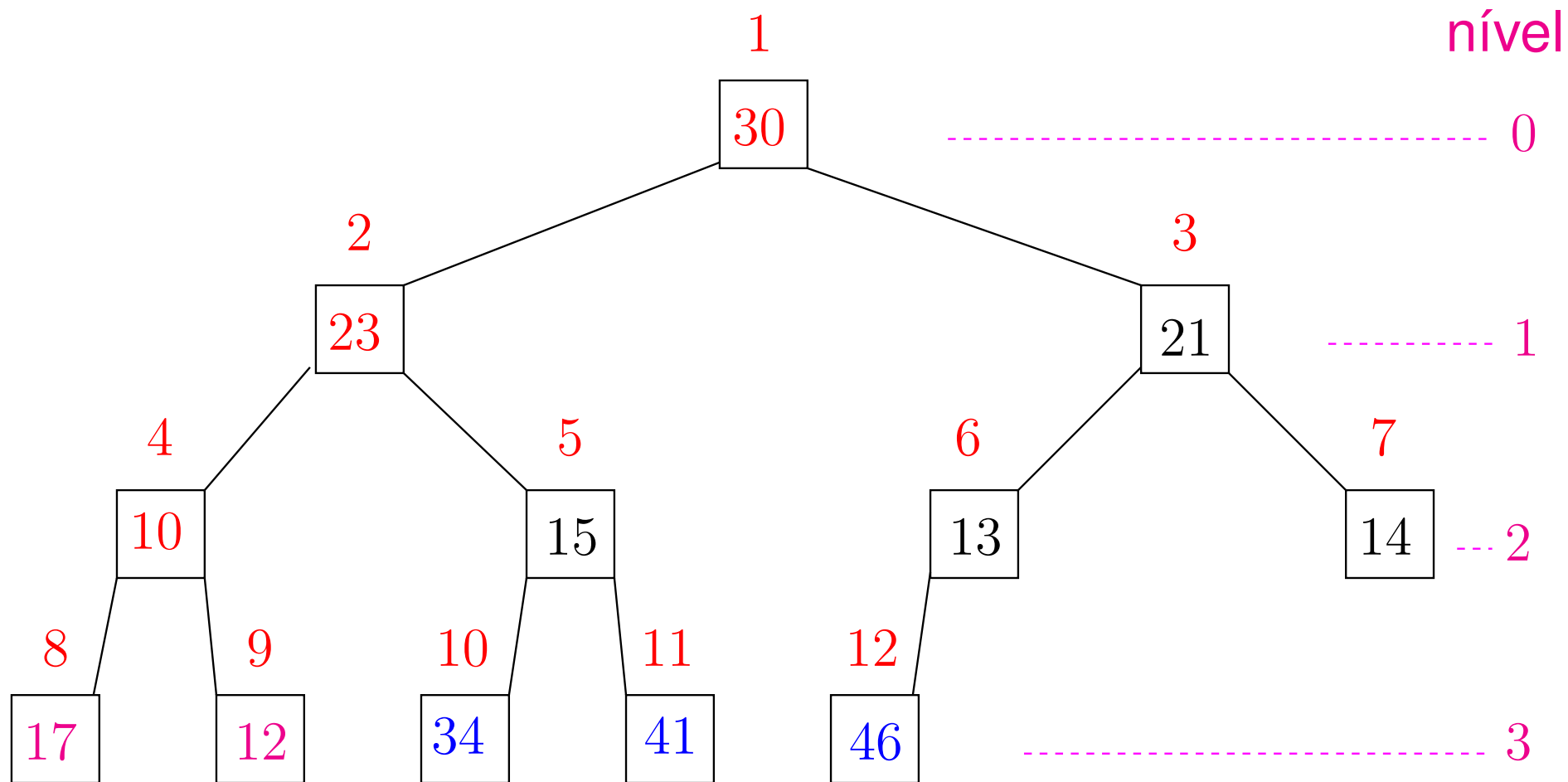


Heap sort



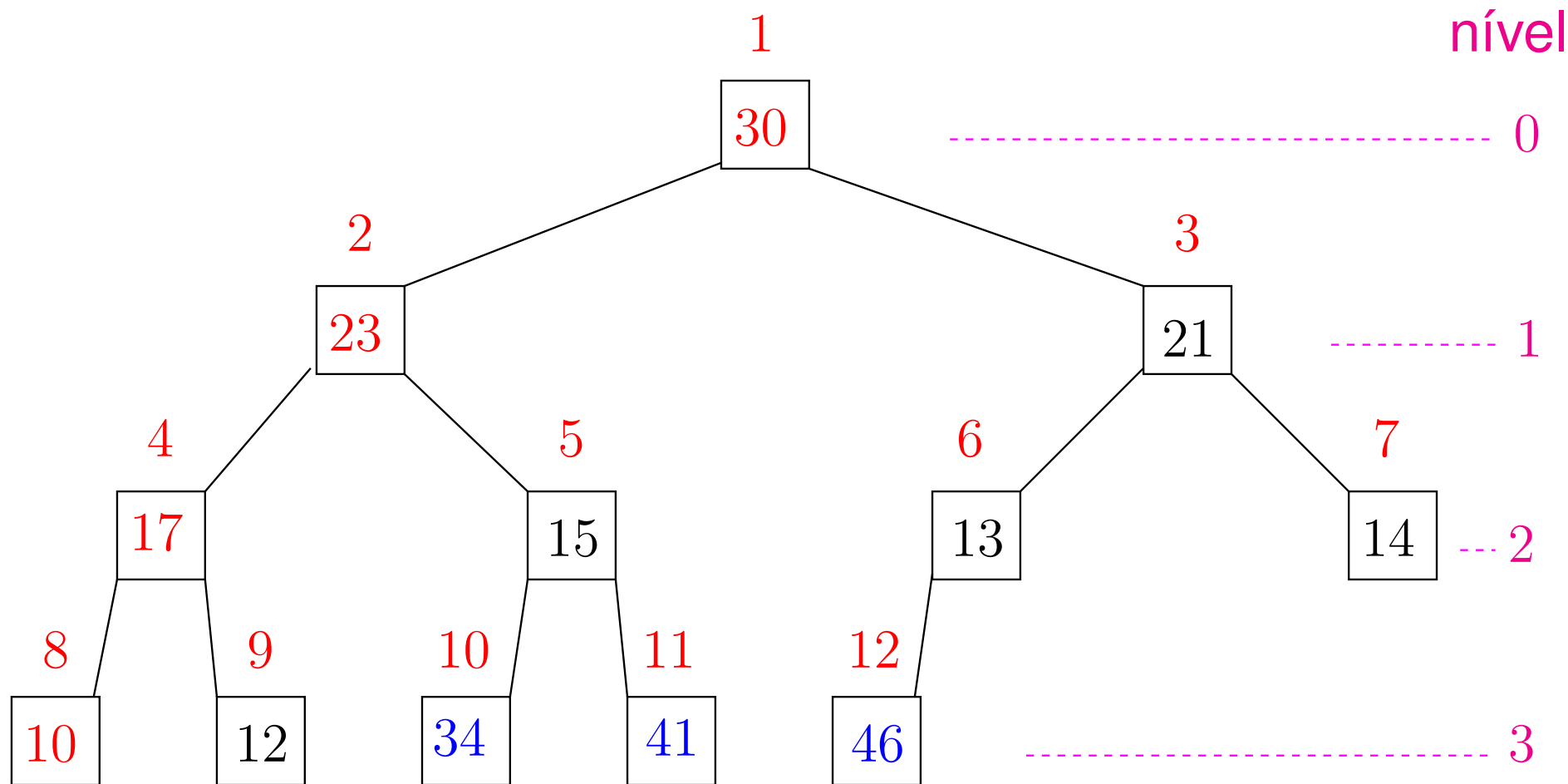
1	2	3	4	5	6	7	8	9	10	11	12
30	10	21	23	15	13	14	17	12	34	41	46

