Quicksort aleatorizado

CLRS 7.4

Quicksort aleatorizado

```
PARTICIONE-ALEA(A, p, d)
    i ← RANDOM(p, d)
    A[i] ↔ A[d]
    devolva PARTICIONE(A, p, d)
```

```
QUICKSORT-ALE(A, p, d)
    se p < d
        então q ← PARTICIONE-ALEA(A, p, d)
              QUICKSORT-ALE(A, p, q − 1)
              QUICKSORT-ALE(A, q + 1, d)
```

Análise do consumo esperado de tempo?
Basta calcular o número esperado de execuções da linha 4 do PARTICIONE numa chamada do QUICKSORT-ALE.

Particione

Rearranja $A[p \,..\, d]$ de modo que $p \leq q \leq d$ e
$A[p \,..\, q{-}1] \leq A[q] < A[q{+}1 \,..\, d]$

PARTICIONE (A, p, d)
$x \leftarrow A[d]$ $\triangleright$ x é o "pivô"
$i \leftarrow p{-}1$
para $j \leftarrow p$ **até** $d - 1$ **faça**
  se $A[j] \leq x$
    então $i \leftarrow i + 1$
      $A[i] \leftrightarrow A[j]$
$A[i{+}1] \leftrightarrow A[d]$
devolva $i + 1$

Exemplos

Número médio de execuções da linha 4 do PARTICIONE.

Suponha que $A[p\,..\,r]$ é permutação de $1\,..\,n$.

$A[p\,..\,r]$	execs
1,2	1
2,1	1
média	1

$A[p\,..\,r]$	execs
1,2,3	2+1
2,1,3	2+1
1,3,2	2+0
3,1,2	2+0
2,3,1	2+1
3,2,1	2+1
média	16/6

Mais um exemplo

$A[p\,..\,r]$	execs	$A[p\,..\,r]$	execs
1,2,3,4	3+3	1,3,4,2	3+1
2,1,3,4	3+3	3,1,4,2	3+1
1,3,2,4	3+2	1,4,3,2	3+1
3,1,2,4	3+2	4,1,3,2	3+1
2,3,1,4	3+3	3,4,1,2	3+1
3,2,1,4	3+3	4,3,1,2	3+1
1,2,4,3	3+1	2,3,4,1	3+3
2,1,4,3	3+1	3,2,4,1	3+3
1,4,2,3	3+1	2,4,3,1	3+2
4,1,2,3	3+1	4,2,3,1	3+2
2,4,1,3	3+1	3,4,2,1	3+3
4,2,1,3	3+1	4,3,2,1	3+3
		média	116/24

Formalização

Seja X o número de execuções da linha 4 do PARTICIONE numa chamada do QUICKSORT-ALE.

Formalização

Seja X o número de execuções da linha 4 do PARTICIONE numa chamada do QUICKSORT-ALE.

X é uma variável aleatória que depende dos sorteios efetuados pelo algoritmo QUICKSORT-ALE.

Para estimar o consumo de tempo esperado de QUICKSORT-ALE, queremos calcular $\mathrm{E}[X]$.

Formalização

Seja X o número de execuções da linha 4 do PARTICIONE numa chamada do QUICKSORT-ALE.

X é uma variável aleatória que depende dos sorteios efetuados pelo algoritmo QUICKSORT-ALE.

Para estimar o consumo de tempo esperado de QUICKSORT-ALE, queremos calcular $\mathrm{E}[X]$.

Ideia: Escrever X como uma soma de variáveis aleatórias binárias, cuja esperança é mais fácil de calcular.

Formalização

Seja X o número de execuções da linha 4 do PARTICIONE numa chamada do QUICKSORT-ALE.

X é uma variável aleatória que depende dos sorteios efetuados pelo algoritmo QUICKSORT-ALE.

Para estimar o consumo de tempo esperado de QUICKSORT-ALE, queremos calcular $\mathrm{E}[X]$.

Ideia: Escrever X como uma soma de variáveis aleatórias binárias, cuja esperança é mais fácil de calcular.

Quem serão essas variáveis aleatórias binárias?

Consumo de tempo esperado

Para facilitar, considere que
$A[1\,..\,n]$ é uma permutação de 1 a n.
(A conclusão vale independente dessa hipótese.)

Seja

X_{ab} = número de comparações entre a e b
na linha 4 de PARTICIONE.

Queremos calcular

$$
\begin{aligned}
X &= \text{total de execuções da linha 4 do PARTICIONE} \\
&= \sum_{a=1}^{n-1} \sum_{b=a+1}^{n} X_{ab}
\end{aligned}
$$

Consumo de tempo esperado

Supondo $a < b$,

$$X_{ab} = \begin{cases} 1 & \text{se o primeiro pivô em } \{a, \ldots, b\} \text{ é } a \text{ ou } b \\ 0 & \text{caso contrário.} \end{cases}$$

Qual a probabilidade de X_{ab} valer 1?

Consumo de tempo esperado

Supondo $a < b$,

$$X_{ab} = \begin{cases} 1 & \text{se o primeiro pivô em } \{a, \ldots, b\} \text{ é } a \text{ ou } b \\ 0 & \text{caso contrário.} \end{cases}$$

Qual a probabilidade de X_{ab} valer 1?

$$\mathrm{E}[X_{ab}] = \Pr\{X_{ab}=1\} = \frac{1}{b-a+1} + \frac{1}{b-a+1}$$

Consumo de tempo esperado

Supondo $a < b$,

$$X_{ab} = \begin{cases} 1 & \text{se o primeiro pivô em } \{a, \ldots, b\} \text{ é } a \text{ ou } b \\ 0 & \text{caso contrário.} \end{cases}$$

Qual a probabilidade de X_{ab} valer 1?

$$\mathrm{E}[X_{ab}] \;=\; \mathrm{Pr}\,\{X_{ab}{=}1\} \;=\; \frac{1}{b-a+1} + \frac{1}{b-a+1}$$

$$X = \sum_{a=1}^{n-1} \sum_{b=a+1}^{n} X_{ab}$$

$\mathrm{E}[X] =$????

Consumo de tempo esperado

$$
\begin{aligned}
\mathrm{E}[X] &= \sum_{a=1}^{n-1} \sum_{b=a+1}^{n} \mathrm{E}[X_{ab}] \\
&= \sum_{a=1}^{n-1} \sum_{b=a+1}^{n} \text{Pr}\{X_{ab}=1\} \\
&= \sum_{a=1}^{n-1} \sum_{b=a+1}^{n} \frac{2}{b-a+1} \\
&= \sum_{a=1}^{n-1} \sum_{k=1}^{n-a} \frac{2}{k+1} \\
&< \sum_{a=1}^{n-1} 2\left(\frac{1}{1}+\frac{1}{2}+\cdots+\frac{1}{n}\right) \\
&< 2n\left(\frac{1}{1}+\frac{1}{2}+\cdots+\frac{1}{n}\right) < 2n\,(1+\ln n)
\end{aligned}
$$

CLRS (A.7), p.1060

Conclusões

O consumo de tempo esperado do algoritmo QUICKSORT-ALE é $\mathrm{O}(n \log n)$.

Do exercício 7.4-4 do CLRS temos que

O consumo de tempo esperado do algoritmo QUICKSORT-ALE é $\Theta(n \log n)$.

Heapsort

CLRS 6

Heap

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

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

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

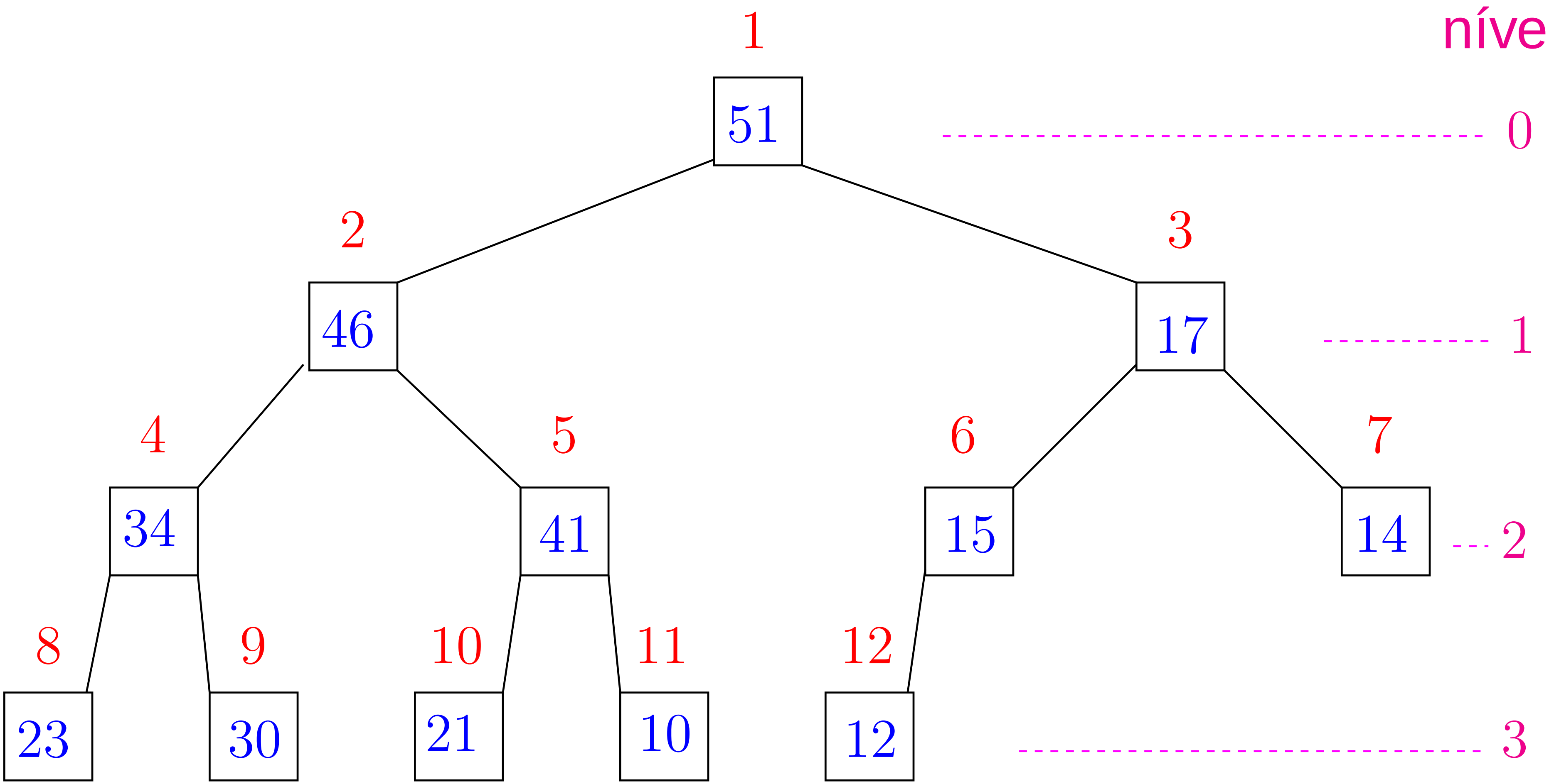
De uma forma mais geral, $A[j\,..\,m]$ é um heap se

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

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

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

Exemplo



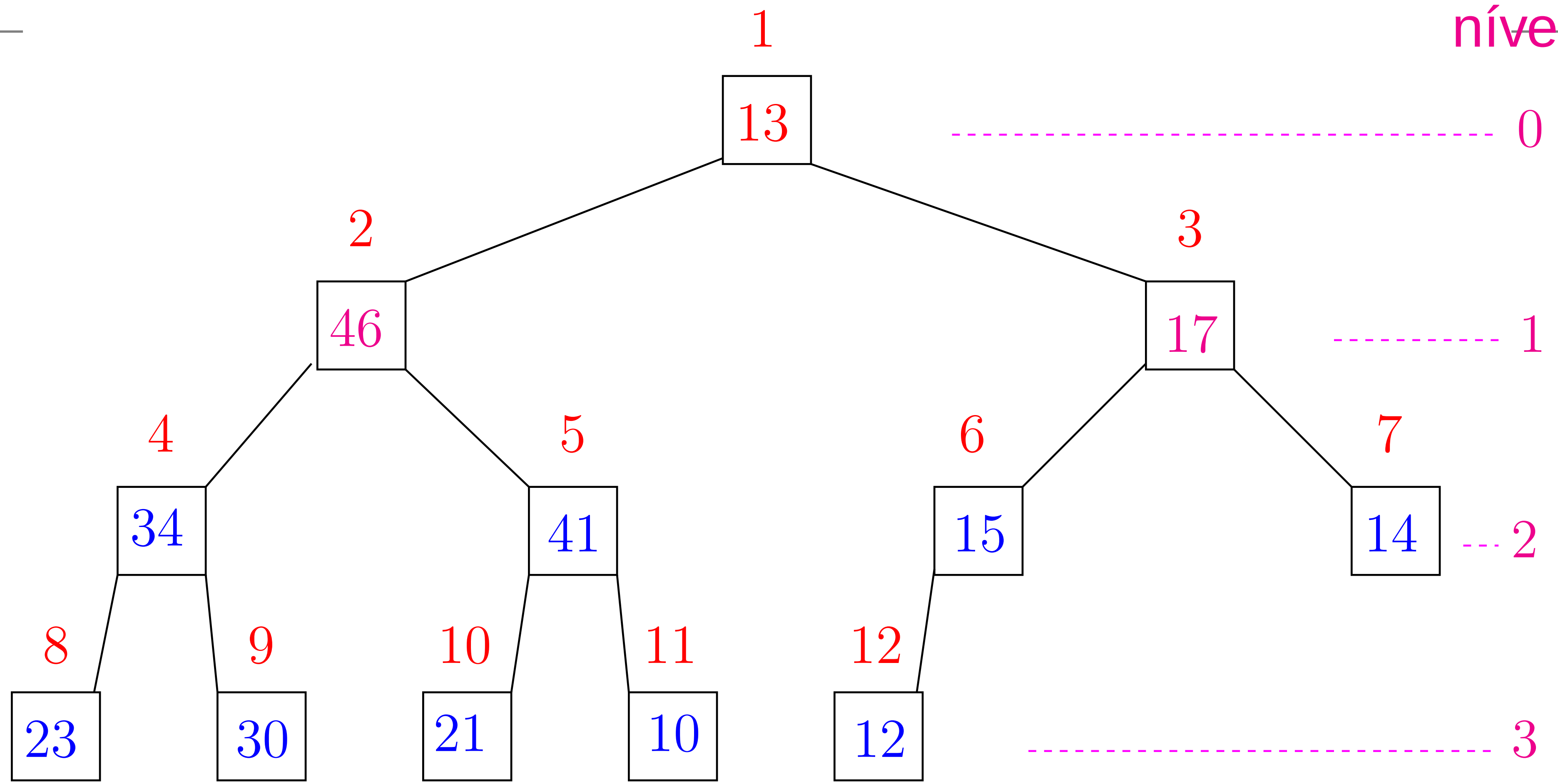
1	2	3	4	5	6	7	8	9	10	11	12
51	46	17	34	41	15	14	23	30	21	10	12

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.

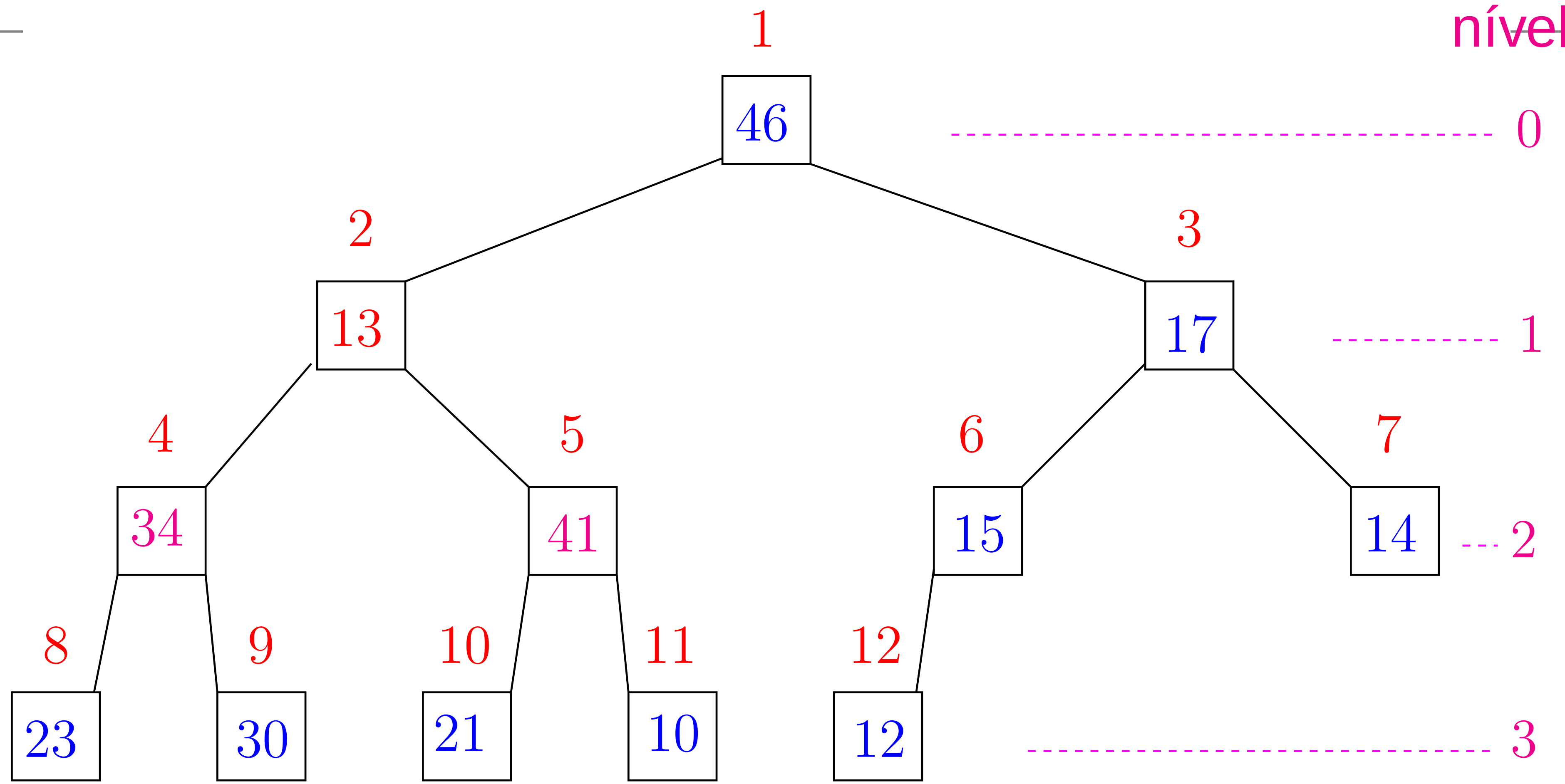
```
DESCE-HEAP (A, m, i)
    e ← 2i
    d ← 2i + 1
    se e ≤ m e A[e] > A[i]
        então maior ← e
        senão maior ← i
    se d ≤ m e A[d] > A[maior]
        então maior ← d
    se maior ≠ i
        então A[i] ↔ A[maior]
            DESCE-HEAP (A, m, maior)
```

