Análise de Algoritmos

CLRS 2.2 e 3.1
AU 3.3, 3.4 e 3.6

Essas transparências foram adaptadas das transparências do Prof. Paulo Feofiloff e do Prof. José Coelho de Pina.

Notação O

Intuitivamente...

$\mathrm{O}(f(n))$ $\approx$ funções que não crescem mais rápido que $f(n)$

$\approx$ funções menores ou iguais a um múltiplo de $f(n)$

n^2 $(3/2)n^2$ $9999n^2$ $n^2/1000$ etc.

crescem todas com a mesma velocidade

Notação O

Intuitivamente...

$\mathrm{O}(f(n))$ $\approx$ funções que não crescem mais rápido que $f(n)$

$\approx$ funções menores ou iguais a um múltiplo de $f(n)$

n^2 $(3/2)n^2$ $9999n^2$ $n^2/1000$ etc.

crescem todas com a mesma velocidade

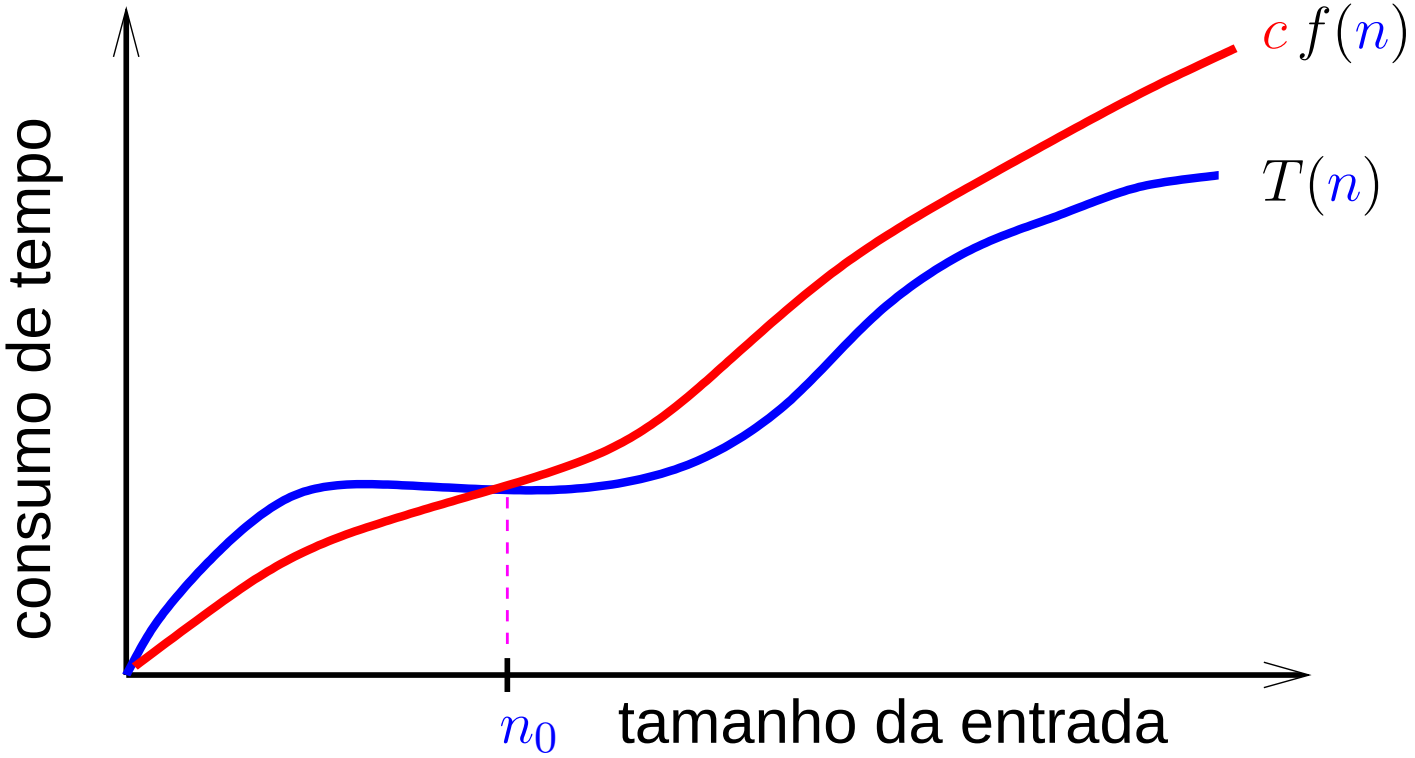
- $n^2 + 99n$ é $\mathrm{O}(n^2)$
- $33n^2$ é $\mathrm{O}(n^2)$
- $9n + 2$ é $\mathrm{O}(n^2)$
- $0{,}00001n^3 - 200n^2$ não é $\mathrm{O}(n^2)$

Definição

Sejam $T(n)$ e $f(n)$ funções dos inteiros nos reais. Dizemos que $T(n)$ é $\mathrm{O}(f(n))$ se existem constantes positivas c e n_0 tais que

$$T(n) \leq c\, f(n)$$

para todo $n \geq n_0$.