Heap sort



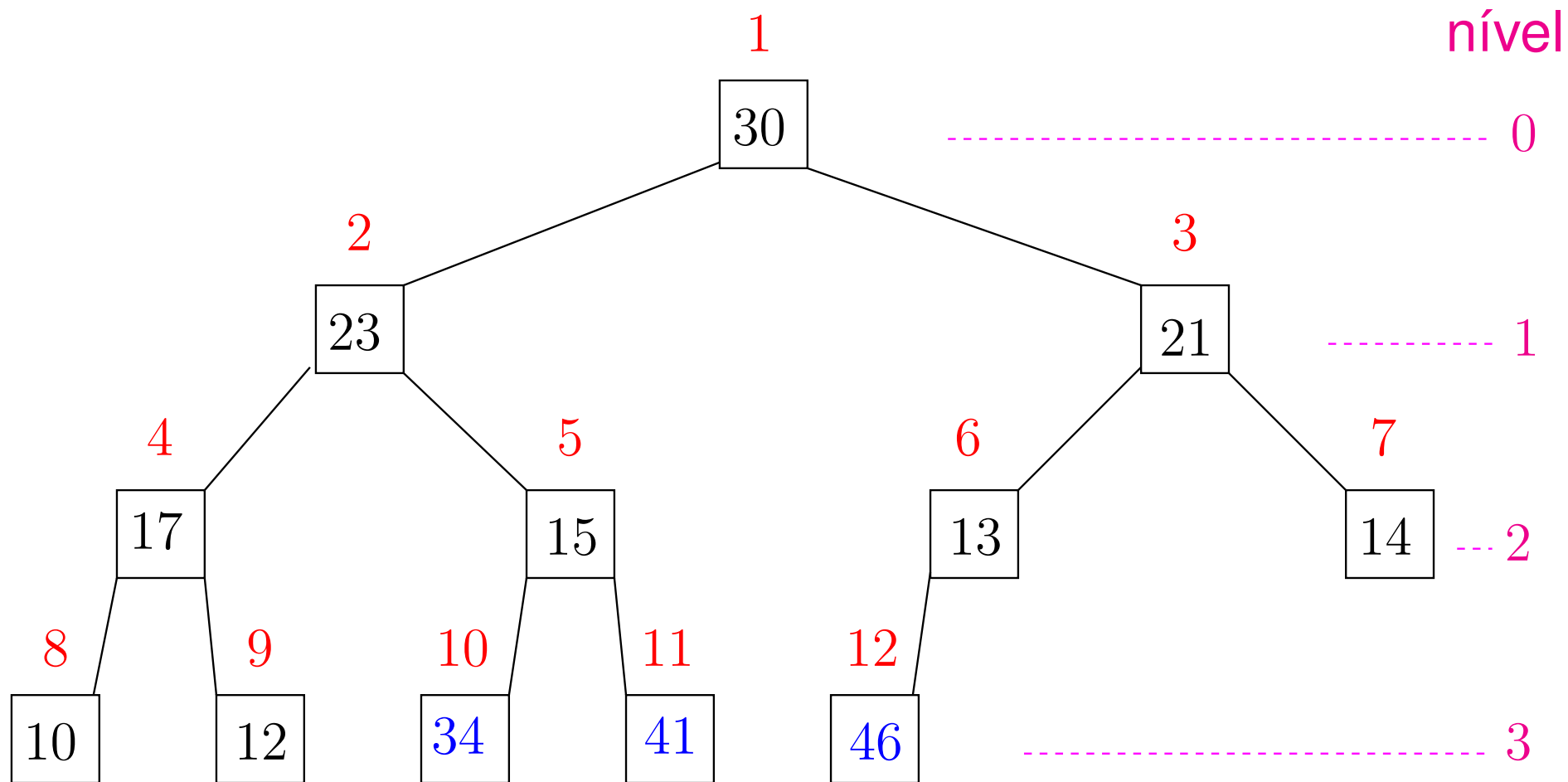
1	2	3	4	5	6	7	8	9	10	11	12
30	23	21	10	15	13	14	17	12	34	41	46