Simulação



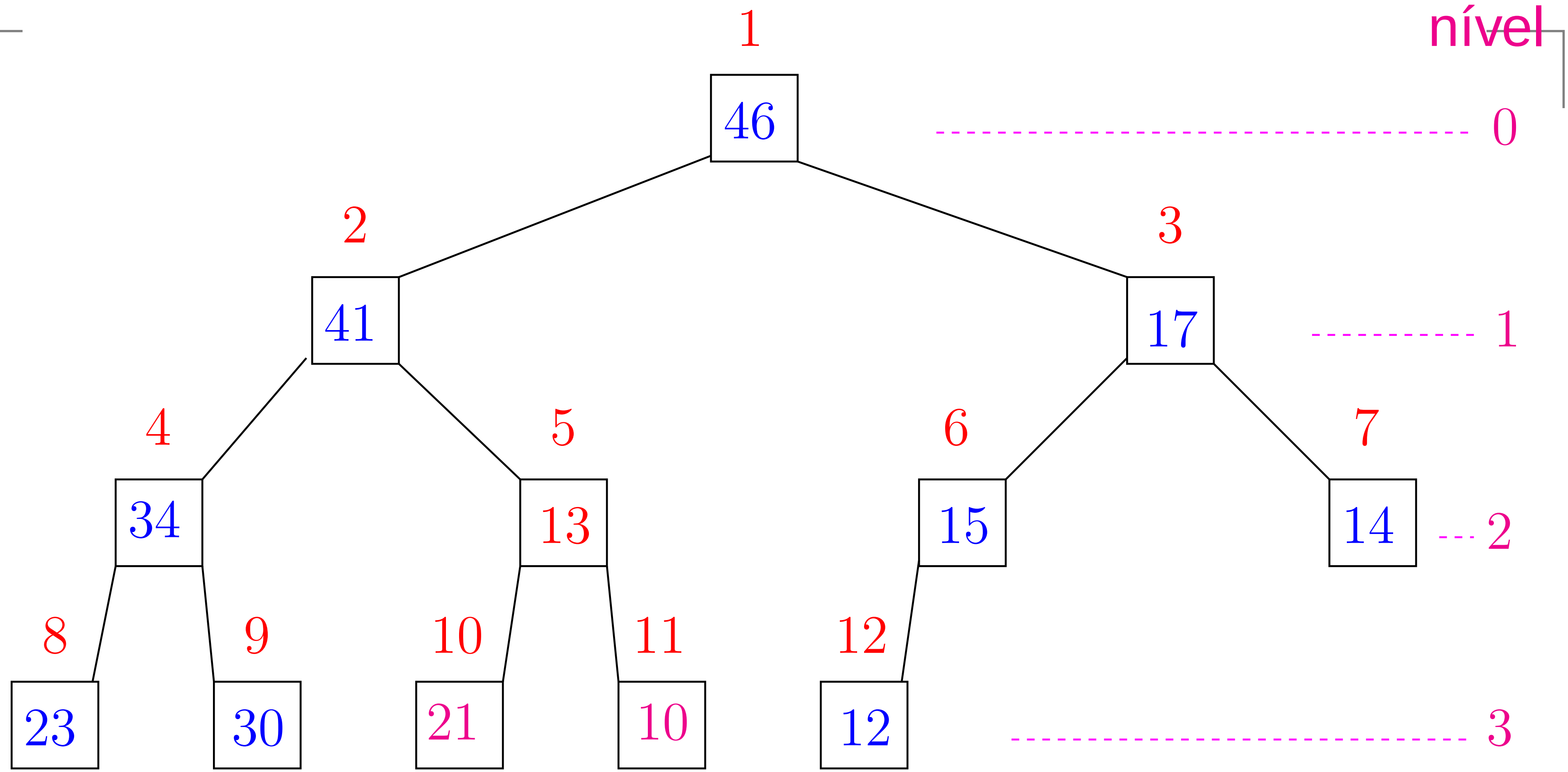
1	2	3	4	5	6	7	8	9	10	11	12
13	46	17	34	41	15	14	23	30	21	10	12

Simulação



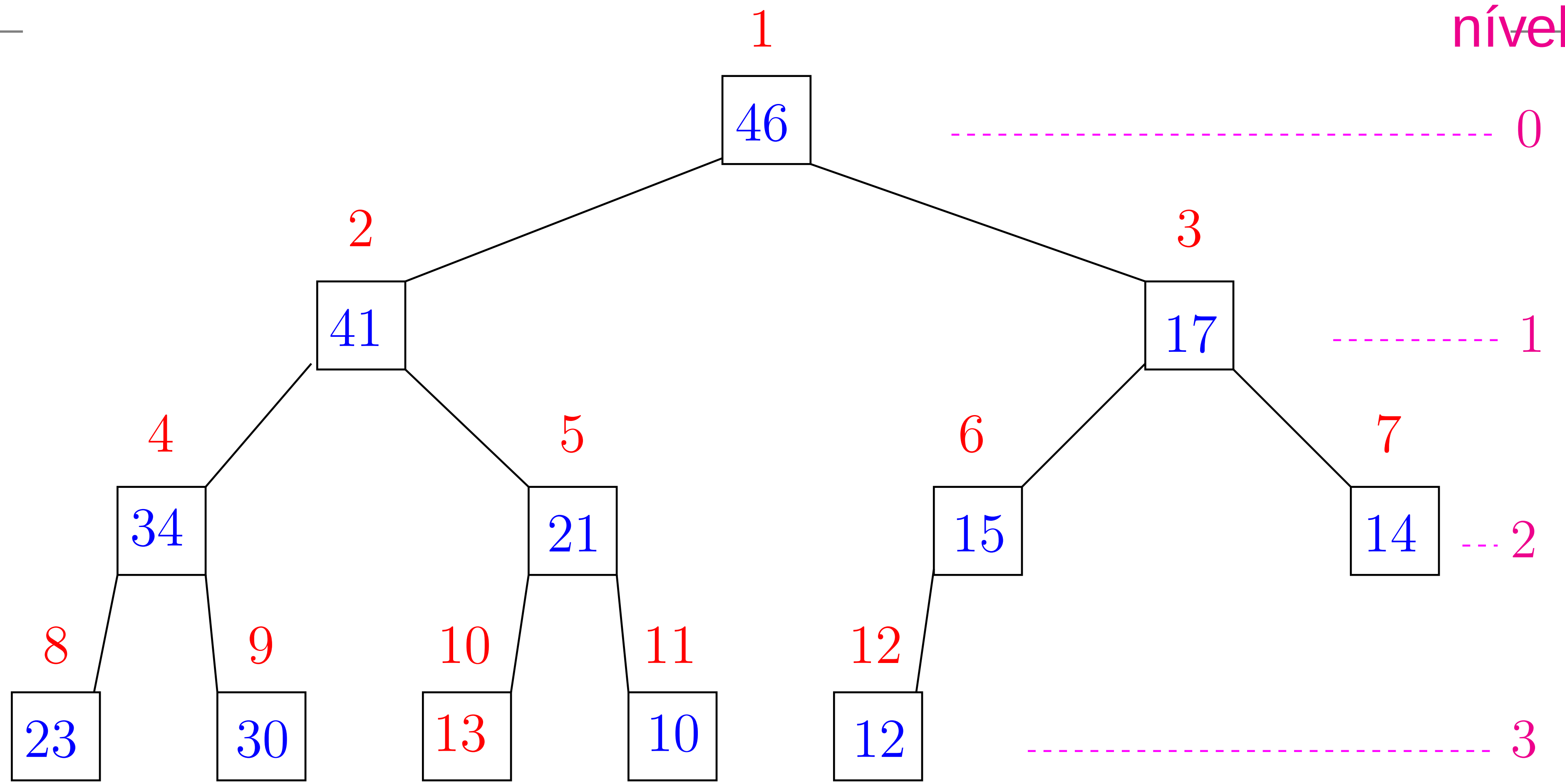
1	2	3	4	5	6	7	8	9	10	11	12
46	13	17	34	41	15	14	23	30	21	10	12

Simulação



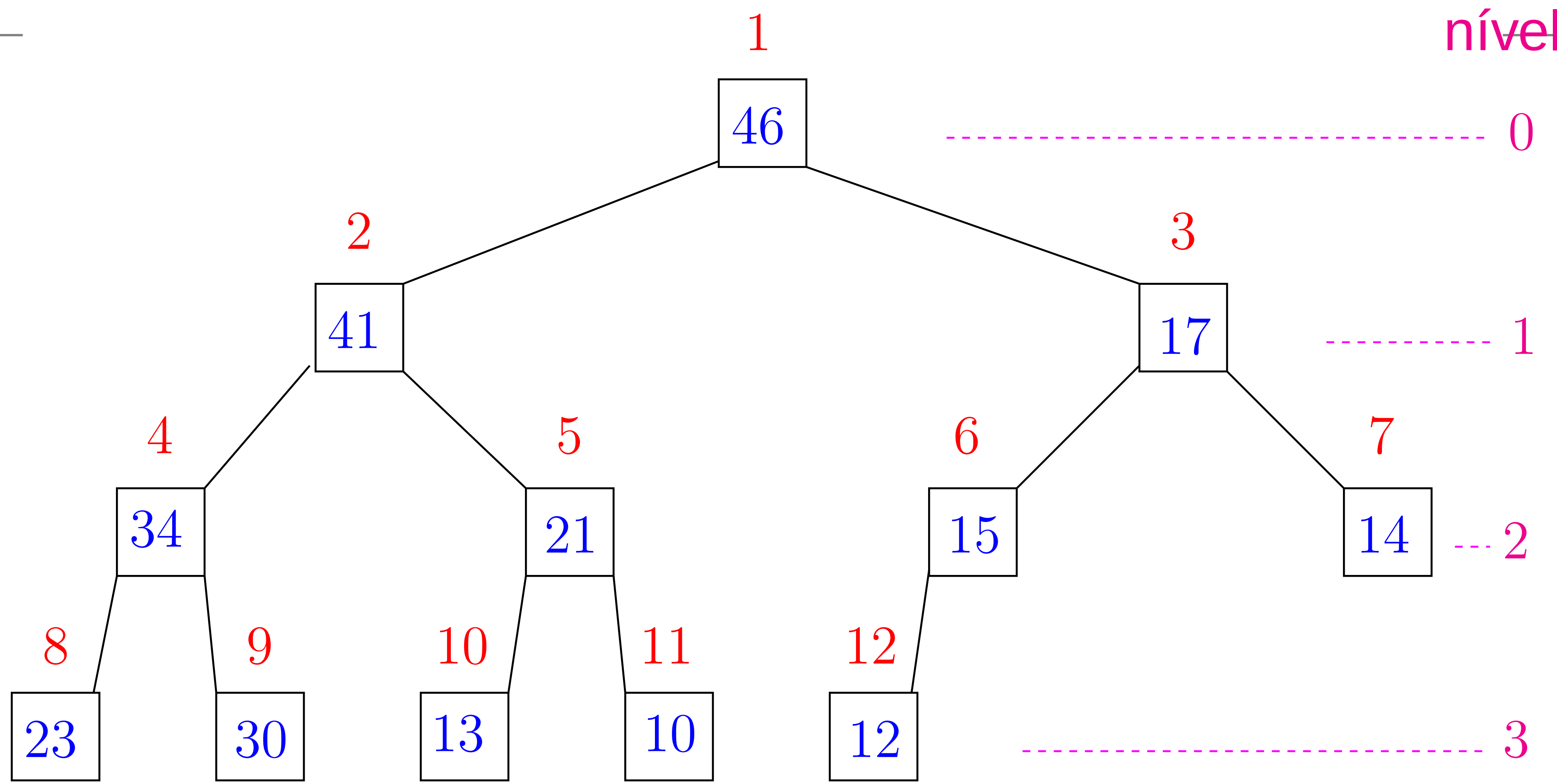
1	2	3	4	5	6	7	8	9	10	11	12
46	41	17	34	13	15	14	23	30	21	10	12

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

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

linha		todas as execuções da linha
1-3	$=$	$\Theta(1)$
4-5	$=$	$\Theta(1)$
6	$=$	$\Theta(1)$
7	$=$	$\mathrm{O}(1)$
8	$=$	$\Theta(1)$
9	$=$	$\mathrm{O}(1)$
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: $T(n)$ é ???.

Consumo de tempo

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

Recorrência associada:

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

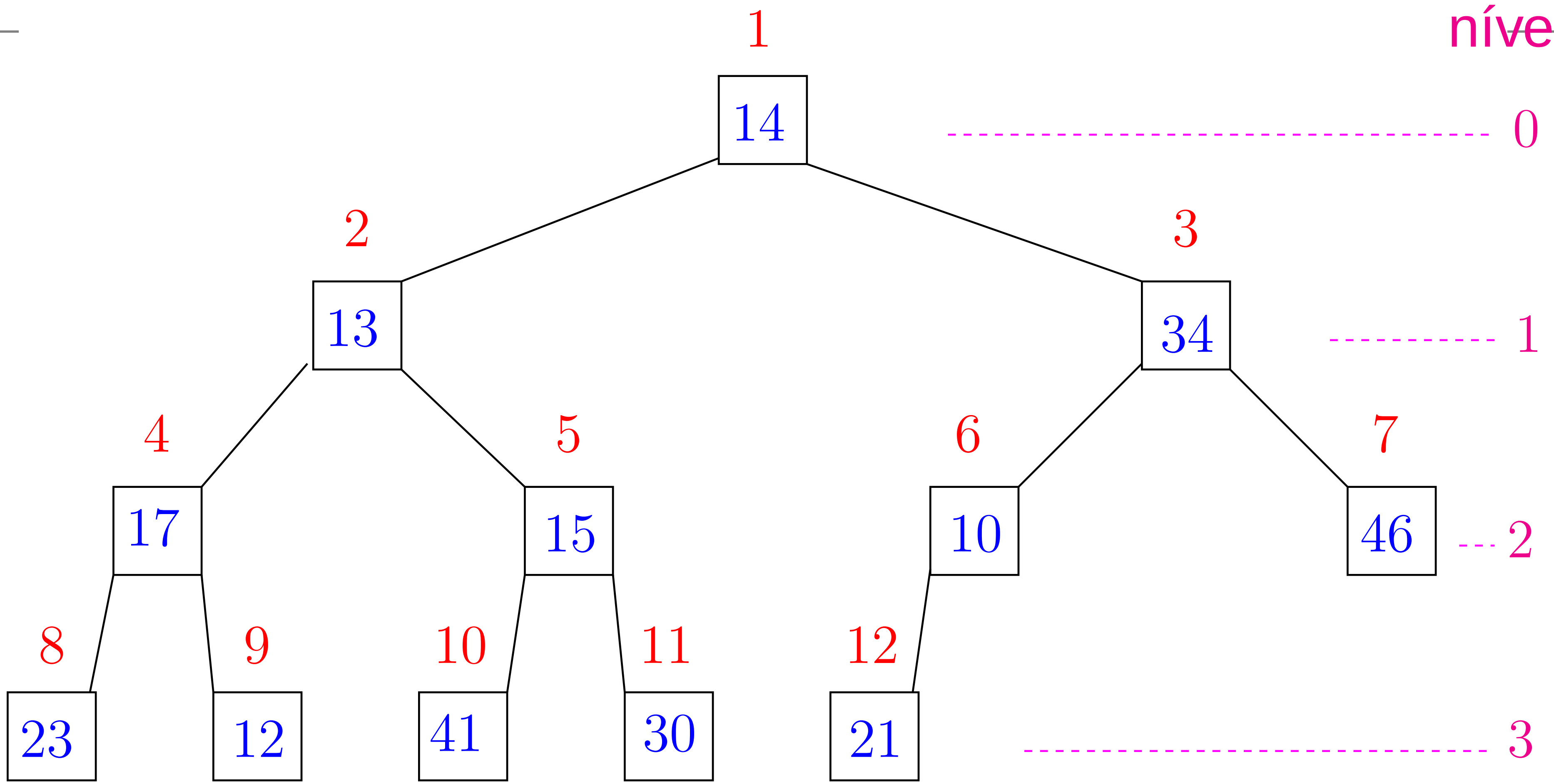
pois altura de $maior$ é $h-1$.

Solução assintótica: $T(n)$ é $\mathrm{O}(h)$.

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

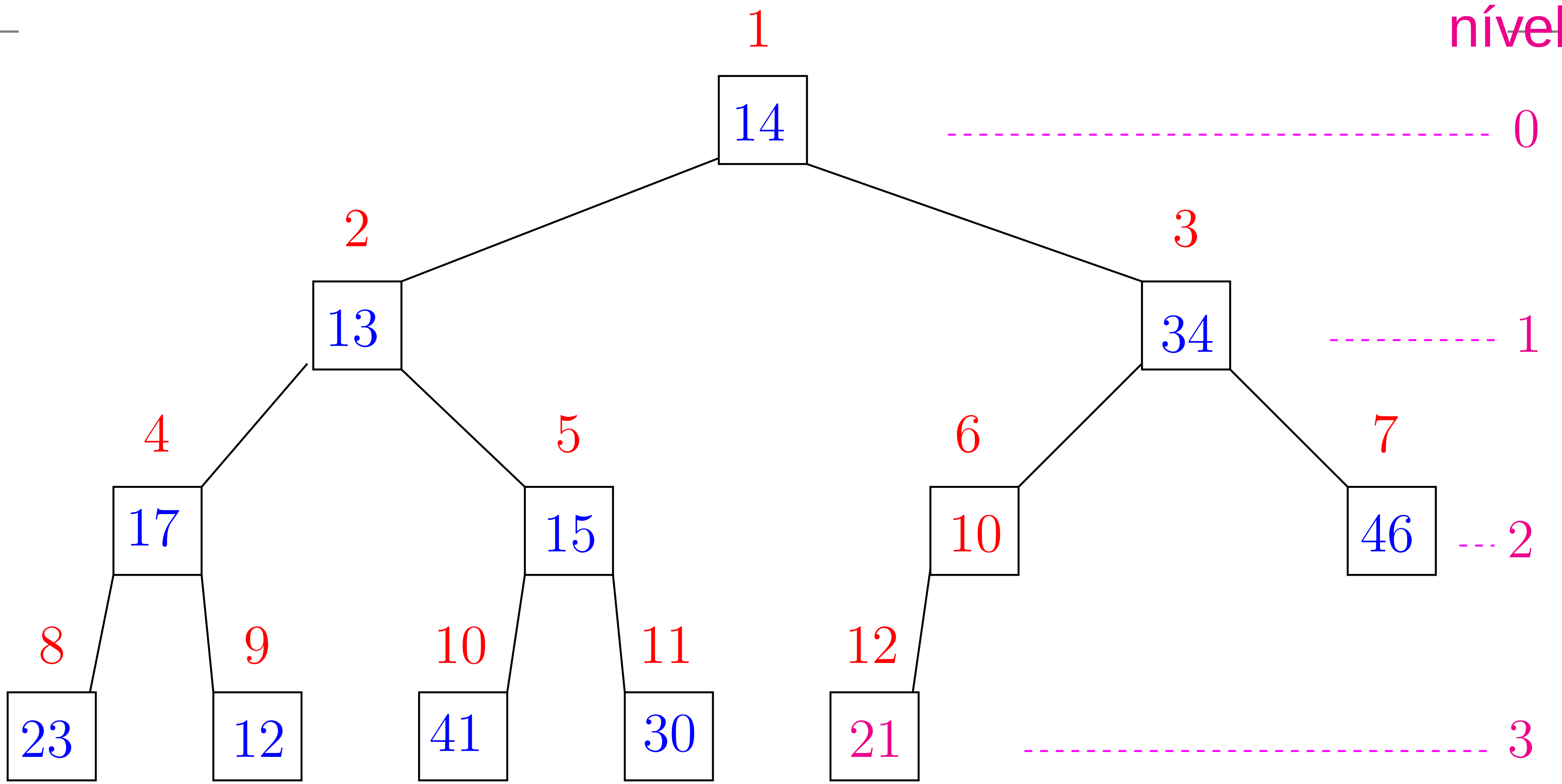
> O consumo de tempo do algoritmo DESCE-HEAP é $\mathrm{O}(\lg m)$ (ou melhor ainda, $\mathrm{O}(h)$).