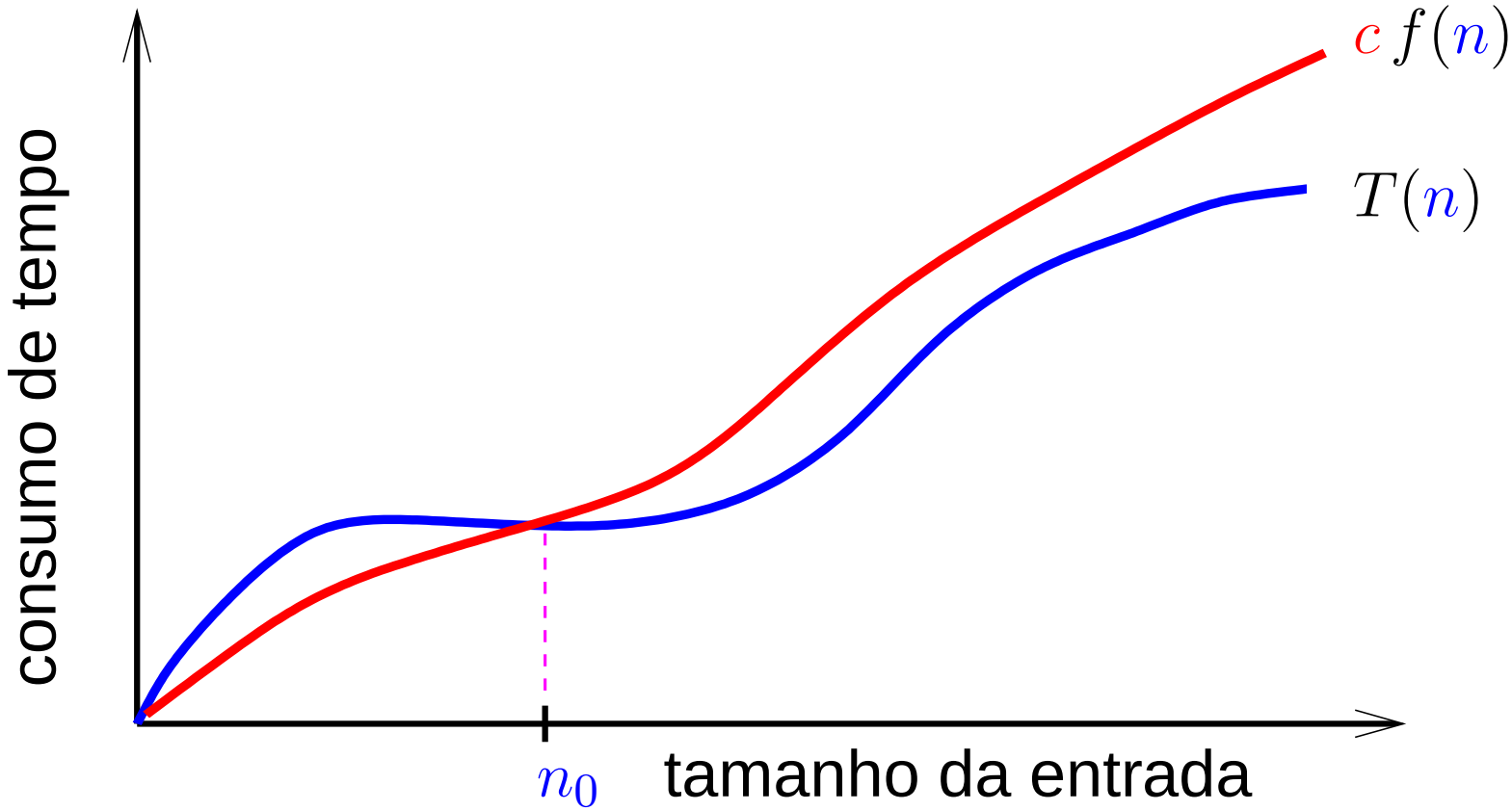


Mais informal

$T(n)$ é $\mathrm{O}(f(n))$ se existe $c > 0$ tal que

$$T(n) \leq c\, f(n)$$

para todo n suficientemente GRANDE.