Heap sort



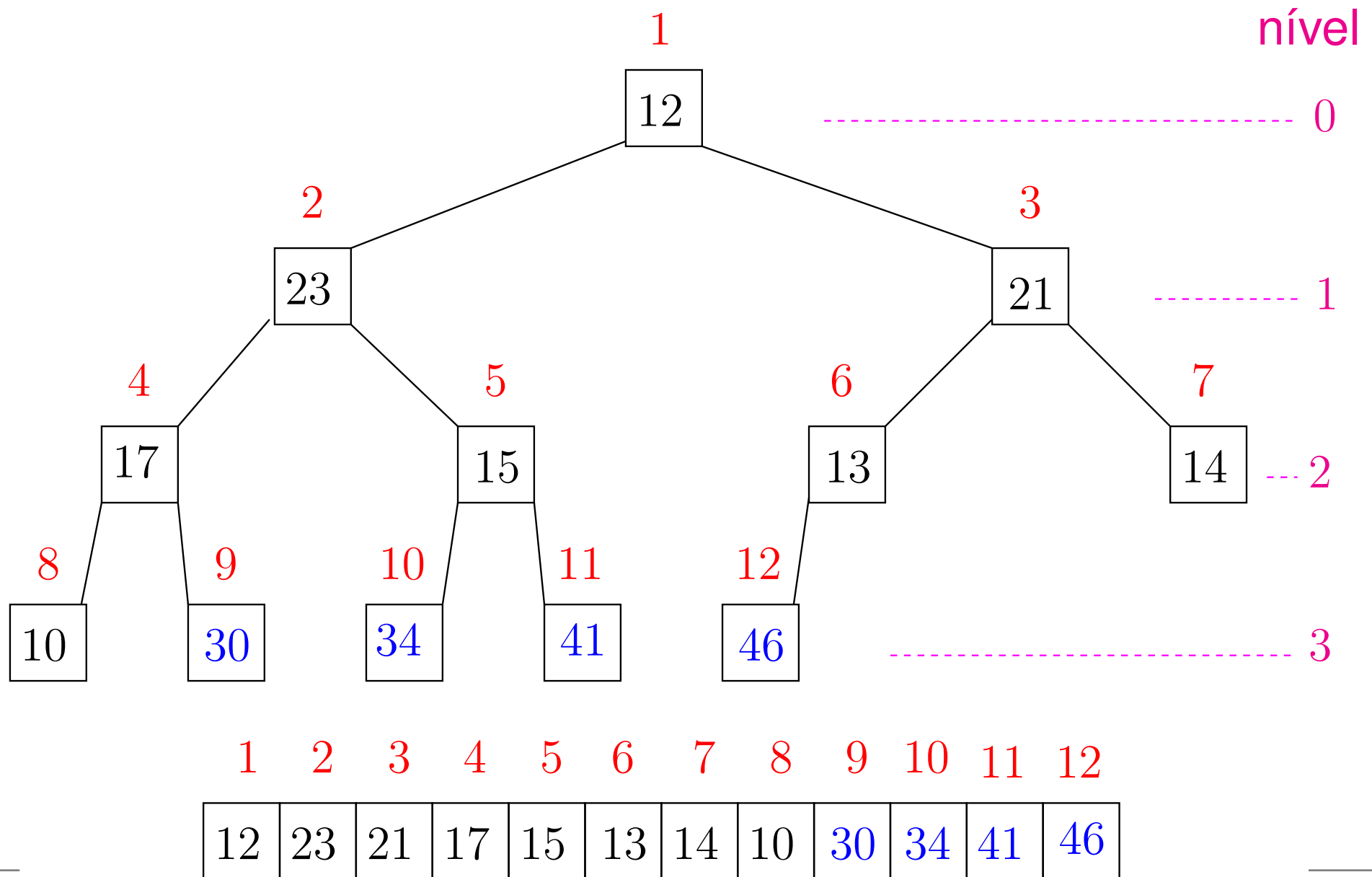
1	2	3	4	5	6	7	8	9	10	11	12
30	23	21	17	15	13	14	10	12	34	41	46