Construção de um heap



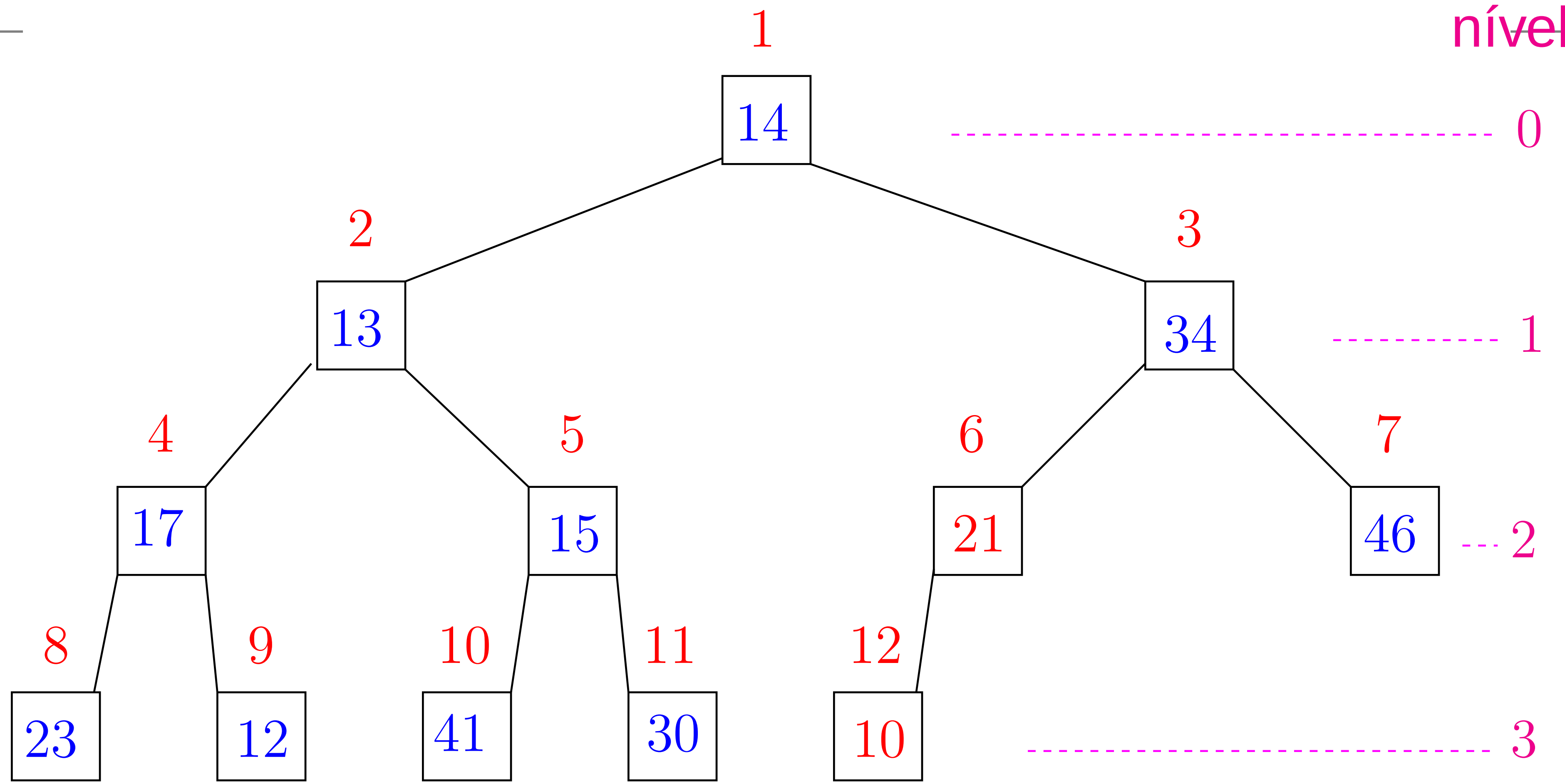
1	2	3	4	5	6	7	8	9	10	11	12
14	13	34	17	15	10	46	23	12	41	30	21

Construção de um heap



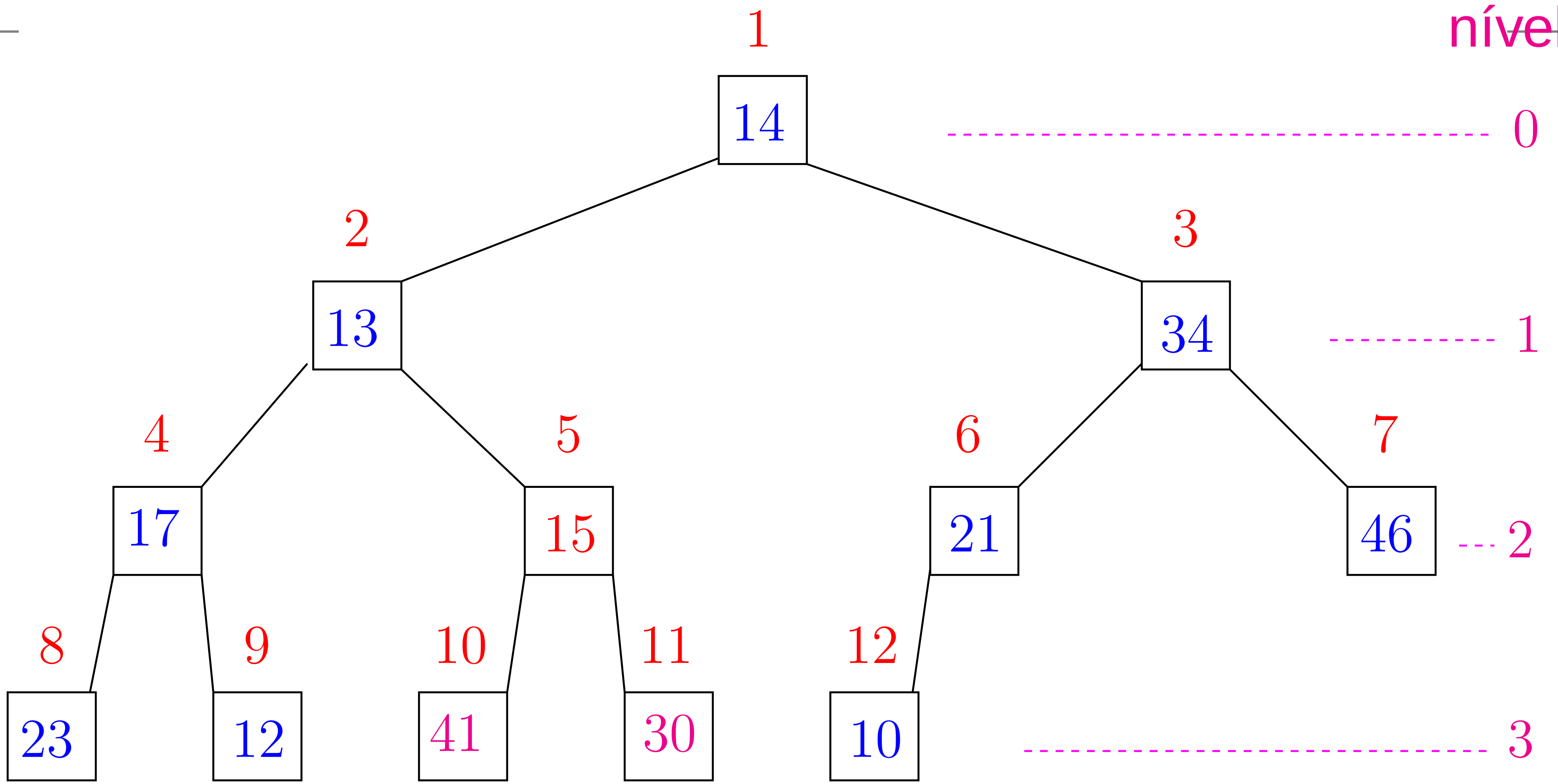
1	2	3	4	5	6	7	8	9	10	11	12
14	13	34	17	15	10	46	23	12	41	30	21

Construção de um heap



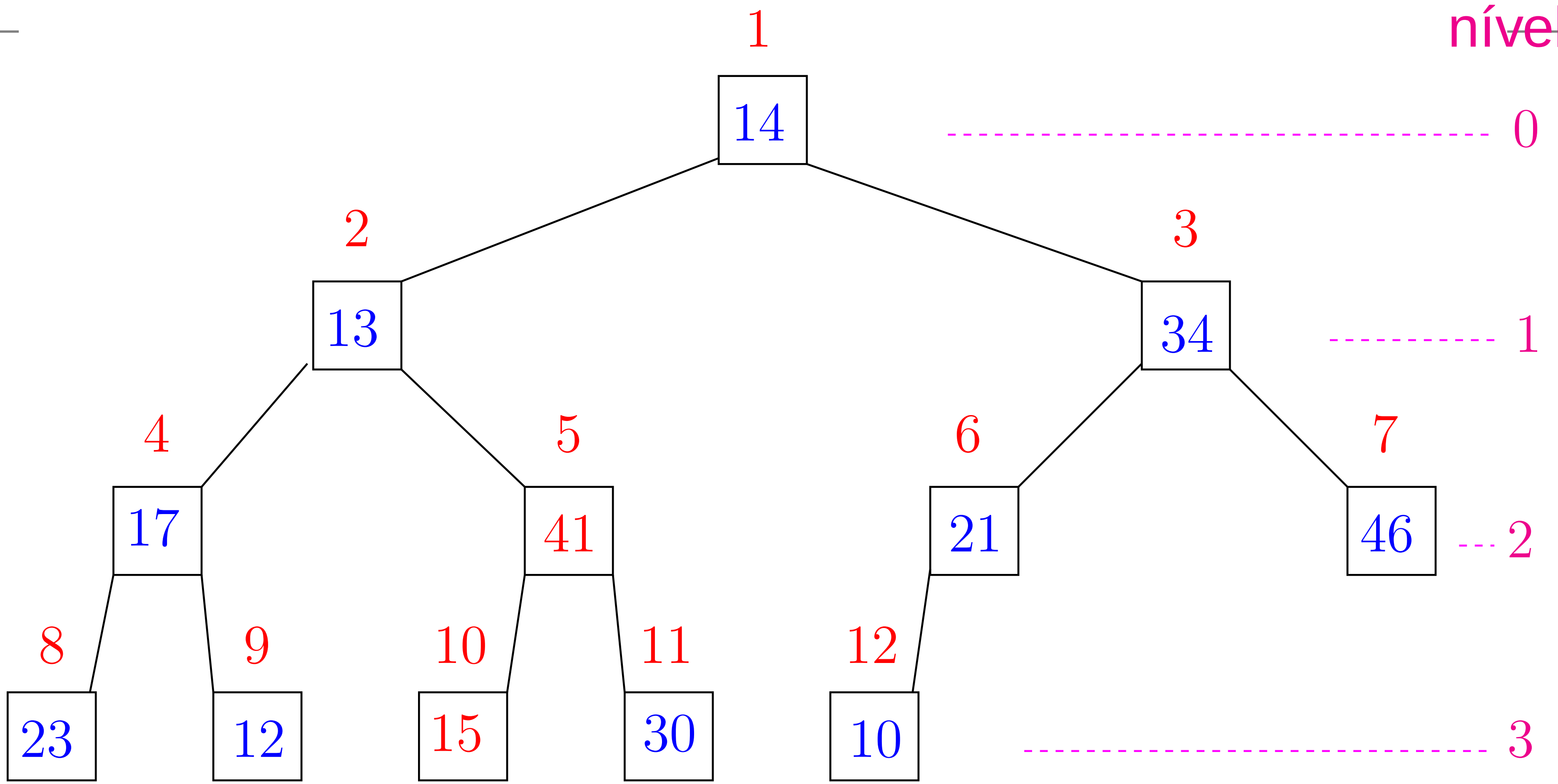
1	2	3	4	5	6	7	8	9	10	11	12
14	13	34	17	15	21	46	23	12	41	30	10

Construção de um heap



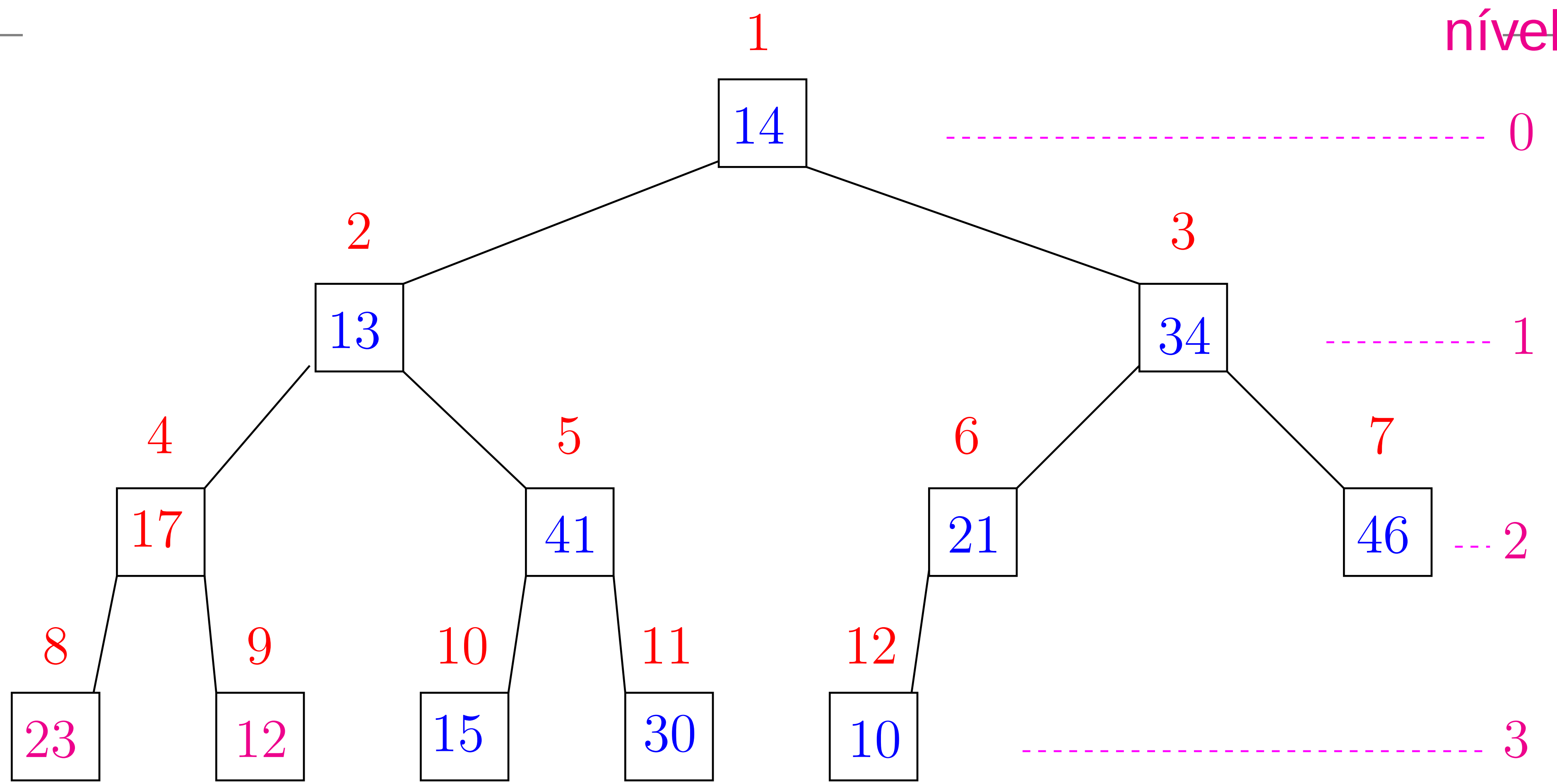
1	2	3	4	5	6	7	8	9	10	11	12
14	13	34	17	15	21	46	23	12	41	30	10

Construção de um heap



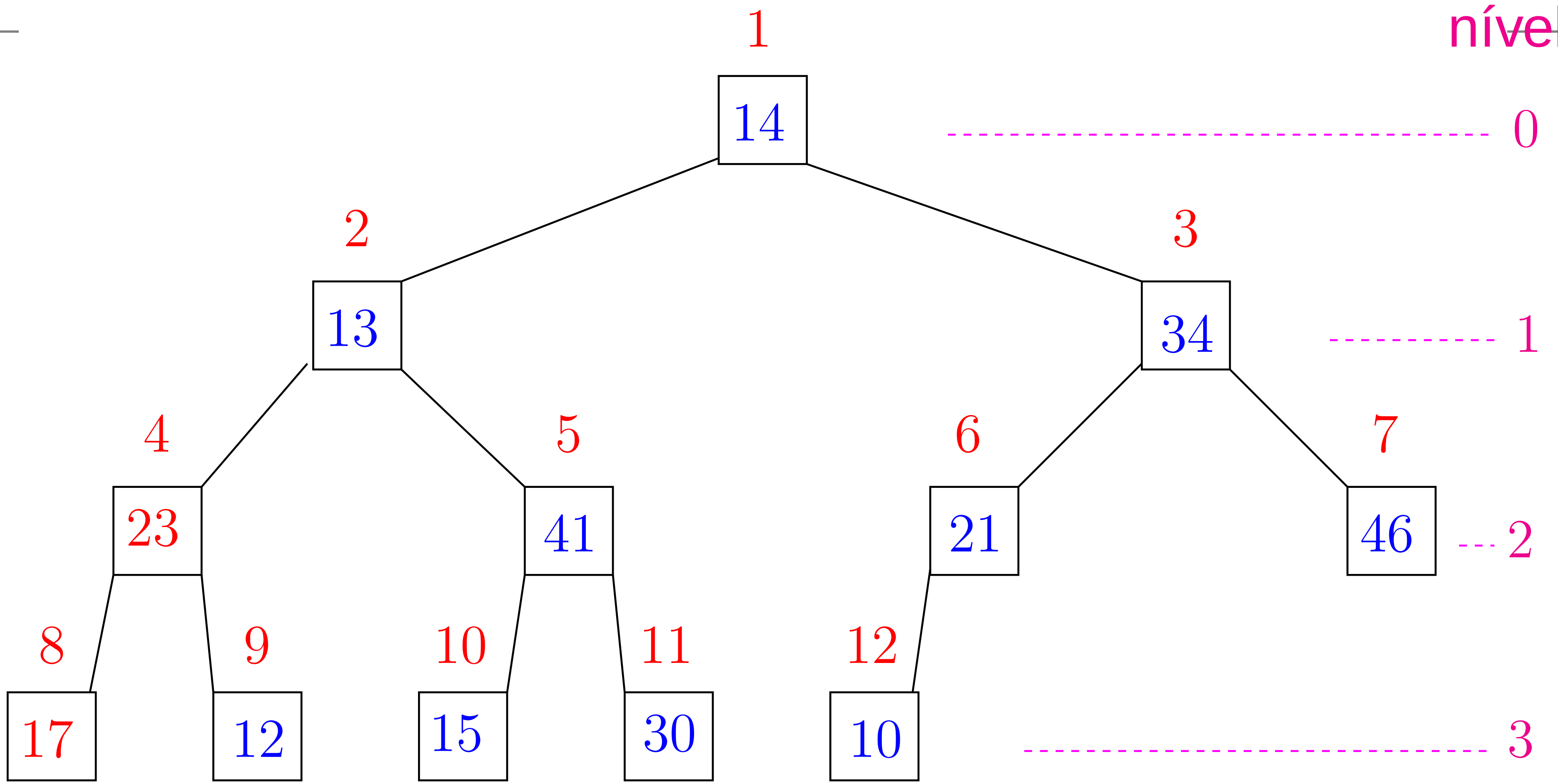
1	2	3	4	5	6	7	8	9	10	11	12
14	13	34	17	41	21	46	23	12	15	30	10

Construção de um heap



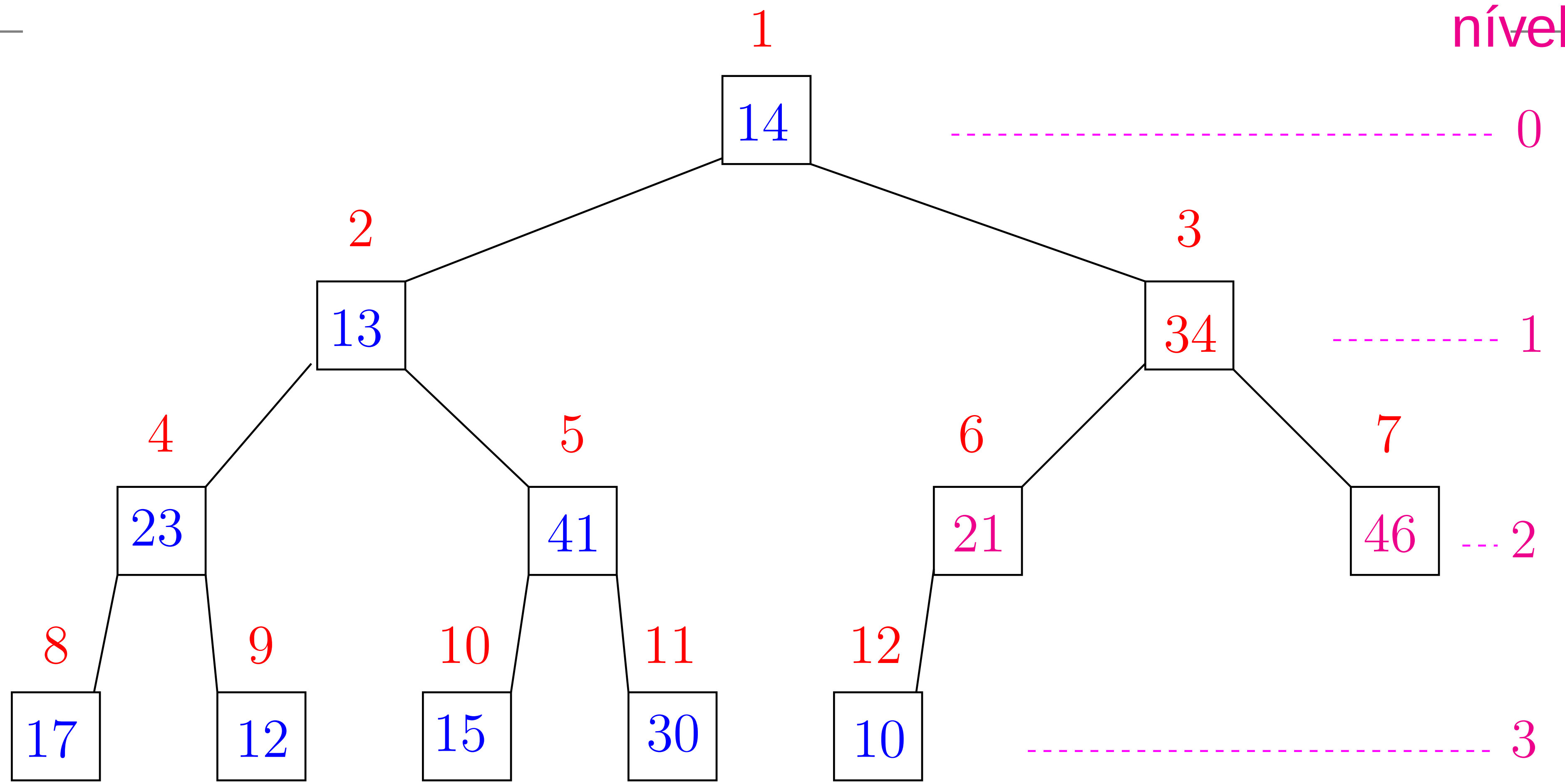
1	2	3	4	5	6	7	8	9	10	11	12
14	13	34	17	41	21	46	23	12	15	30	10