Exemplo

$20n^3 + 10n \log n + 5$ é $\mathrm{O}(n^3)$.

Exemplo

$20n^3 + 10n \log n + 5$ é $\mathrm{O}(n^3)$.

Prova: Para $n \geq 1$, tem-se que

$$20n^3 + 10n \lg n + 5 \leq 20n^3 + 10n^3 + 5n^3 = 35\, n^3.$$

Exemplo

$20n^3 + 10n \log n + 5$ é $\mathrm{O}(n^3)$.

Prova: Para $n \geq 1$, tem-se que

$$20n^3 + 10n \lg n + 5 \leq 20n^3 + 10n^3 + 5n^3 = 35\, n^3.$$

Outra prova: Para $n \geq 10$, tem-se que

$$20n^3 + 10n \lg n + 5 \leq 20n^3 + n\, n \lg n + n \leq 20n^3 + n^3 + n^3 = 22\, n^3.$$

Uso da notação O

$\mathrm{O}(f(n)) = \{T(n)$: existem c e n_0 tq $T(n) \leq cf(n), n \geq n_0\}$

"$T(n)$ é $\mathrm{O}(f(n))$" deve ser entendido como "$T(n) \in \mathrm{O}(f(n))$".

"$T(n) = \mathrm{O}(f(n))$" deve ser entendido como "$T(n) \in \mathrm{O}(f(n))$".

"$T(n) \leq \mathrm{O}(f(n))$" é feio.

"$T(n) \geq \mathrm{O}(f(n))$" não faz sentido!

"$T(n)$ é $g(n) + \mathrm{O}(f(n))$" significa que existe constantes positivas c e n_0 tais que

$$T(n) \leq g(n) + c\, f(n)$$

para todo $n \geq n_0$.

Nomes de classes O

classe	nome
$\mathrm{O}(1)$	constante
$\mathrm{O}(\lg n)$	logarítmica
$\mathrm{O}(n)$	linear
$\mathrm{O}(n \lg n)$	$n \log n$
$\mathrm{O}(n^2)$	quadrática
$\mathrm{O}(n^3)$	cúbica
$\mathrm{O}(n^k)$ com $k \geq 1$	polinomial
$\mathrm{O}(2^n)$	exponencial
$\mathrm{O}(a^n)$ com $a > 1$	exponencial

Exemplo: número de inversões

Problema: Dada uma permutação $p[1\,..\,n]$, determinar o número de inversões em p.

Uma inversão é um par (i, j) de índices de p tal que $i < j$ e $p[i] > p[j]$.

Entrada:

	1	2	3	4	5	6	7	8	9
p	2	4	1	9	5	3	8	6	7

Exemplo: número de inversões

Problema: Dada uma permutação $p[1\,..\,n]$, determinar o número de inversões em p.

Uma inversão é um par (i, j) de índices de p tal que $i < j$ e $p[i] > p[j]$.

Entrada:

	1	2	3	4	5	6	7	8	9
p	2	4	1	9	5	3	8	6	7

Saída: 11

Inversões: $(1, 3)$, $(2, 3)$, $(4, 5)$, $(2, 6)$, $(4, 6)$, $(5, 6)$, $(4, 7)$, $(4, 8)$, $(7, 8)$, $(4, 9)$ e $(7, 9)$.

Número de inversões

CONTA-INVERSÕES (p, n)

```
c ← 0
para i ← 1 até n − 1 faça
        para j ← i + 1 até n faça
                se p[i] > p[j]
                        então c ← c + 1
devolva c
```

Número de inversões

CONTA-INVERSÕES (p, n)

```
1   c ← 0
2   para i ← 1 até n − 1 faça
3           para j ← i + 1 até n faça
4                   se p[i] > p[j]
5                           então c ← c + 1
6   devolva c
```

linha	consumo de todas as execuções da linha
1	?
2	?
3	?
4	?
5	?
6	?
total	?