Heap sort



1	2	3	4	5	6	7	8	9	10	11	12
30	23	21	17	15	13	14	10	12	34	41	46

Heap sort



Heap sort

Algoritmo rearranja $A[1 \dots n]$ em ordem crescente.

HEAPSORT (A, n)

0 **CONSTRÓI-HEAP** (A, n) $\triangleright$ pré-processamento

1 $m \leftarrow n$

2 **para** $i \leftarrow n$ **decrecendo até** 2 **faça**

3 $A[1] \leftrightarrow A[i]$

4 $m \leftarrow m - 1$

5 **DESCE-HEAP** ($A, m, 1$)

Relações invariantes: Na linha 2 vale que:

(i0) $A[m \dots n]$ é crescente;

(i1) $A[1 \dots m] \leq A[m + 1]$;

(i2) $A[1 \dots m]$ 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	$= \Theta(n)$
5	$= nO(\lg n)$
total	$= nO(\lg n) + \Theta(n) = O(n \lg n)$

O consumo de tempo do algoritmo **HEAPSORT** é $O(n \lg n)$.

Exercícios

Exercício 9.A

A altura de i em $A[1..m]$ é o comprimento da mais longa seqüência da forma

$$\langle \text{filho}(i), \text{filho}(\text{filho}(i)), \text{filho}(\text{filho}(\text{filho}(i))), \dots \rangle$$

onde $\text{filho}(i)$ 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 $\lceil m/2^{h+1} \rceil \leq m/2^h$ quando $h \leq \lfloor \lg m \rfloor$.

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 número 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 que

$$m'' \leq m' < 2m/3.$$

Mais exercícios

Exercício 9.F

Mostre que a solução da recorrência

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

é $O(\log k)$. Mais geral: mostre que se $T(k) = T(2k/3) + O(1)$ então $O(\log k)$.

(Curiosidade: Essa é a recorrência do **DESCE-HEAP** (A, m, i) se interpretarmos k como sendo o número de nós na subárvore com raiz i).

Exercício 9.G

Escreva uma versão iterativa do algoritmo **DESCE-HEAP**. Faça uma análise do consumo de tempo do algoritmo.

Mais exercícios ainda

Exercício 9.H

Discuta a seguinte variante do algoritmo **DESCE-HEAP**:

D-H (A, m, i)

1 $e \leftarrow 2i$

2 $d \leftarrow 2i + 1$

3 **se** $e \leq m$ e $A[e] > A[i]$

4 **então** $A[i] \leftrightarrow A[e]$

5 **D-H** (A, m, e)

6 **se** $d \leq m$ e $A[d] > A[i]$

7 **então** $A[i] \leftrightarrow A[d]$

8 **D-H** (A, m, d)