Construção de um heap



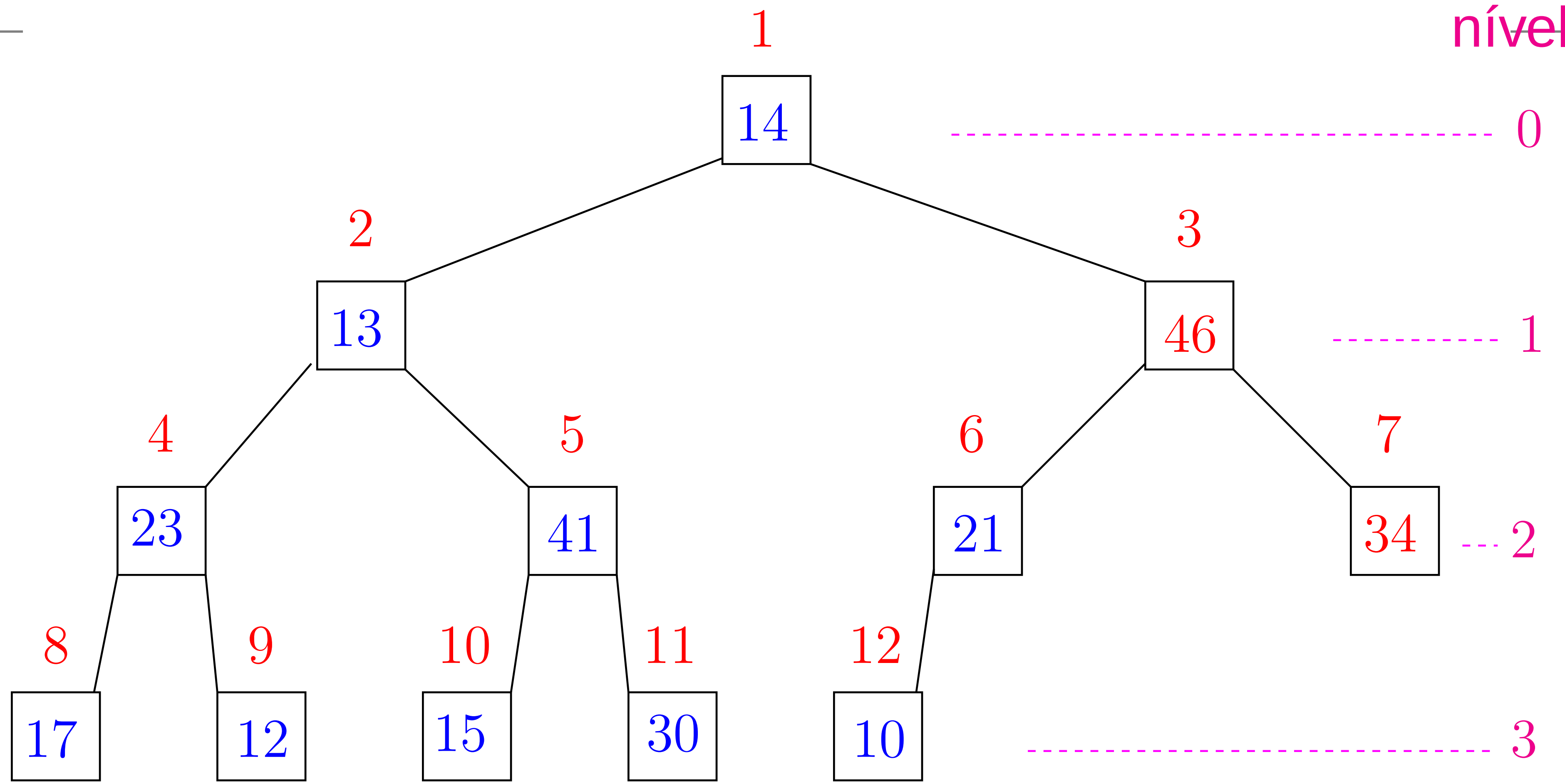
1	2	3	4	5	6	7	8	9	10	11	12
14	13	34	23	41	21	46	17	12	15	30	10

Construção de um heap



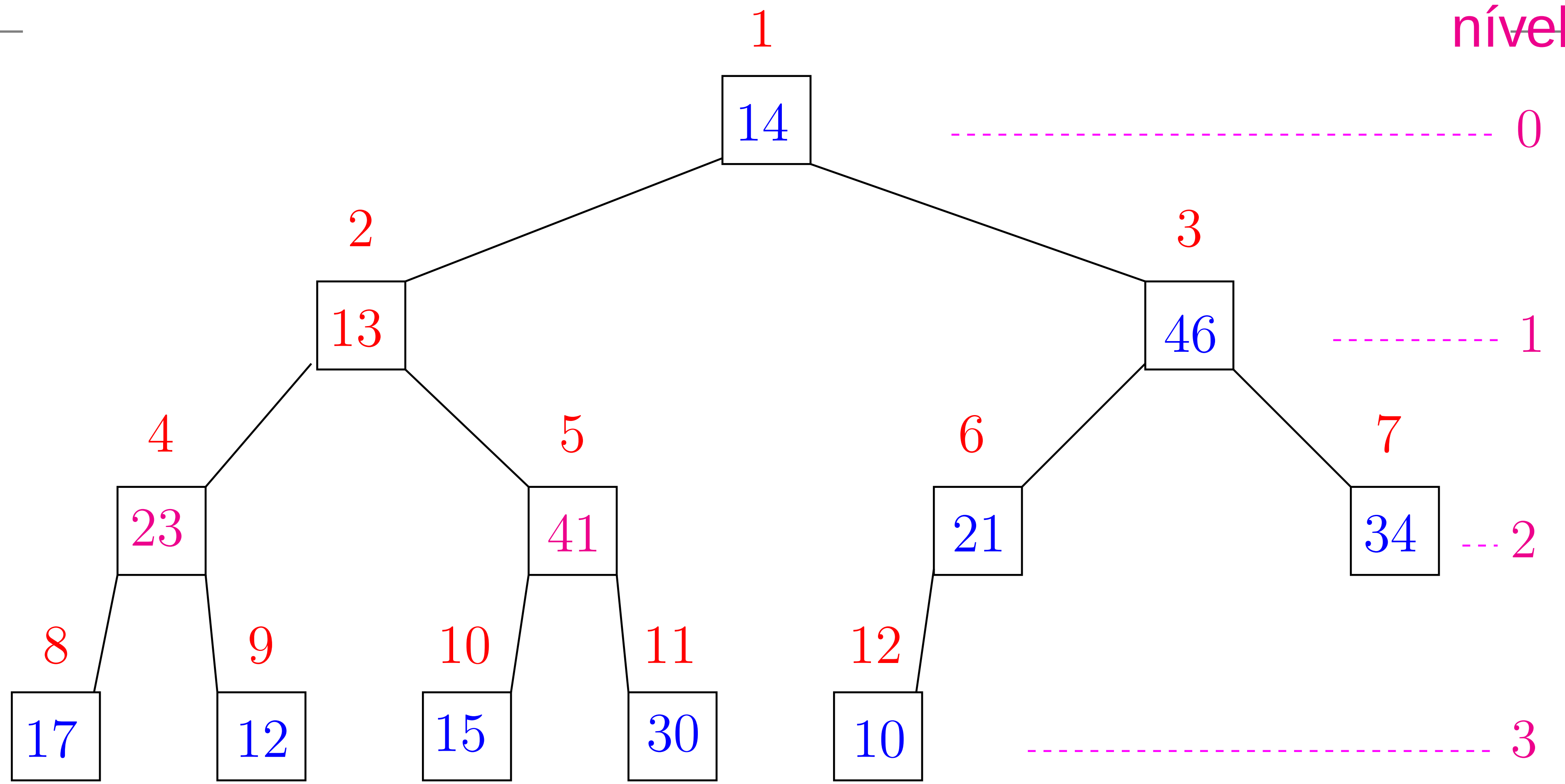
1	2	3	4	5	6	7	8	9	10	11	12
14	13	34	23	41	21	46	17	12	15	30	10

Construção de um heap



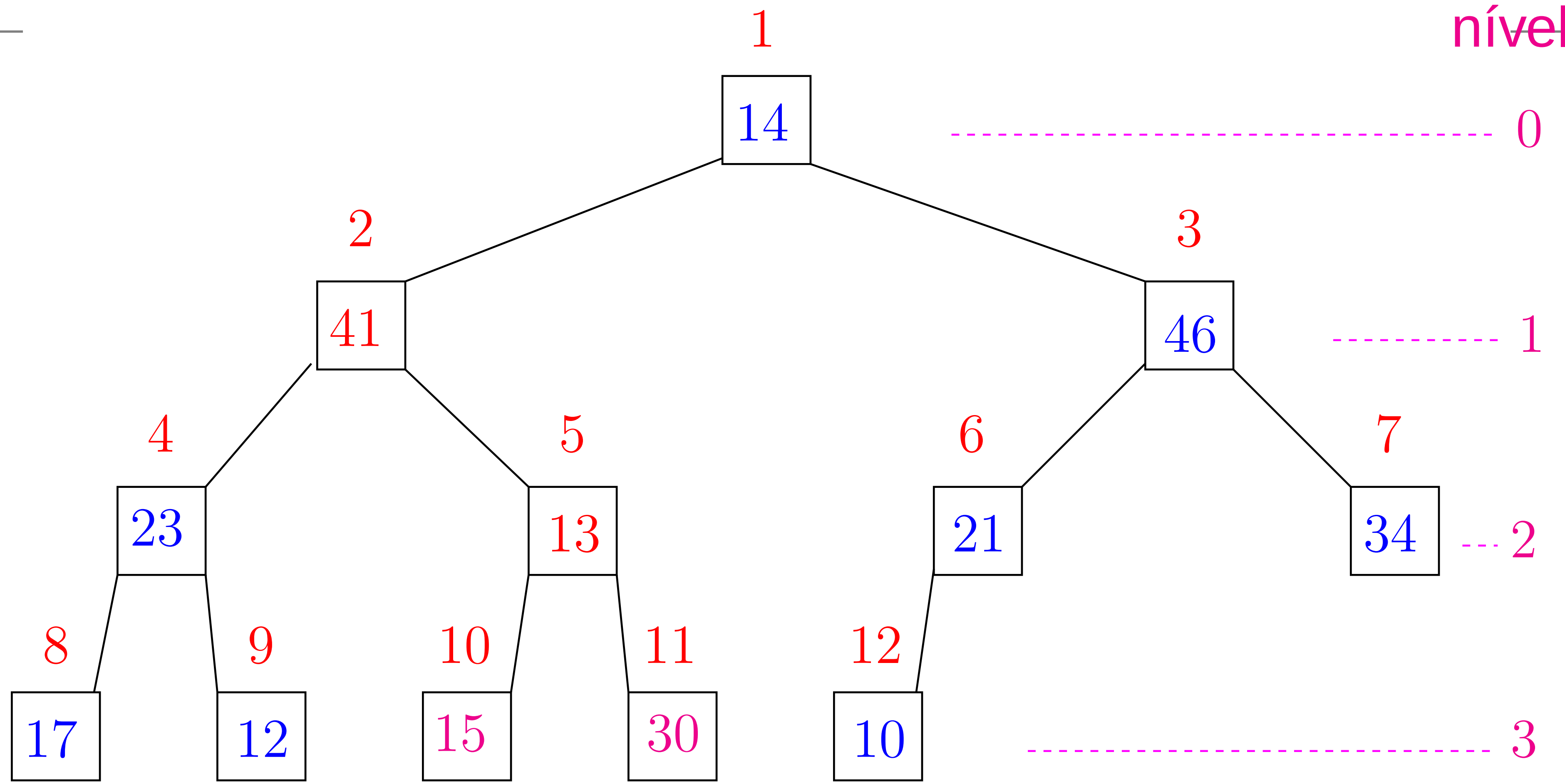
1	2	3	4	5	6	7	8	9	10	11	12
14	13	46	23	41	21	34	17	12	15	30	10

Construção de um heap



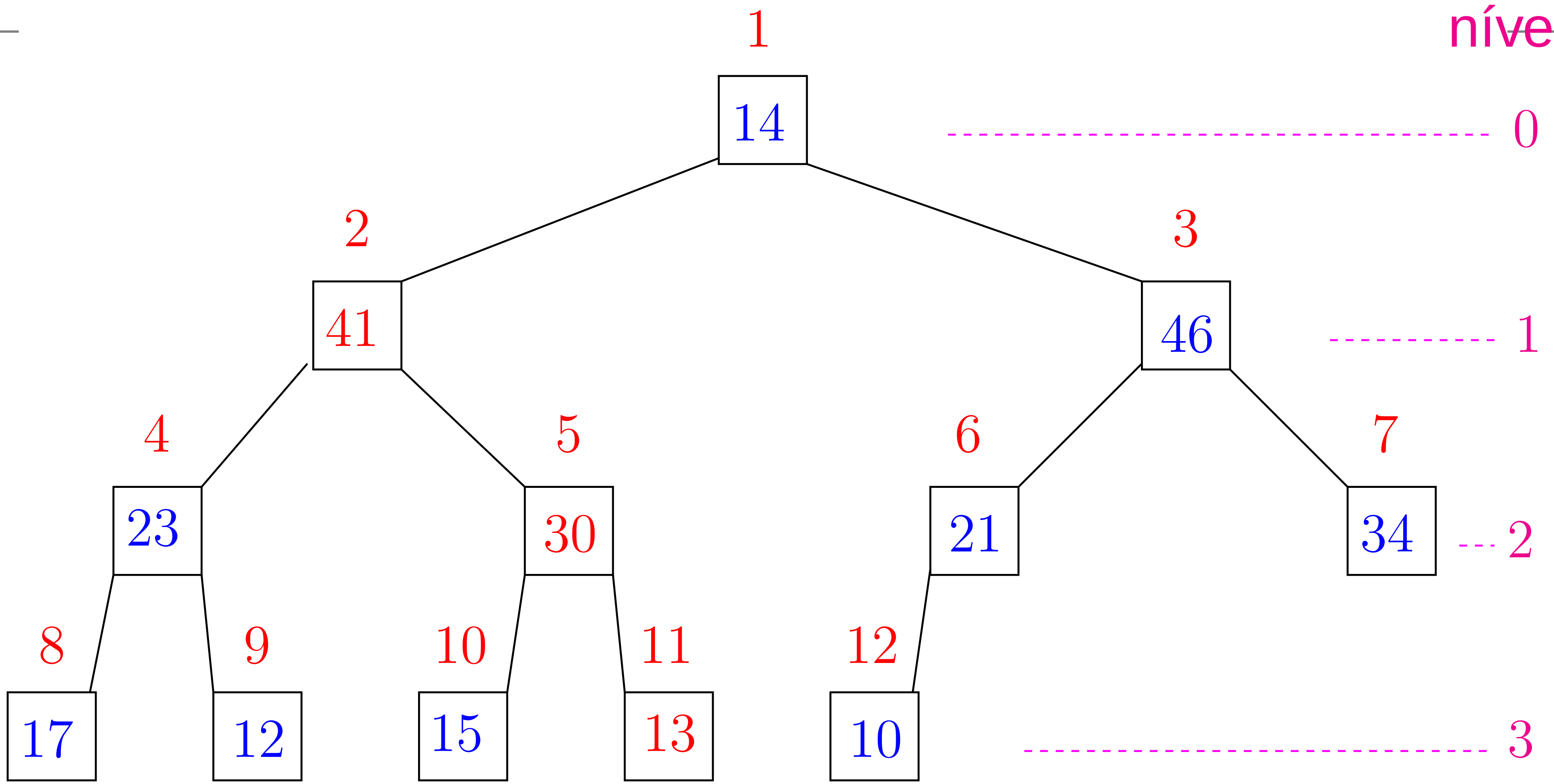
1	2	3	4	5	6	7	8	9	10	11	12
14	13	46	23	41	21	34	17	12	15	30	10

Construção de um heap



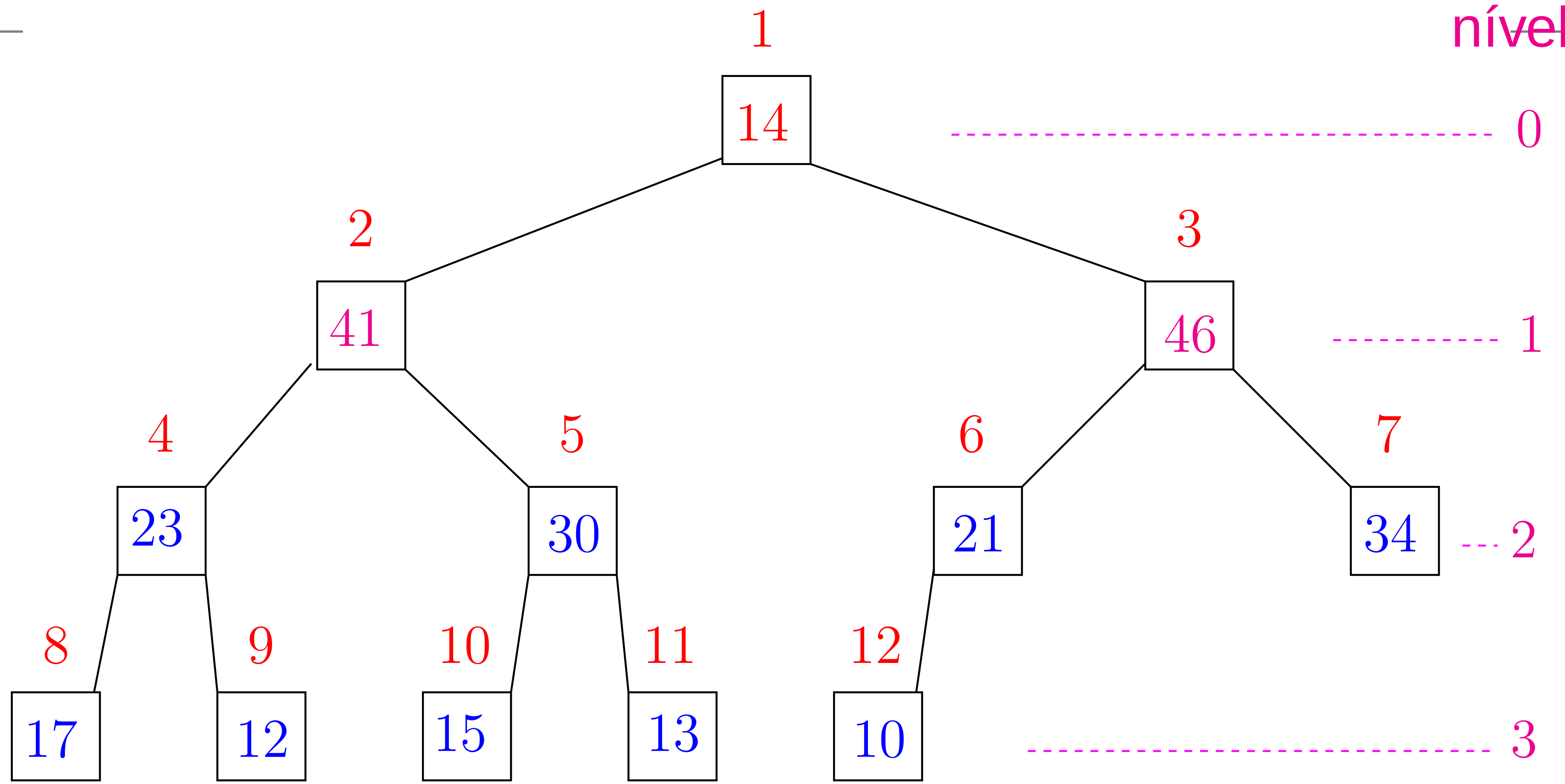
1	2	3	4	5	6	7	8	9	10	11	12
14	41	46	23	13	21	34	17	12	15	30	10