Número de inversões

CONTA-INVERSÕES (p, n)

```
1   c ← 0
2   para i ← 1 até n − 1 faça
3           para j ← i + 1 até n faça
4                   se p[i] > p[j]
5                           então c ← c + 1
6   devolva c
```

linha	consumo de todas as execuções da linha
1	$\mathrm{O}(1)$
2	$\mathrm{O}(n)$
3	$\mathrm{O}(n^2)$
4	$\mathrm{O}(n^2)$
5	$\mathrm{O}(n^2)$
6	$\mathrm{O}(1)$
total	$\mathrm{O}(3n^2 + n + 2) = \mathrm{O}(n^2)$

Conclusão

O algoritmo CONTA-INVERSÕES consome $\mathrm{O}(n^2)$ unidades de tempo.

Também escreve-se

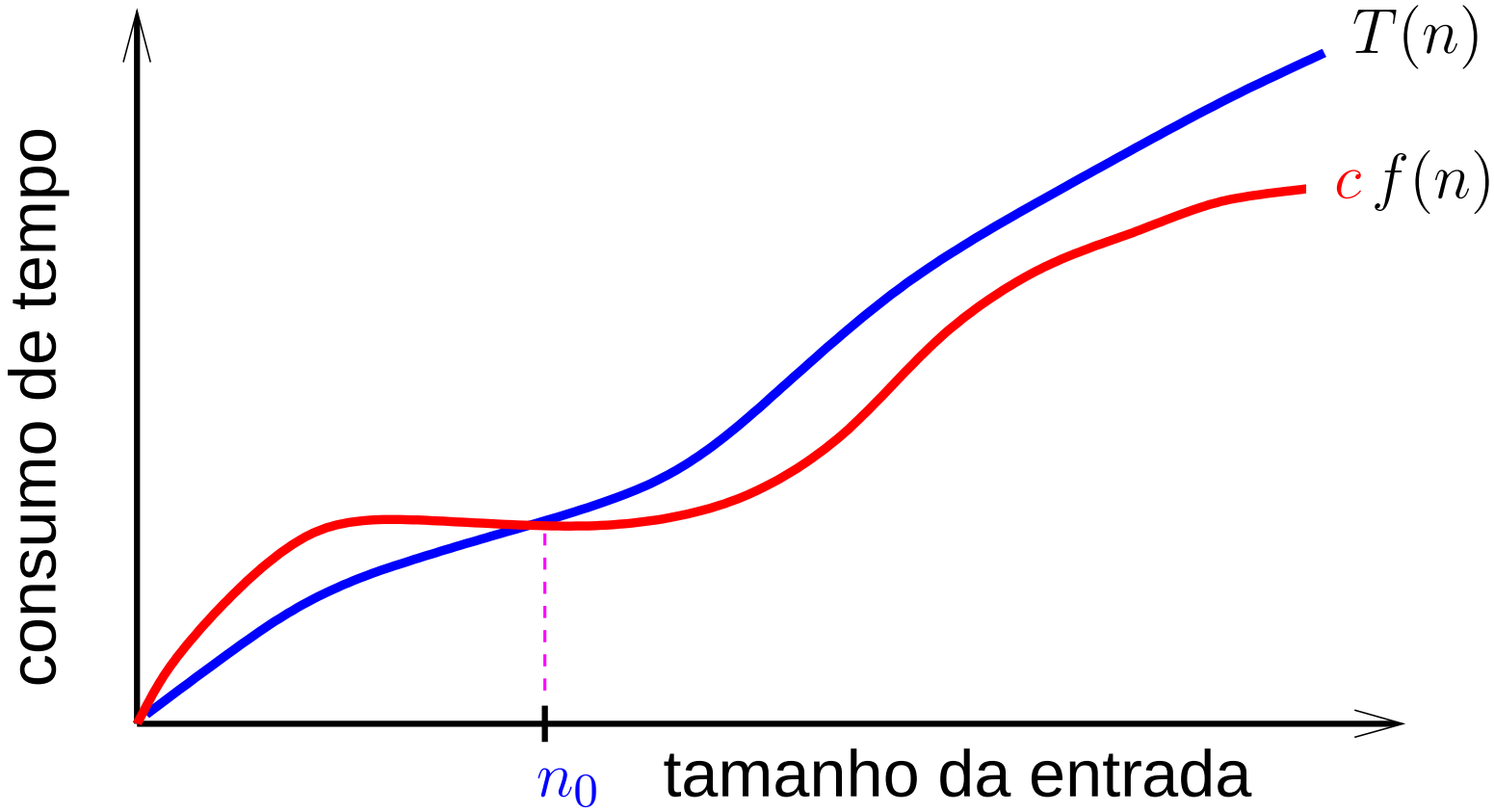
O algoritmo CONTA-INVERSÕES consome tempo $\mathrm{O}(n^2)$.

Notação Omega

Dizemos que $T(n)$ é $\Omega(f(n))$ se existem constantes positivas c e n_0 tais que

$$c\,f(n) \;\leq\; T(n)$$

para todo $n \geq n_0$.

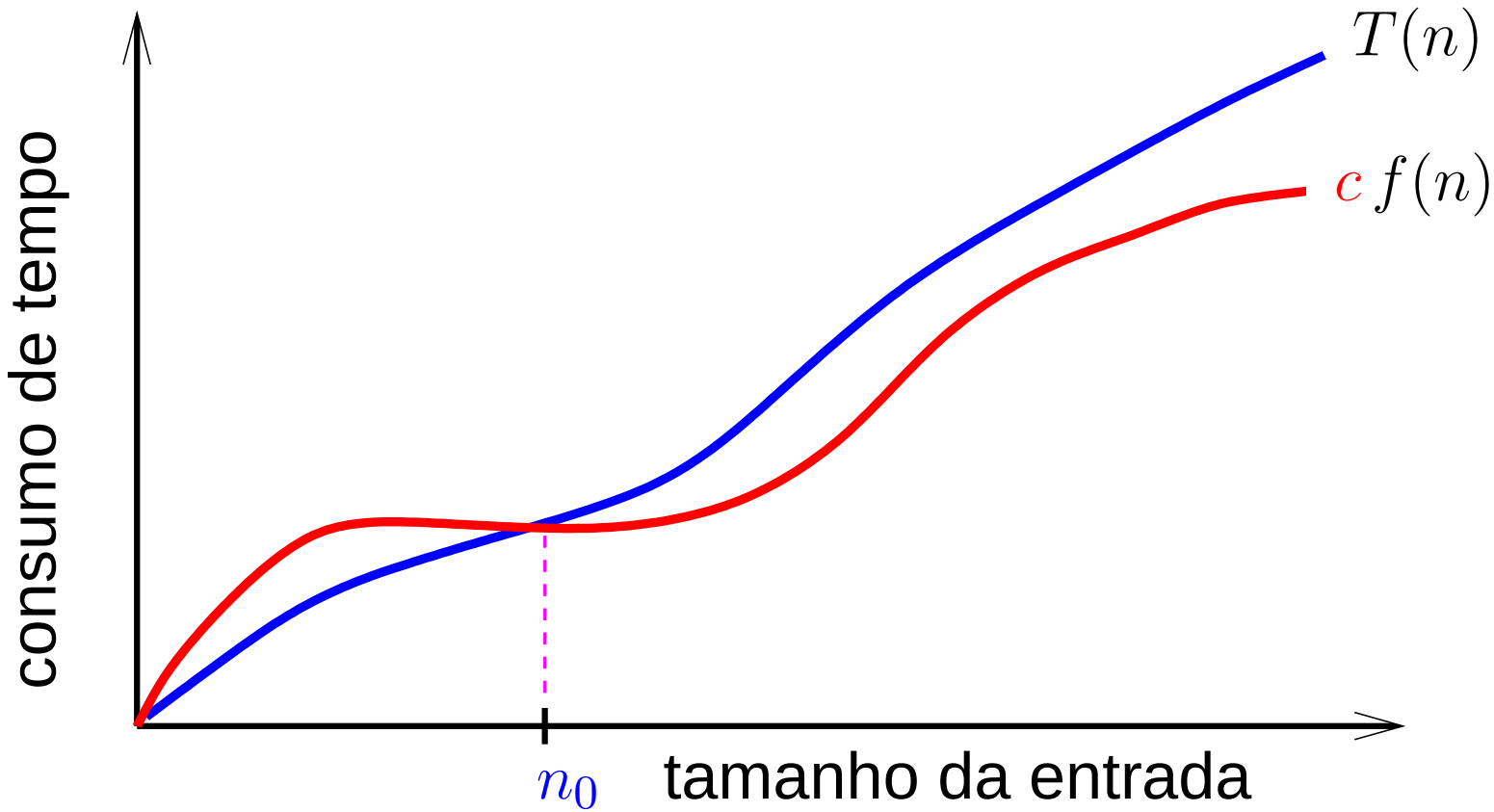


Mais informal

$T(n) = \Omega(f(n))$ se existe $c > 0$ tal que

$$c\, f(n) \leq T(n)$$

para todo n suficientemente GRANDE.