Construção de um heap



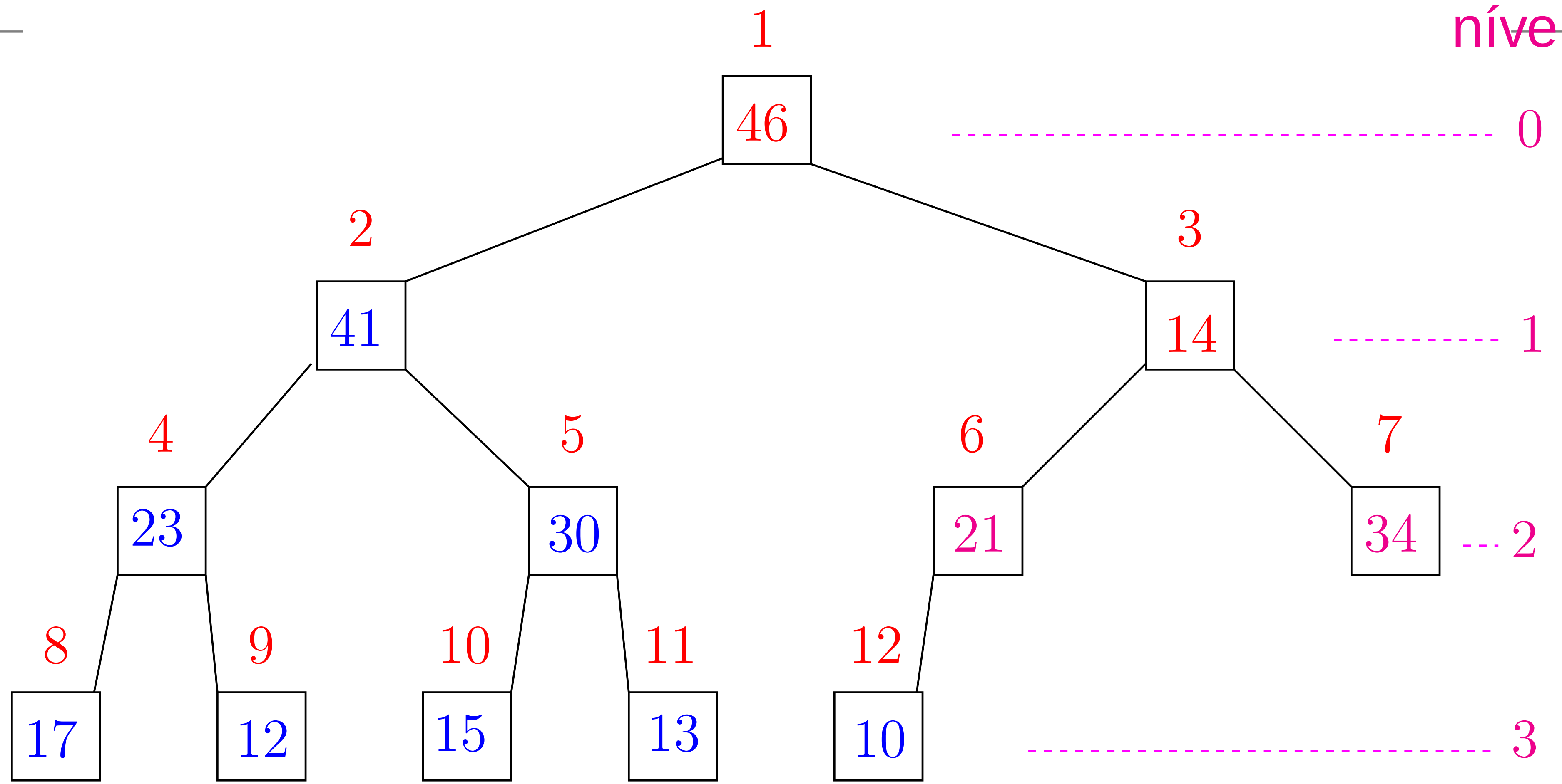
1	2	3	4	5	6	7	8	9	10	11	12
14	41	46	23	30	21	34	17	12	15	13	10

Construção de um heap



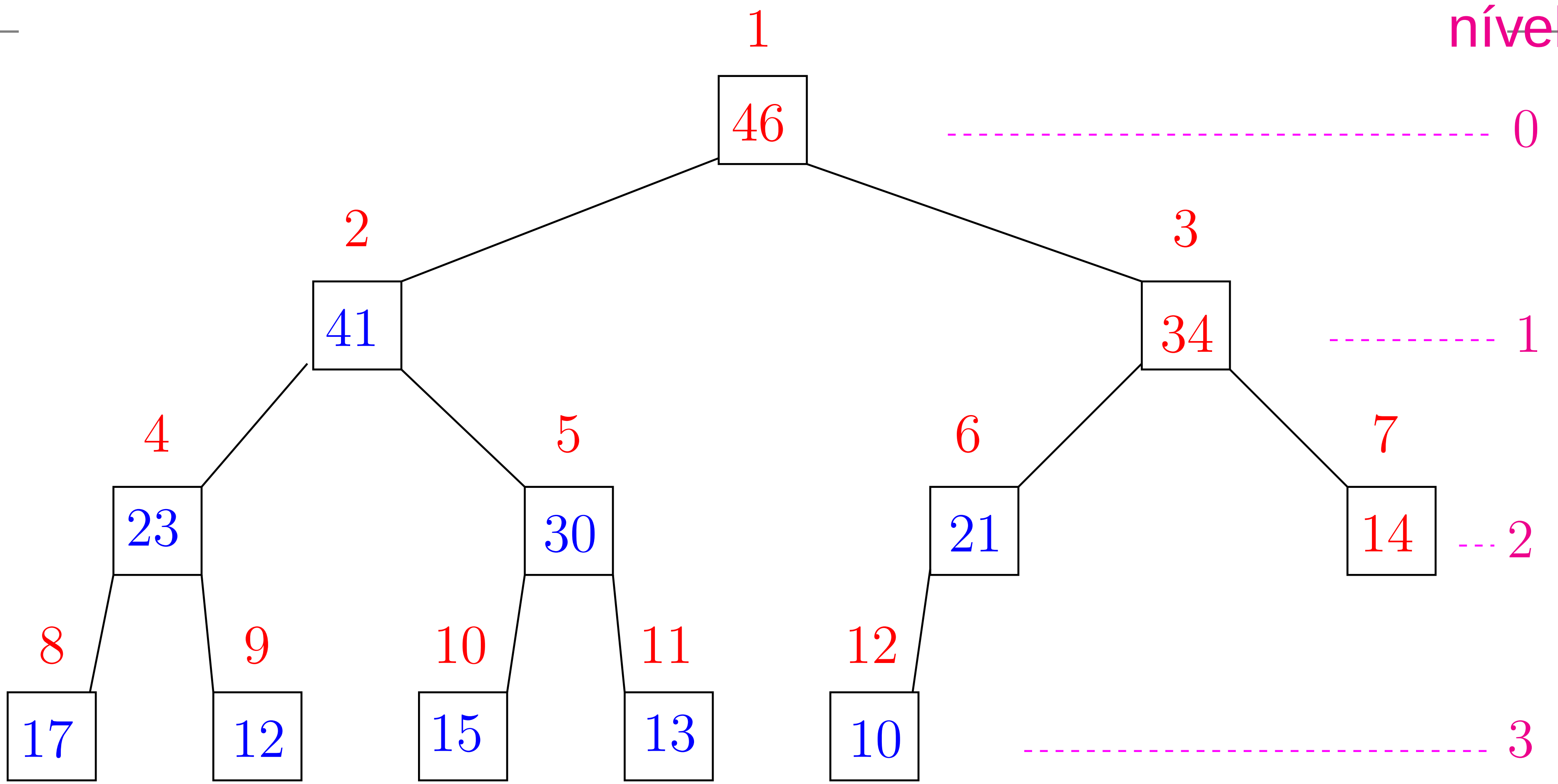
1	2	3	4	5	6	7	8	9	10	11	12
14	41	46	23	30	21	34	17	12	15	13	10

Construção de um heap



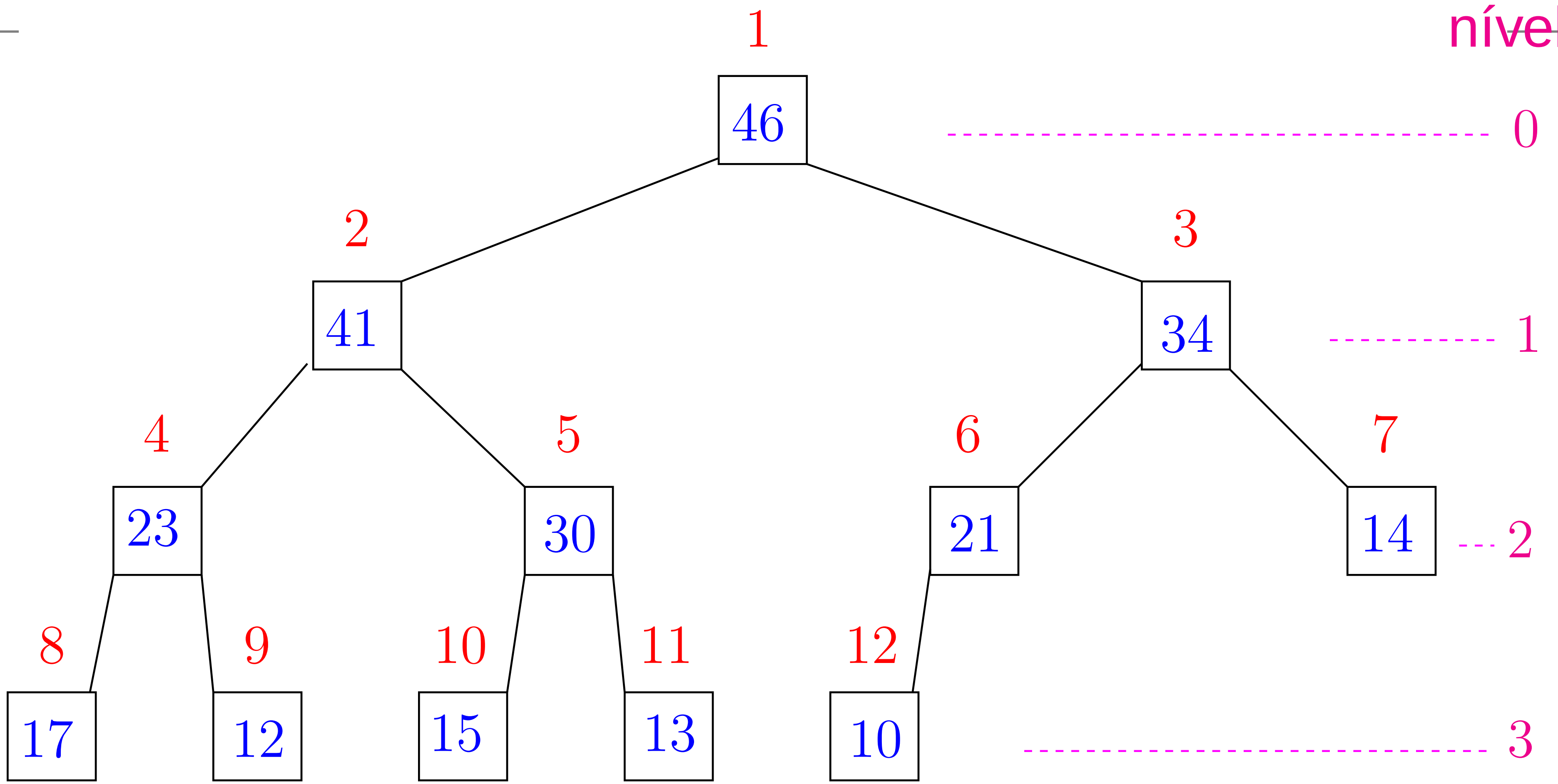
1	2	3	4	5	6	7	8	9	10	11	12
46	41	14	23	30	21	34	17	12	15	13	10

Construção de um heap



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

Construção de um heap



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

Construção de um heap

Recebe um vetor $A[1 \mathinner{..} n]$ 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)

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

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

CONSTRÓI-HEAP (A, n)
2 **para** $i \leftarrow \lfloor n/2 \rfloor$ **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

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

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

$T(n)$ é $\mathrm{O}(n)$

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

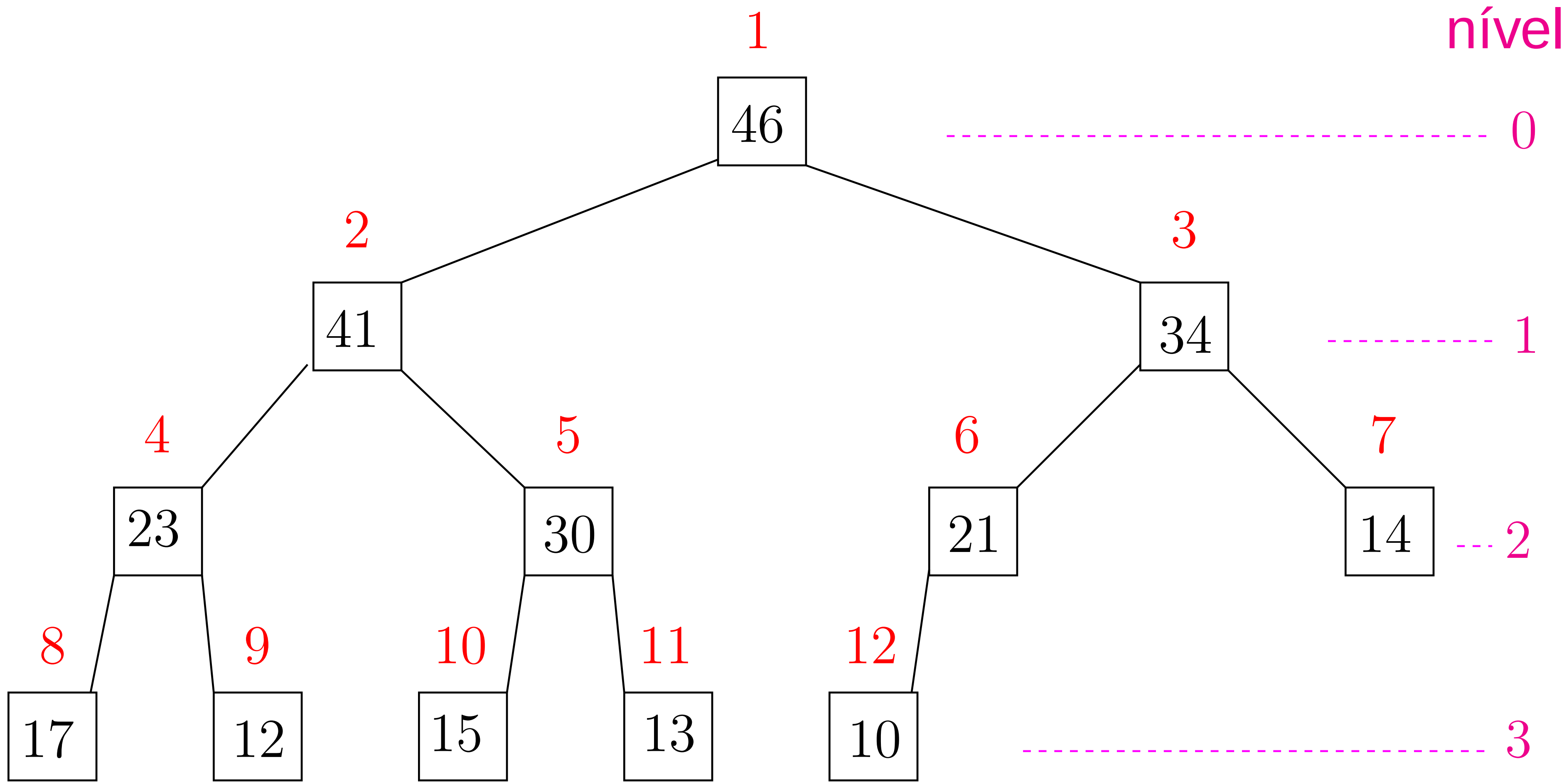
$$
\begin{aligned}
T(n) &= \sum_{h=1}^{\lfloor \lg n \rfloor} 2^{\lfloor \lg n \rfloor - h} h \\
&\leq \sum_{h=1}^{\lfloor \lg n \rfloor} \frac{n}{2^h}\, h \\
&\leq n \left(\frac{1}{2^1} + \frac{2}{2^2} + \frac{3}{2^3} + \cdots + \frac{\lfloor \lg n \rfloor}{2^{\lfloor \lg n \rfloor}} \right) \\
&< n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

$T(n)$ é $\mathrm{O}(n)$

Prova: O consumo de tempo de DESCE-HEAP (A, n, i) é $\mathrm{O}(h)$, onde h é a altura da árvore de raiz i. Logo,

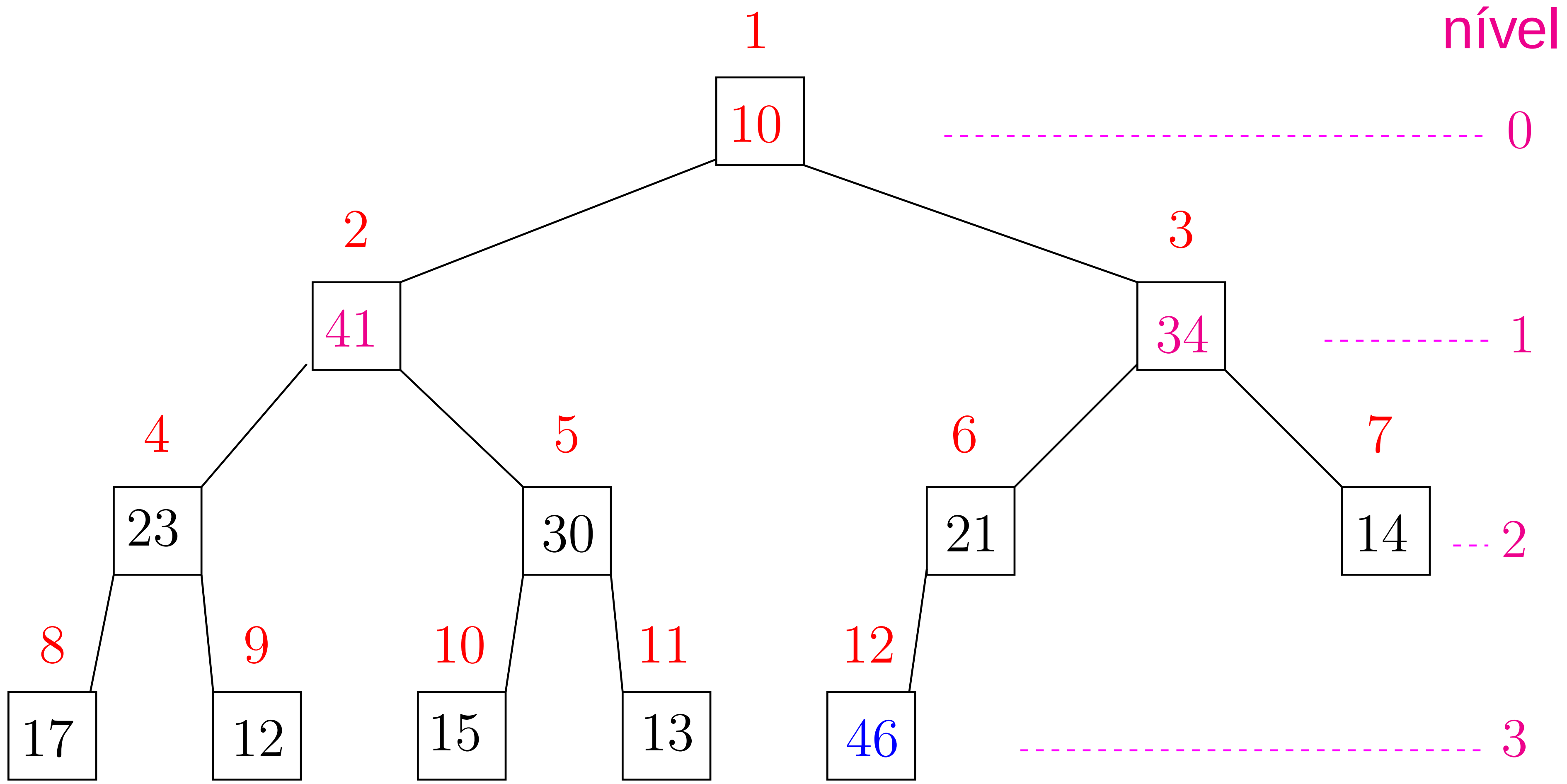
$$
\begin{aligned}
T(n) &= \sum_{h=0}^{\lfloor \lg n \rfloor} 2^{\lfloor \lg n \rfloor - h} \mathrm{O}(h) \\
&= \mathrm{O}\left(n \sum_{h=0}^{\lfloor \lg n \rfloor} \frac{h}{2^h}\right) \\
&= \mathrm{O}\left(n \sum_{h=0}^{\infty} \frac{h}{2^h}\right) \\
&= \mathrm{O}\left(n \frac{1/2}{(1-1/2)^2}\right)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