Exemplos

Exemplo 1

Se $T(n) \geq 0.001n^2$ para todo $n \geq 8$, então $T(n)$ é $\Omega(n^2)$.

Exemplos

Exemplo 1

Se $T(n) \geq 0.001n^2$ para todo $n \geq 8$, então $T(n)$ é $\Omega(n^2)$.

Prova: Aplique a definição com $c = 0.001$ e $n_0 = 8$.

Exemplo 2

O consumo de tempo do CONTA-INVERSÕES é $\mathrm{O}(n^2)$ e também $\Omega(n^2)$.

Exemplo 2

O consumo de tempo do CONTA-INVERSÕES é $\mathrm{O}(n^2)$ e também $\Omega(n^2)$.

```
CONTA-INVERSÕES (p, n)
    c ← 0
    para i ← 1 até n − 1 faça
        para j ← i + 1 até n faça
            se p[i] > p[j]
                então c ← c + 1
    devolva c
```

Exemplo 2

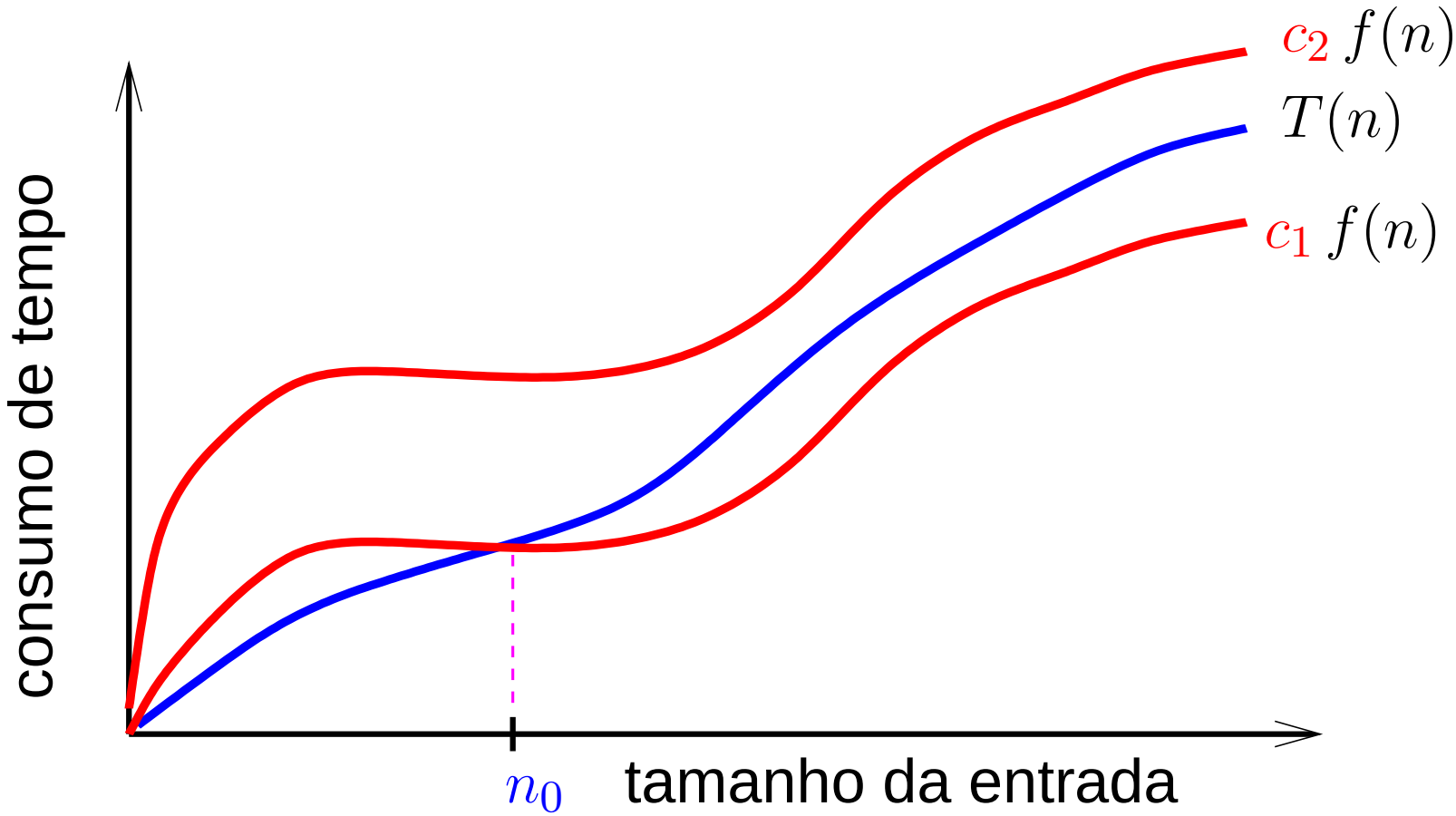
O consumo de tempo do CONTA-INVERSÕES é $\mathrm{O}(n^2)$ e também $\Omega(n^2)$.