Heap sort



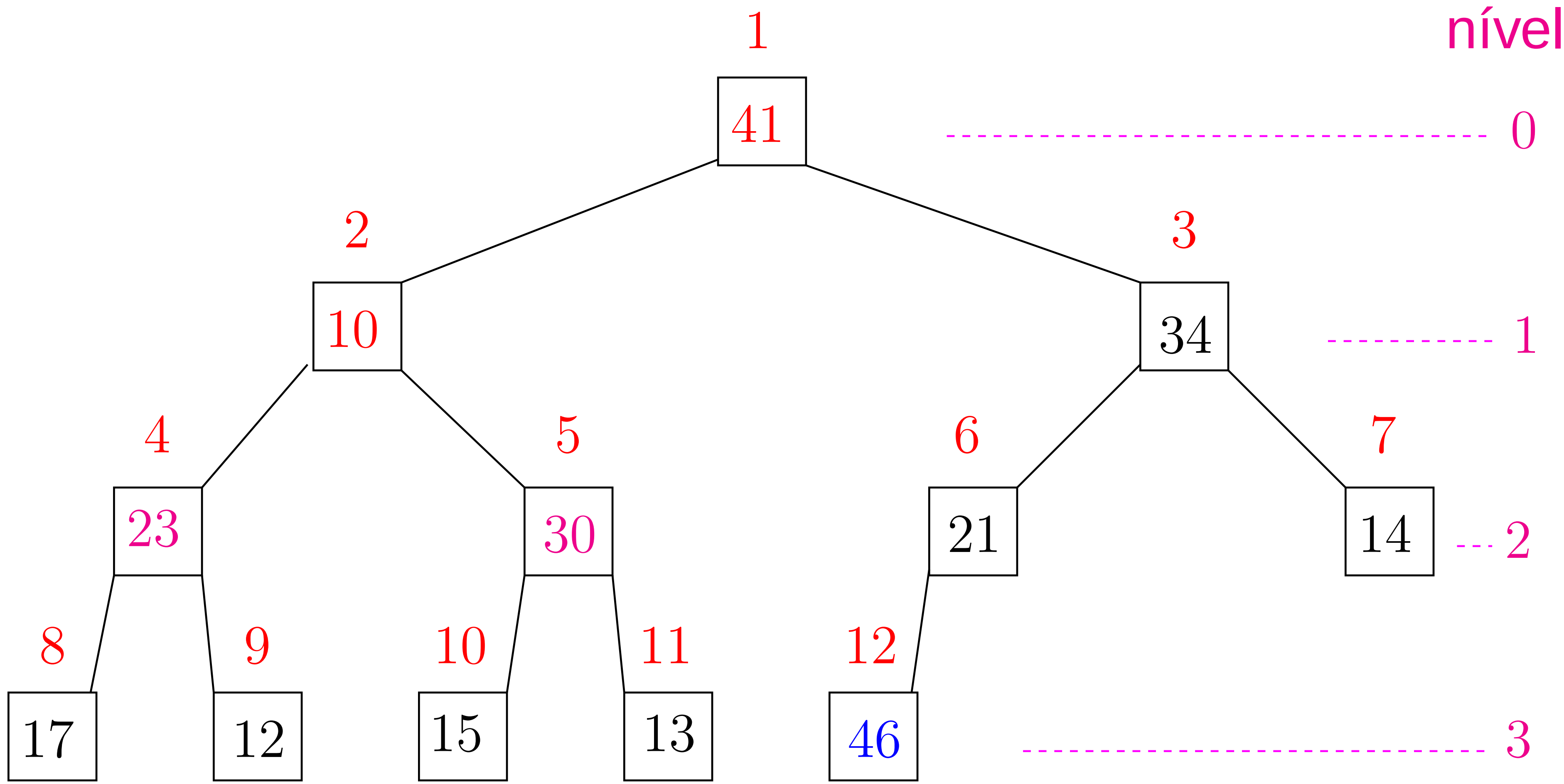
1	2	3	4	5	6	7	8	9	10	11	12
46	41	34	23	30	21	14	17	12	15	13	10

Heap sort



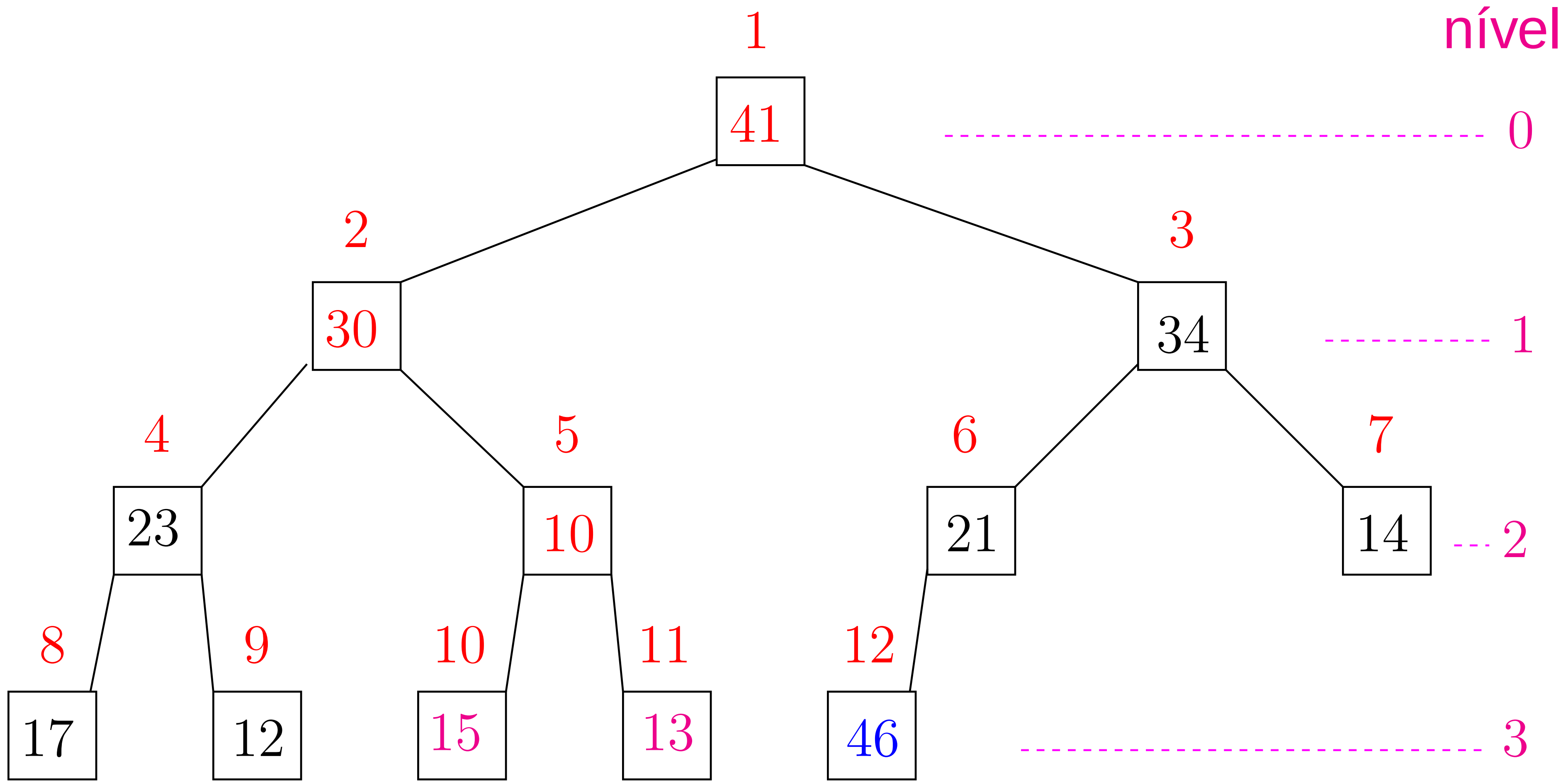
1	2	3	4	5	6	7	8	9	10	11	12
10	41	34	23	30	21	14	17	12	15	13	46

Heap sort



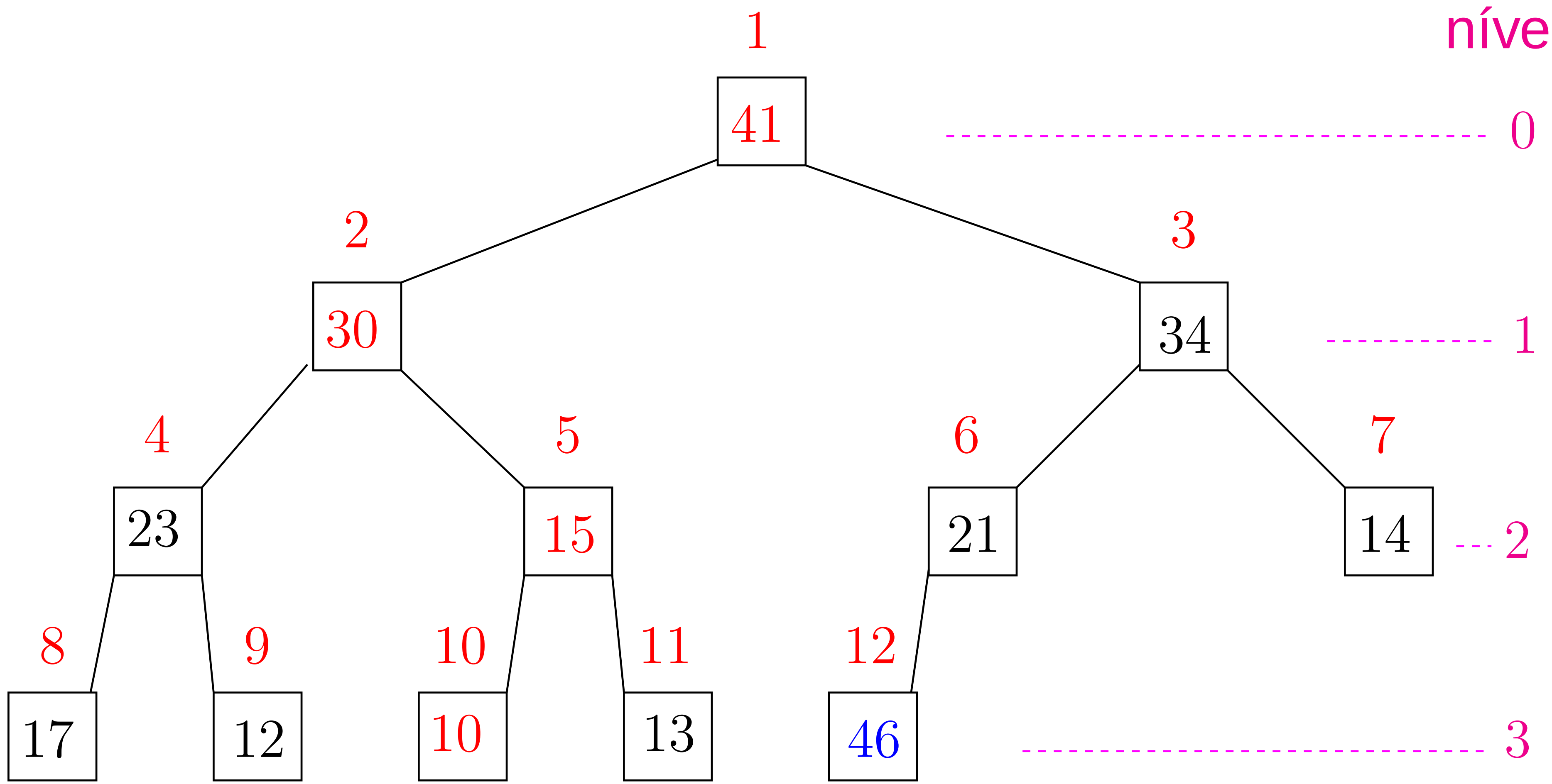
1	2	3	4	5	6	7	8	9	10	11	12
41	10	34	23	30	21	14	17	12	15	13	46

Heap sort



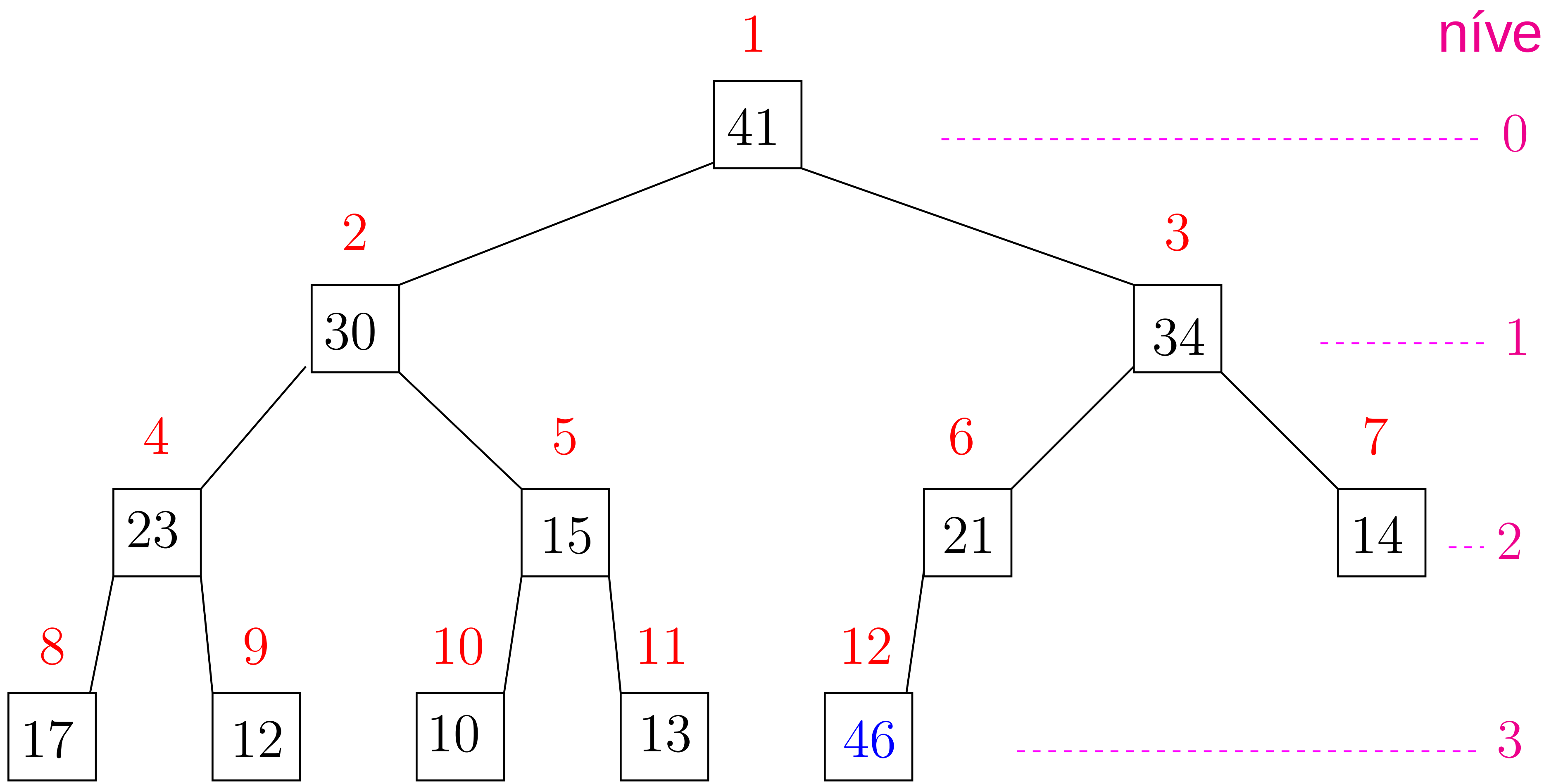
1	2	3	4	5	6	7	8	9	10	11	12
41	30	34	23	10	21	14	17	12	15	13	46

Heap sort



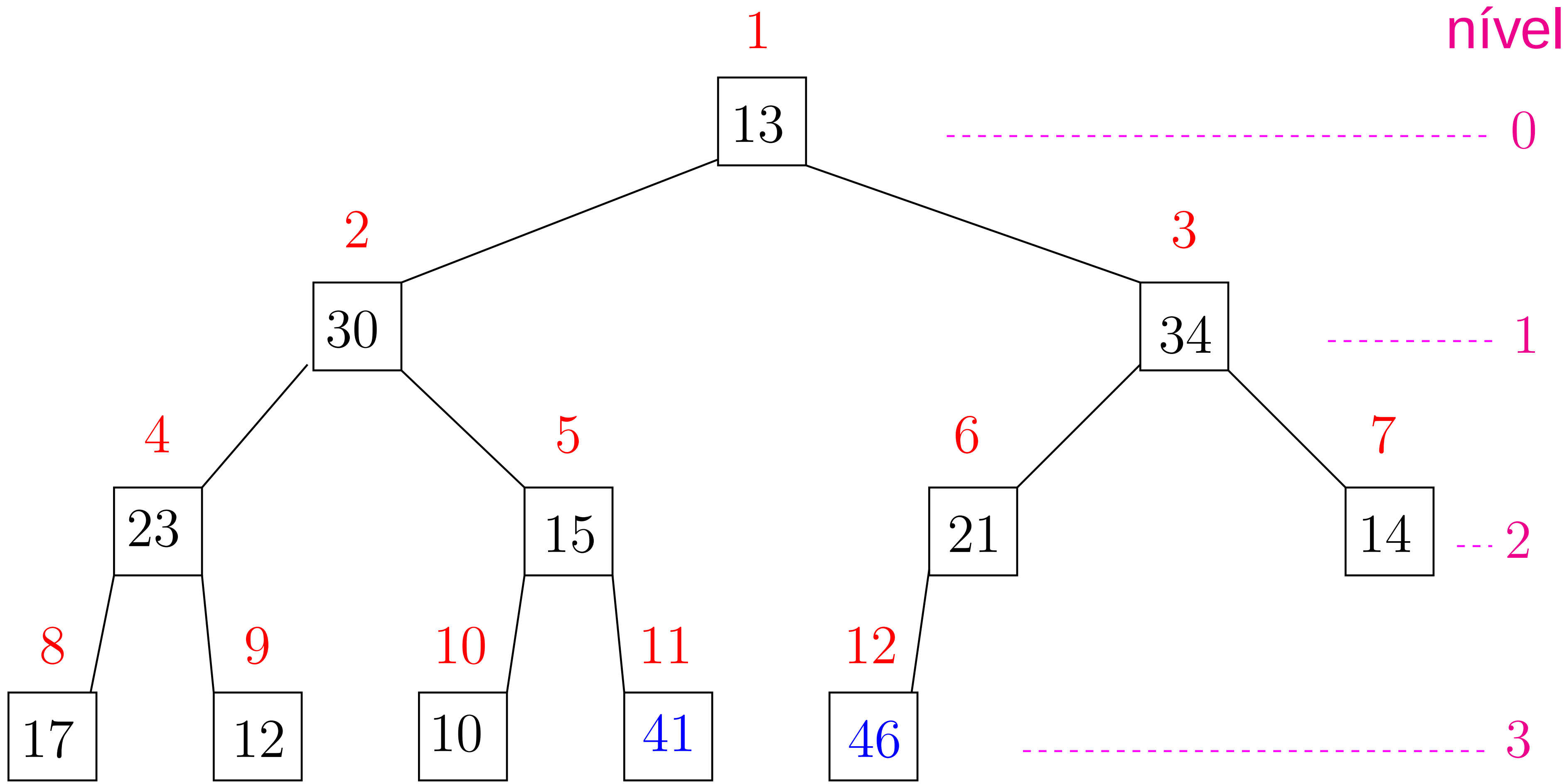
1	2	3	4	5	6	7	8	9	10	11	12
41	30	34	23	15	21	14	17	12	10	13	46

Heap sort



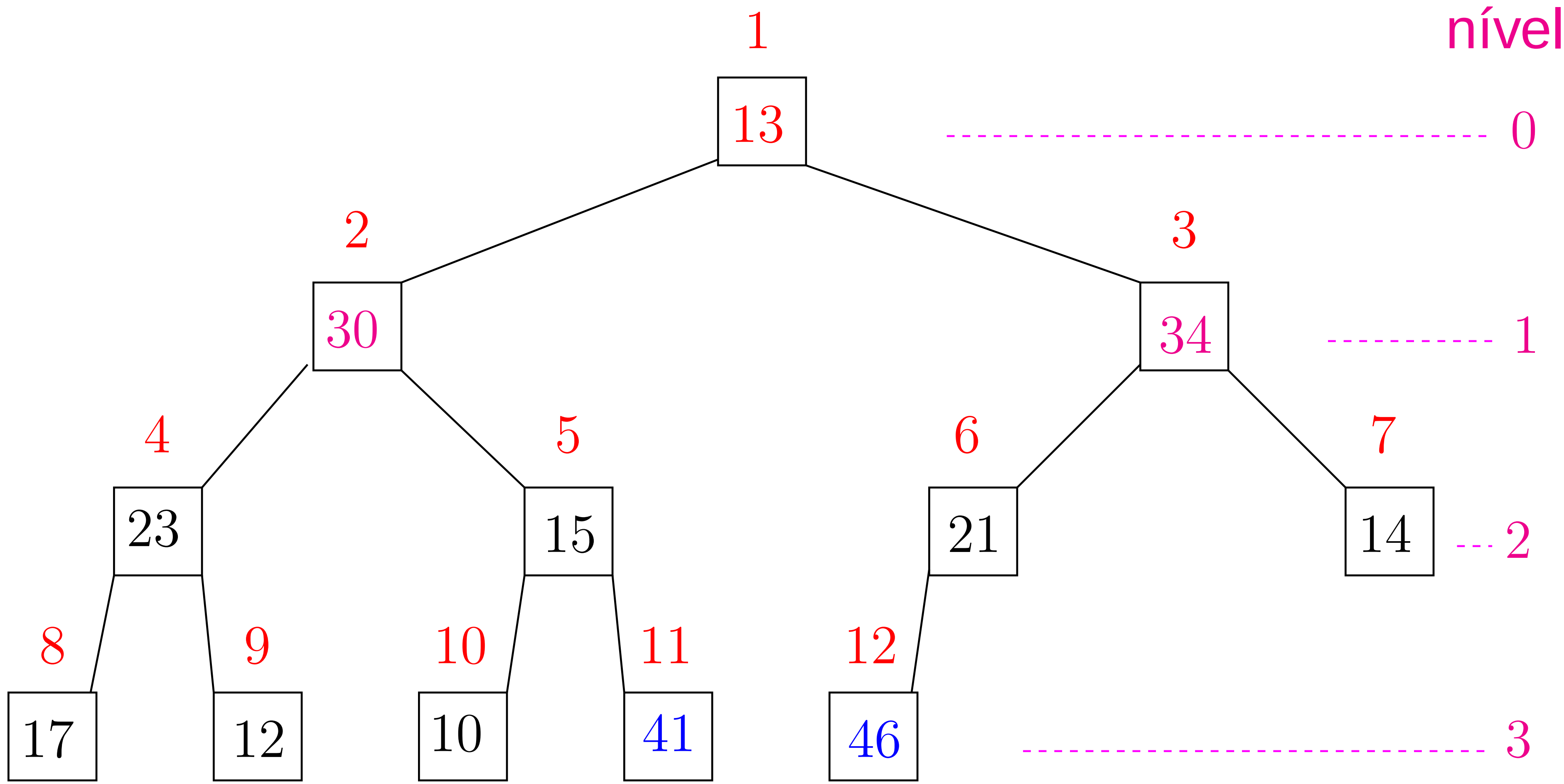
1	2	3	4	5	6	7	8	9	10	11	12
41	30	34	23	15	21	14	17	12	10	13	46

Heap sort



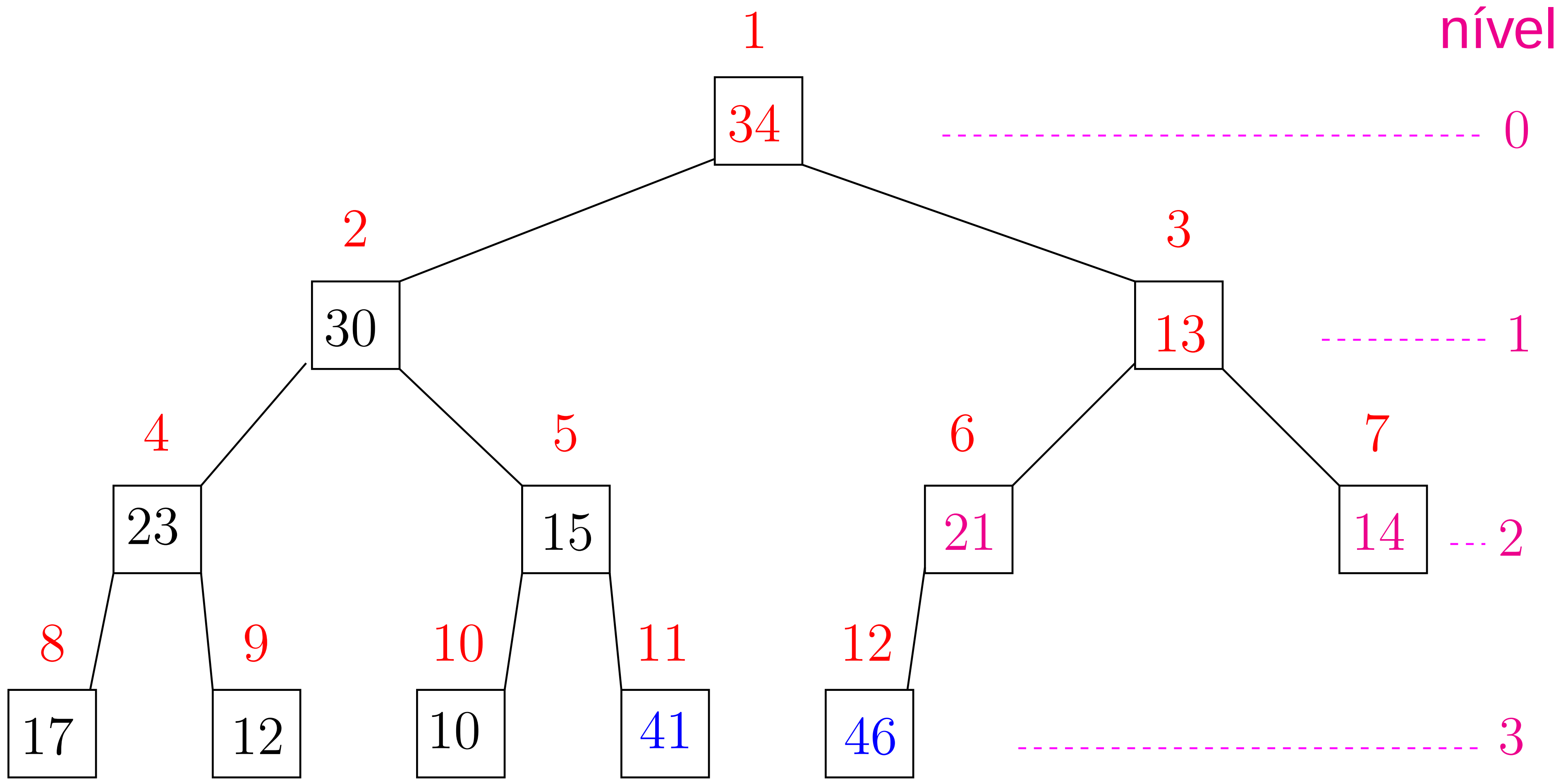
1	2	3	4	5	6	7	8	9	10	11	12
13	30	34	23	15	21	14	17	12	10	41	46

Heap sort



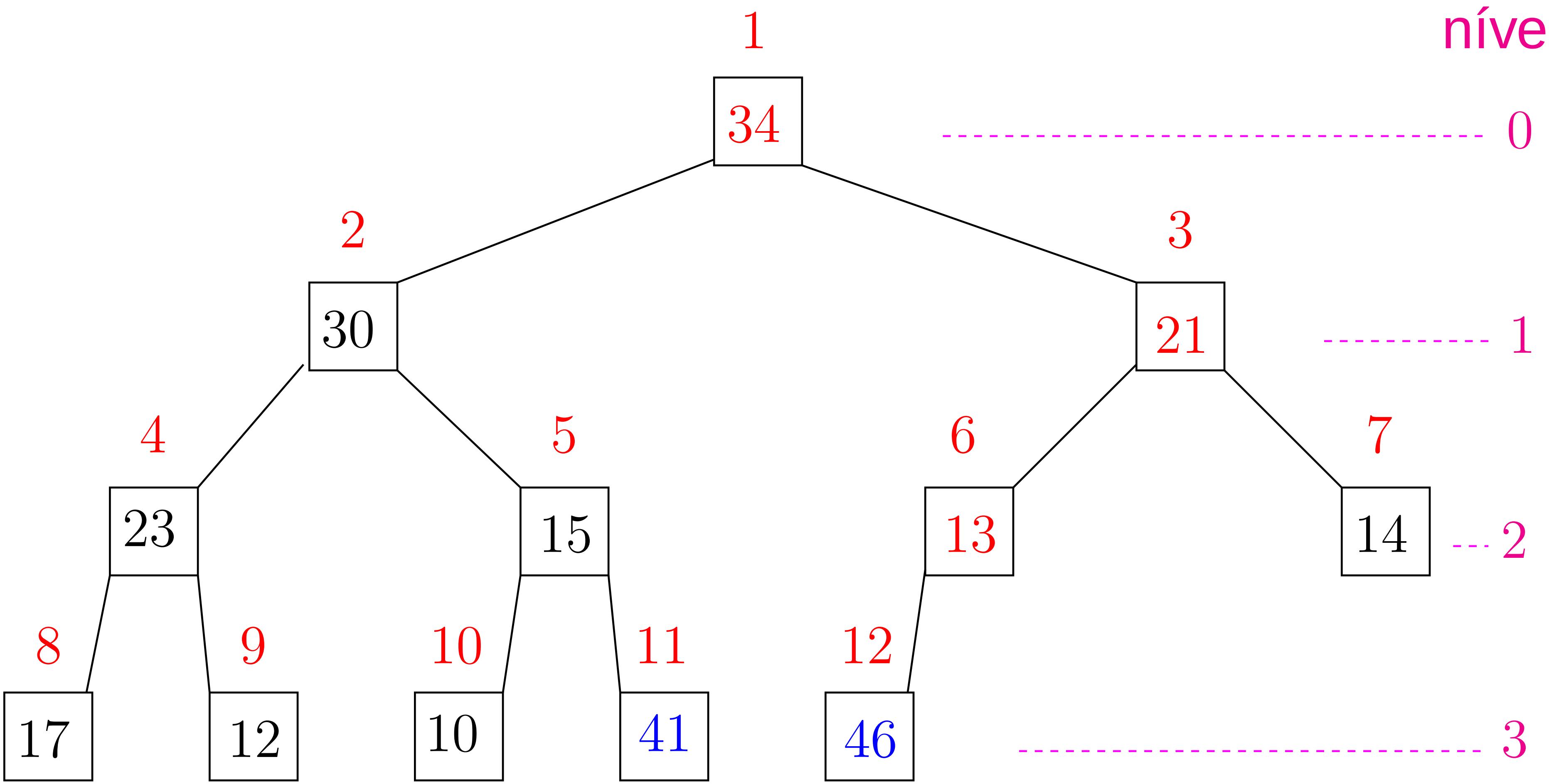
1	2	3	4	5	6	7	8	9	10	11	12
13	30	34	23	15	21	14	17	12	10	41	46

Heap sort



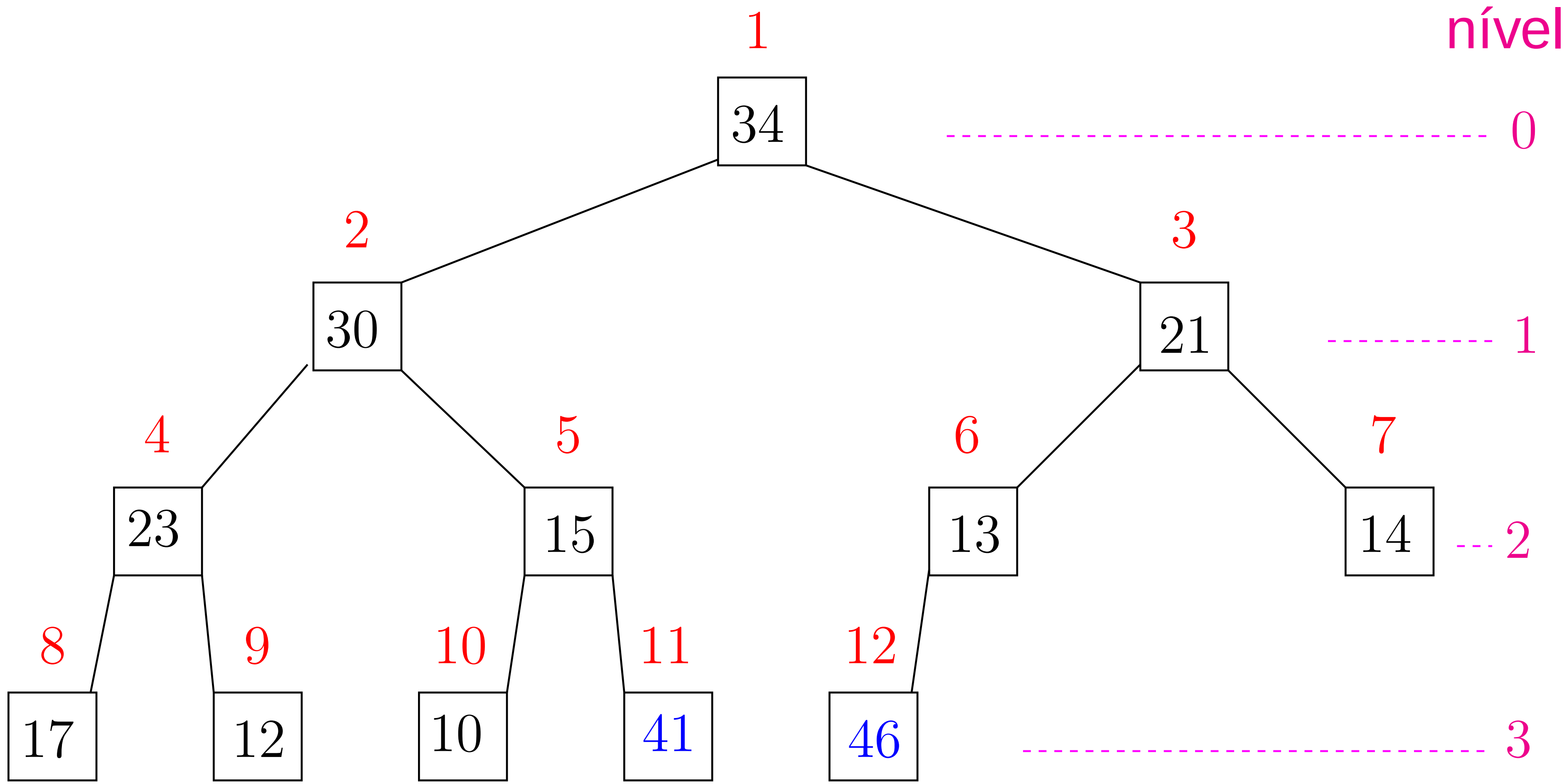
1	2	3	4	5	6	7	8	9	10	11	12
34	30	13	23	15	21	14	17	12	10	41	46

Heap sort



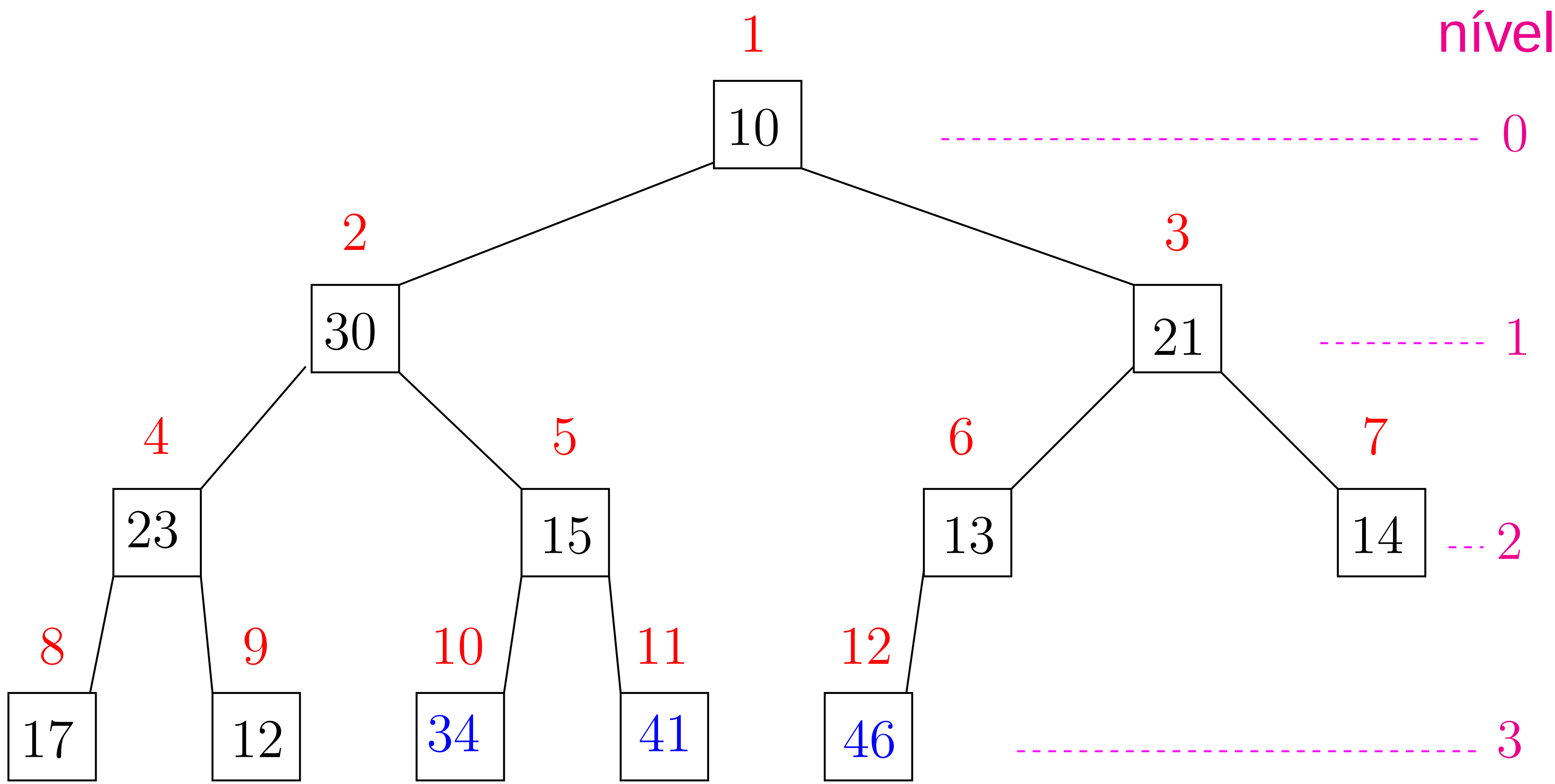
1	2	3	4	5	6	7	8	9	10	11	12
34	30	21	23	15	13	14	17	12	10	41	46

Heap sort



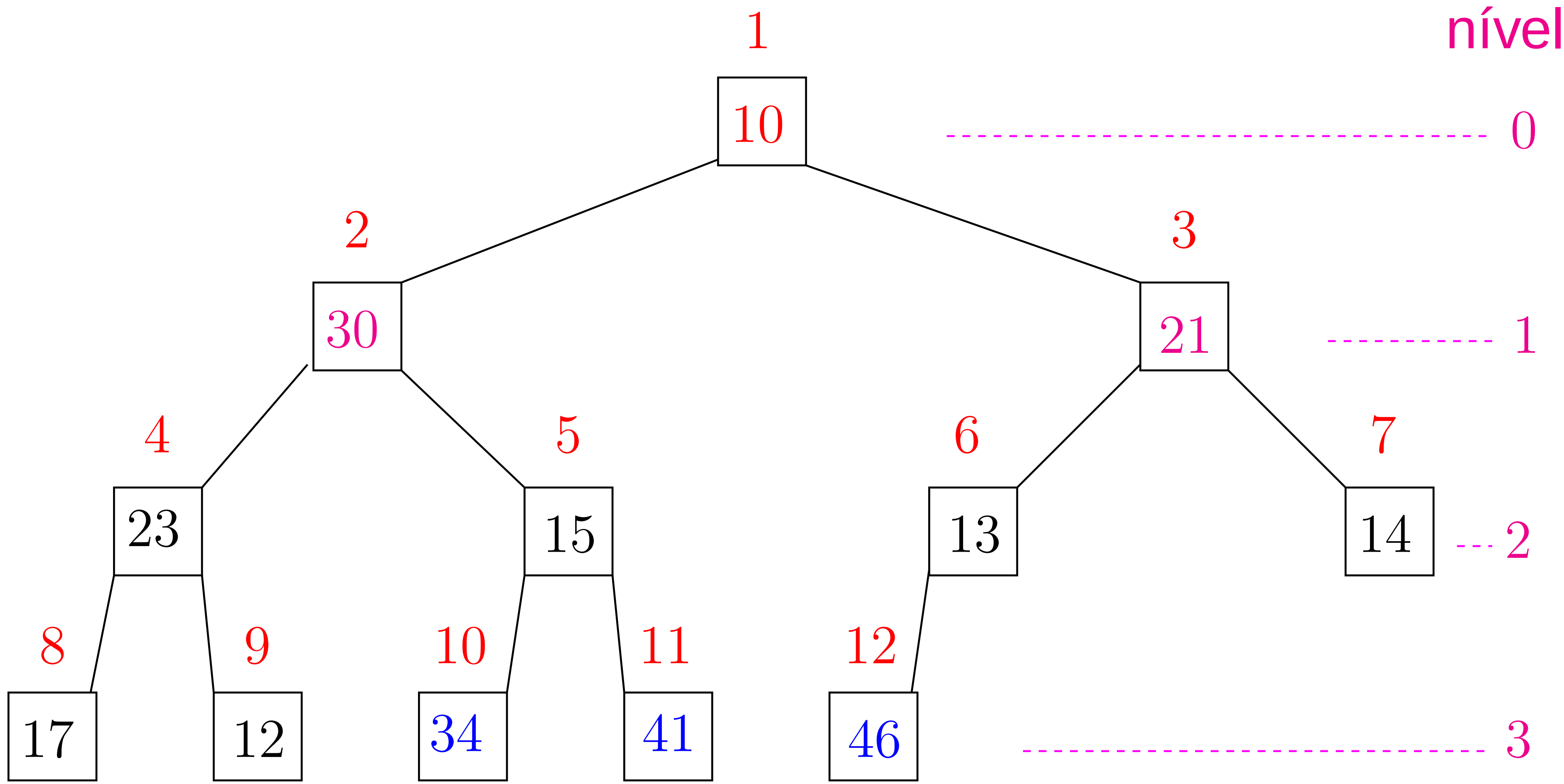
1	2	3	4	5	6	7	8	9	10	11	12
34	30	21	23	15	13	14	17	12	10	41	46