linha	todas as execuções da linha	
1	$=$	1
2	$=$	n
3	$=$	$(n+2)(n-1)/2$
4	$=$	$n(n-1)/2$
5	$\geq$	0
6	$=$	1
total	$\geq$	$n^2 + n \;=\; \Omega(n^2)$

Notação Theta

Sejam $T(n)$ e $f(n)$ funções dos inteiros no reais.
Dizemos que $T(n)$ é $\Theta(f(n))$ se

$T(n)$ é $\mathrm{O}(f(n))$ e $T(n)$ é $\Omega(f(n))$.

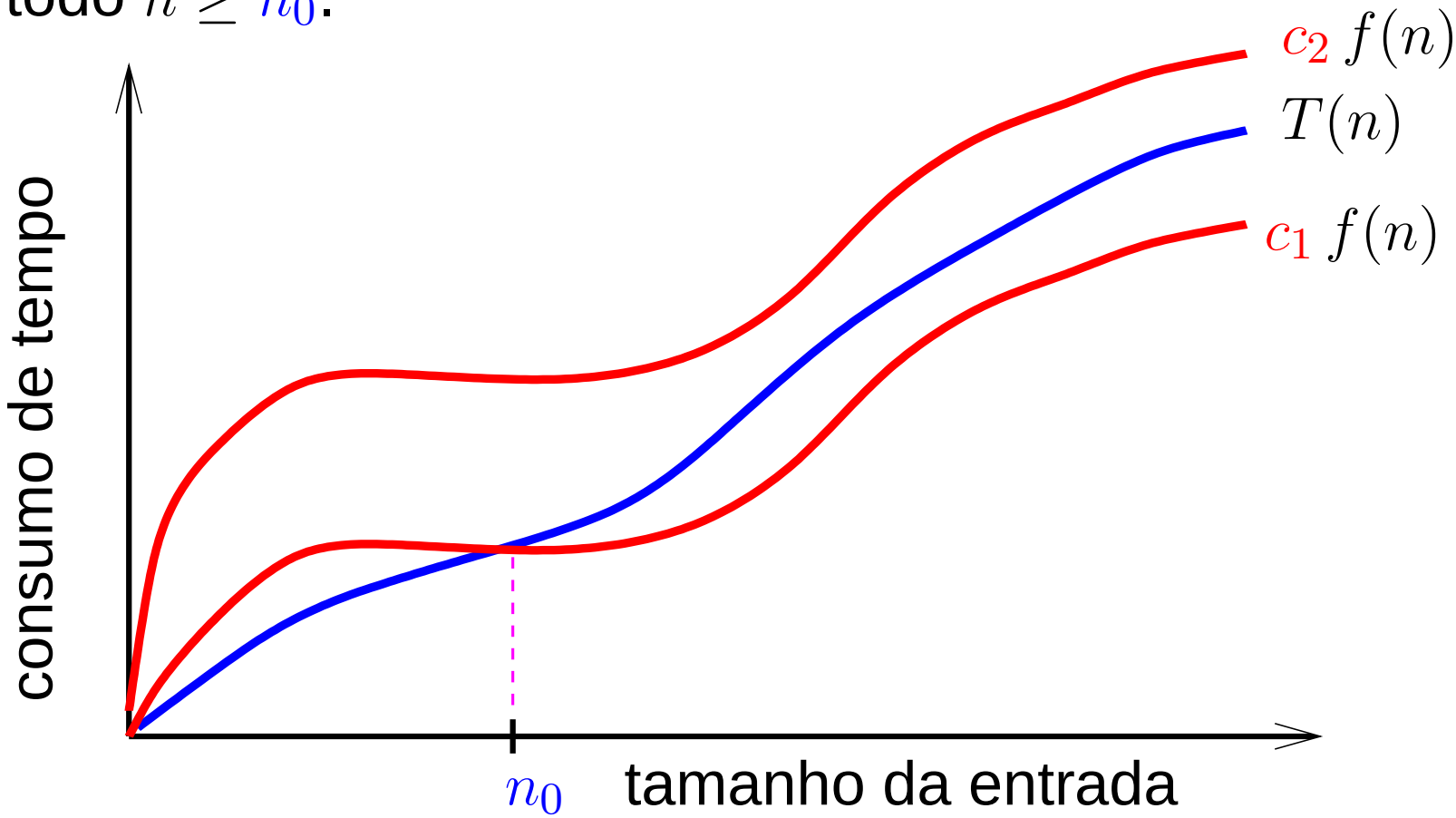


Notação Theta

Dizemos que $T(n)$ é $\Theta(f(n))$ se se existem constantes positivas c_1, c_2 e n_0 tais que

$$c_1\, f(n) \;\leq\; T(n) \;\leq\; c_2\, f(n)$$

para todo $n \geq n_0$.



Intuitivamente

Comparação assintótica, ou seja, para n ENORME.

comparação	comparação assintótica
$T(n) \leq f(n)$	$T(n)$ é $\mathrm{O}(f(n))$
$T(n) \geq f(n)$	$T(n)$ é $\Omega(f(n))$
$T(n) = f(n)$	$T(n)$ é $\Theta(f(n))$

Tamanho máximo de problemas

Suponha que cada operação consome 1 microsegundo ($1\mu s$).

consumo de tempo (μs)	Tamanho máximo de problemas (n)		
	1 segundo	1 minuto	1 hora
$400n$	2500	150000	9000000
$20\,n\,\lceil \lg n \rceil$	4096	166666	7826087
$2n^2$	707	5477	42426
n^4	31	88	244
2^n	19	25	31

Michael T. Goodrich e Roberto Tamassia, *Projeto de Algoritmos*, Bookman.

Crescimento de algumas funções

n	$\lg n$	$\sqrt{n}$	$n \lg n$	n^2	n^3	2^n
2	1	1,4	2	4	8	4
4	2	2	8	16	64	16
8	3	2,8	24	64	512	256
16	4	4	64	256	4096	65536
32	5	5,7	160	1024	32768	4294967296
64	6	8	384	4096	262144	1,8 10^{19}
128	7	11	896	16384	2097152	3,4 10^{38}
256	8	16	1048	65536	16777216	1,1 10^{77}
512	9	23	4608	262144	134217728	1,3 10^{154}