Heap sort



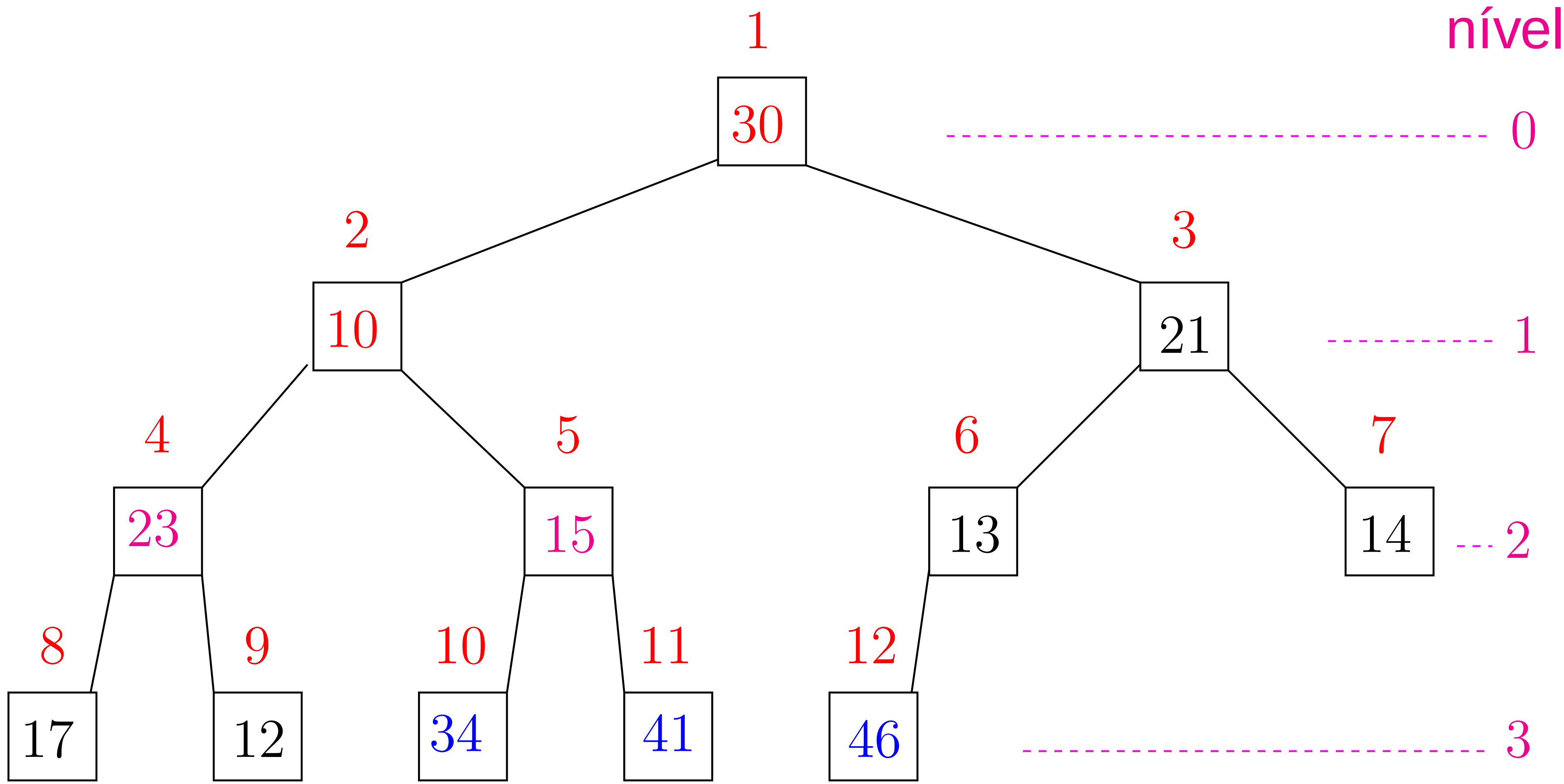
1	2	3	4	5	6	7	8	9	10	11	12
10	30	21	23	15	13	14	17	12	34	41	46

Heap sort



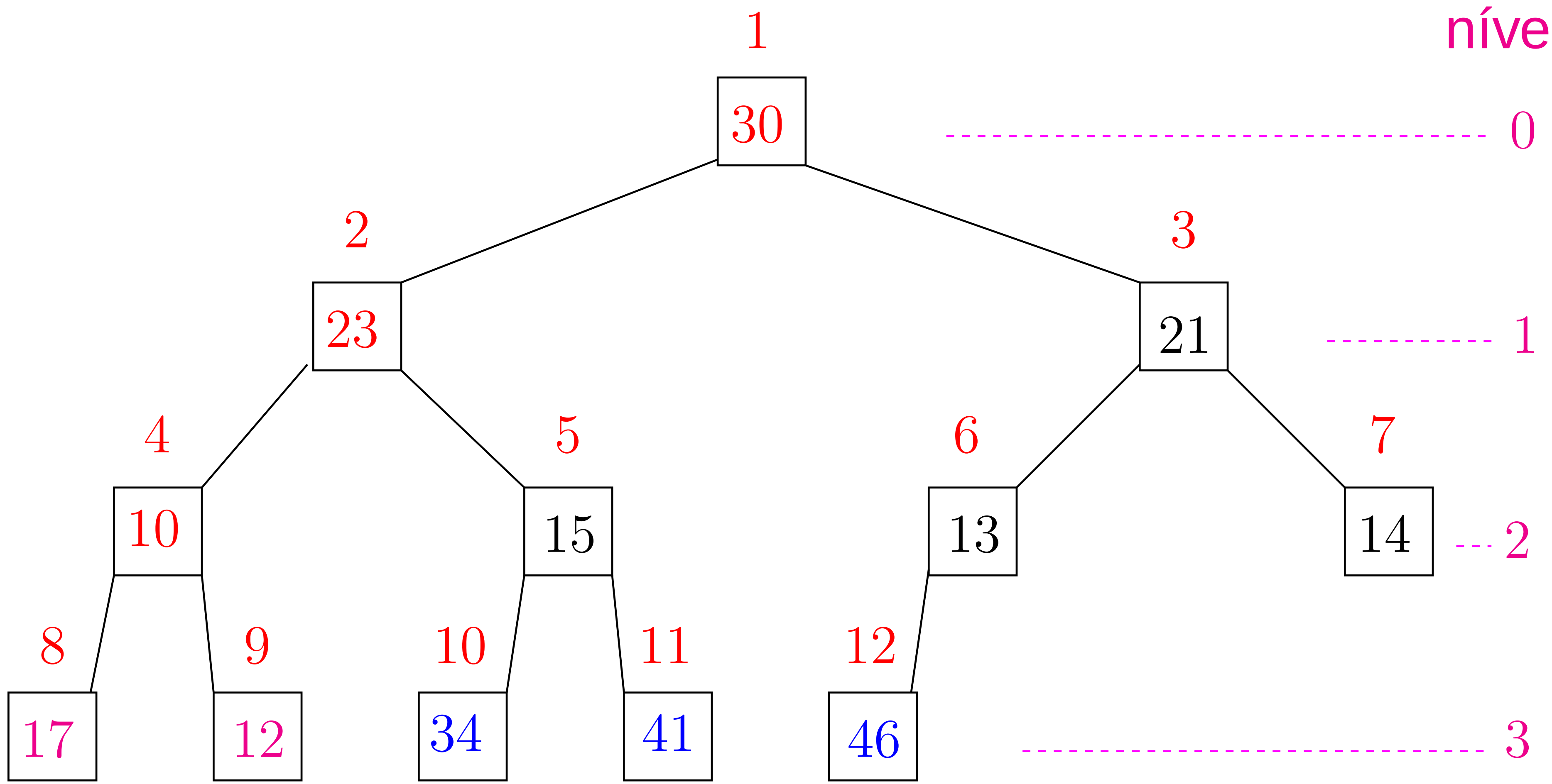
1	2	3	4	5	6	7	8	9	10	11	12
10	30	21	23	15	13	14	17	12	34	41	46

Heap sort



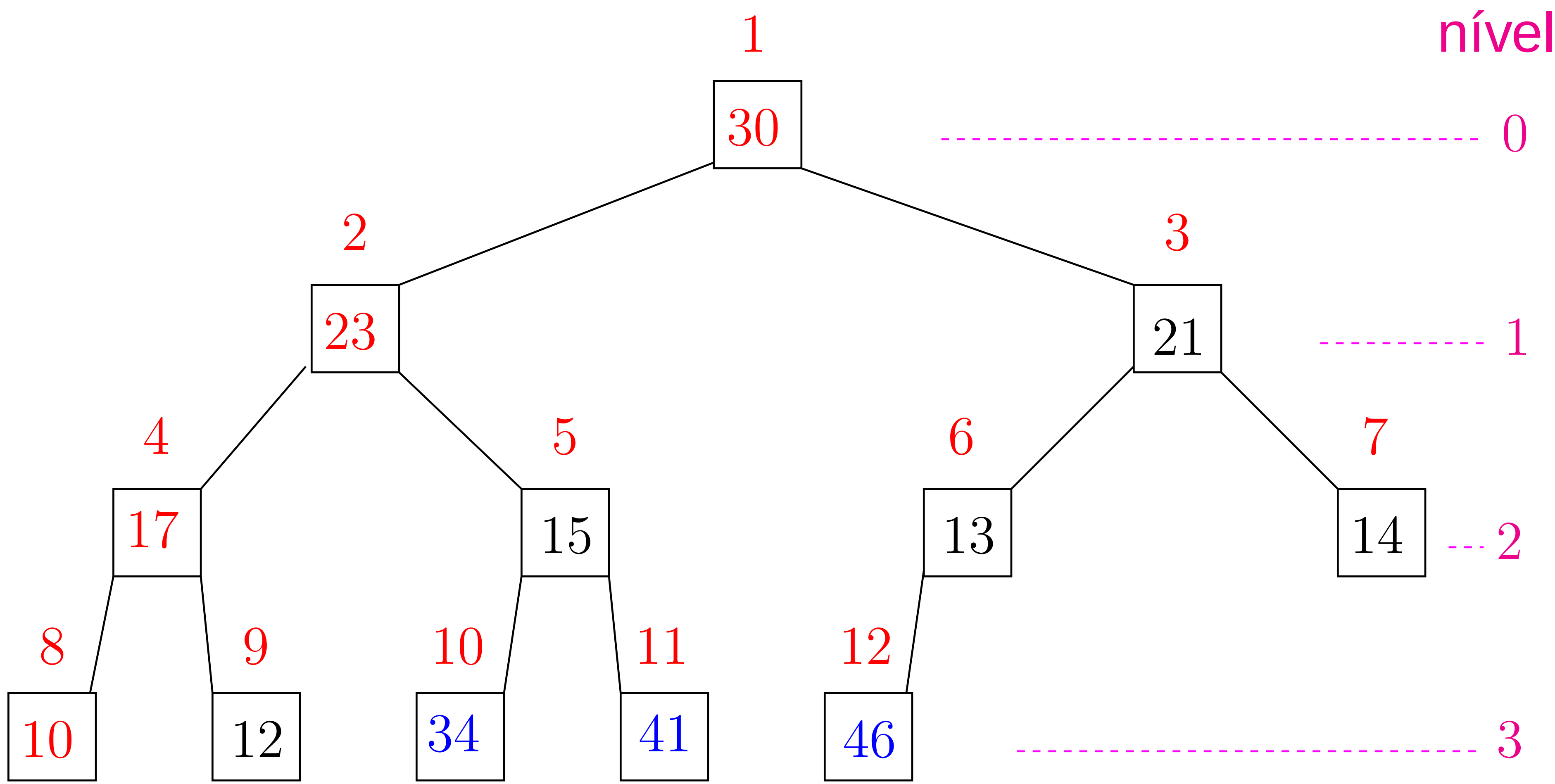
1	2	3	4	5	6	7	8	9	10	11	12
30	10	21	23	15	13	14	17	12	34	41	46

Heap sort



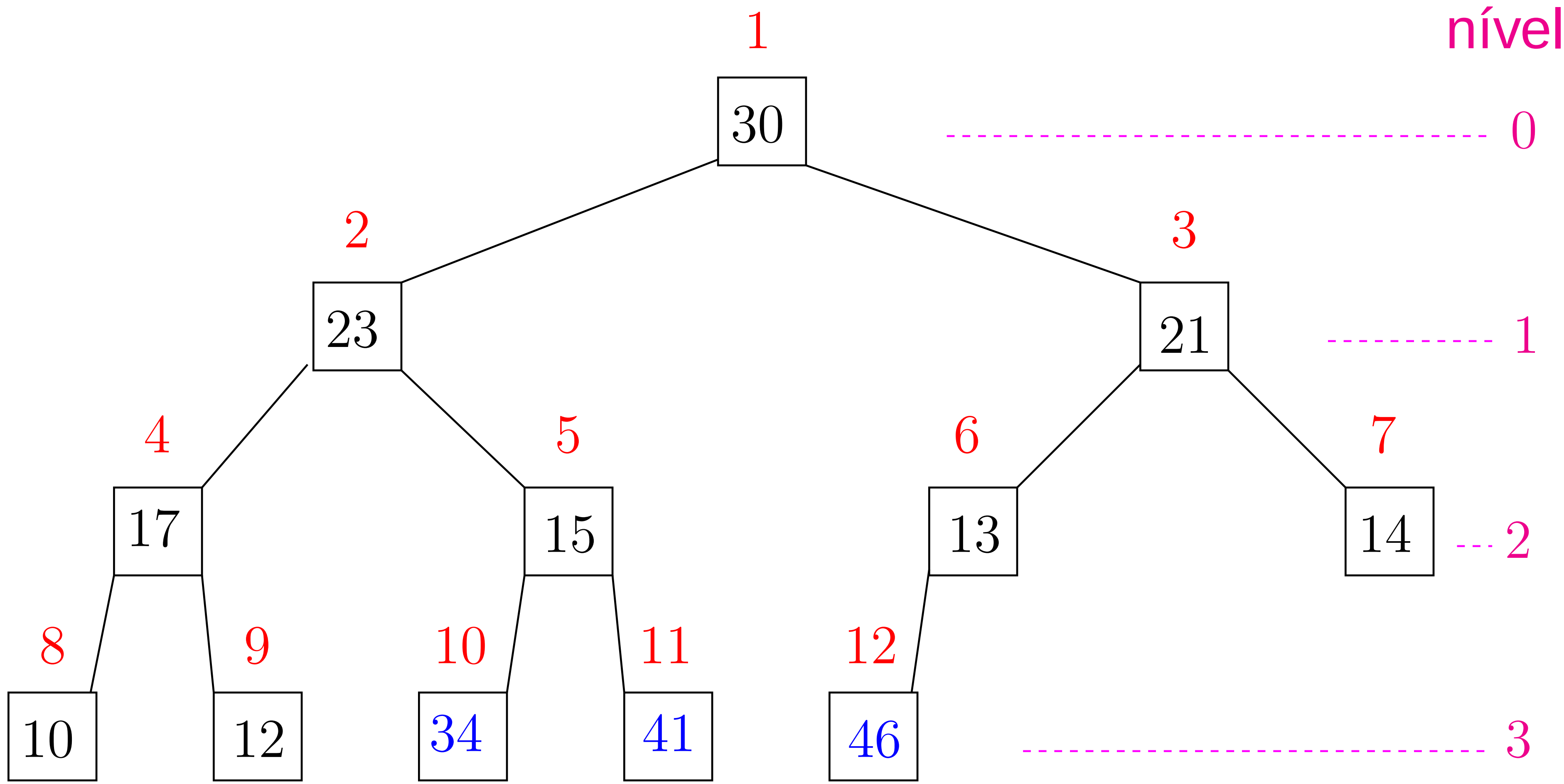
1	2	3	4	5	6	7	8	9	10	11	12
30	23	21	10	15	13	14	17	12	34	41	46

Heap sort



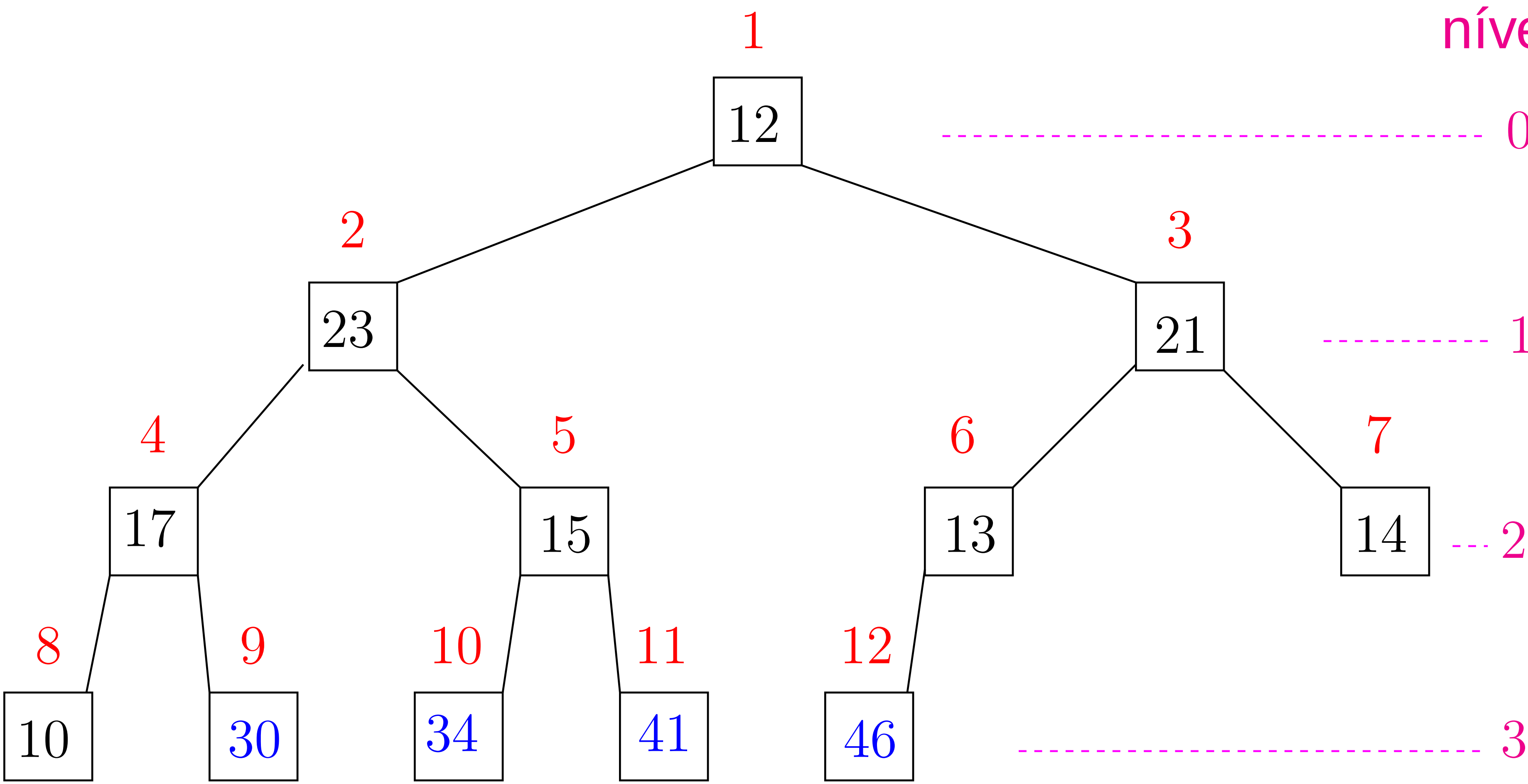
1	2	3	4	5	6	7	8	9	10	11	12
30	23	21	17	15	13	14	10	12	34	41	46

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



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

Heap sort

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

HEAPSORT (A, n)

0 CONSTRÓI-HEAP (A, n) $\triangleright$ pré-processamento
1 $m \leftarrow n$
2 **para** $i \leftarrow n$ **decrescendo até** 2 **faça**
3 $\quad A[1] \leftrightarrow A[i]$
4 $\quad m \leftarrow m - 1$
5 $\quad$ DESCE-HEAP $(A, m, 1)$

Relações invariantes: Na linha 2 vale que:

(i0) $A[m\,..\,n]$ é crescente;

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

(i2) $A[1\,..\,m]$ 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	$=$	$\Theta(n)$
5	$=$	$n\mathrm{O}(\lg n)$
total	$=$	$n\mathrm{O}(\lg n) + \Theta(n) = \mathrm{O}(n \lg n)$

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

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

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 execícios

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

$e \leftarrow 2i$

$d \leftarrow 2i + 1$

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

  então $A[i] \leftrightarrow A[e]$

    D-H (A, m, e)

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

  então $A[i] \leftrightarrow A[d]$

    D-H (A, m, d)