1024	10	32	10240	1048576	1,1 10^{9}	1,7 10^{308}

Nomes de classes Θ

classe	nome
$\Theta(1)$	constante
$\Theta(\log n)$	logarítmica
$\Theta(n)$	linear
$\Theta(n \log n)$	$n \log n$
$\Theta(n^2)$	quadrática
$\Theta(n^3)$	cúbica
$\Theta(n^k)$ com $k \geq 1$	polinomial
$\Theta(2^n)$	exponencial
$\Theta(a^n)$ com $a > 1$	exponencial

Palavras de Cautela

Suponha que $\mathcal{A}$ e $\mathcal{B}$ são algoritmos para um mesmo problema. Suponha que o consumo de tempo de $\mathcal{A}$ é "essencialmente" $100\,n$ e que o consumo de tempo de $\mathcal{B}$ é "essencialmente" $n \log_{10} n$.

Palavras de Cautela

Suponha que $\mathcal{A}$ e $\mathcal{B}$ são algoritmos para um mesmo problema. Suponha que o consumo de tempo de $\mathcal{A}$ é "essencialmente" $100\,n$ e que o consumo de tempo de $\mathcal{B}$ é "essencialmente" $n \log_{10} n$.

$100\,n$ é $\Theta(n)$ e $n \log_{10} n$ é $\Theta(n \lg n)$.
Logo, $\mathcal{A}$ é assintoticamente mais eficiente que $\mathcal{B}$.

Palavras de Cautela

Suponha que $\mathcal{A}$ e $\mathcal{B}$ são algoritmos para um mesmo problema. Suponha que o consumo de tempo de $\mathcal{A}$ é "essencialmente" $100\,n$ e que o consumo de tempo de $\mathcal{B}$ é "essencialmente" $n \log_{10} n$.

$100\,n$ é $\Theta(n)$ e $n \log_{10} n$ é $\Theta(n \lg n)$.
Logo, $\mathcal{A}$ é assintoticamente mais eficiente que $\mathcal{B}$.

$\mathcal{A}$ é mais eficiente que $\mathcal{B}$ para $n \geq 10^{100}$.

10^{100} = um *googol*
$\approx$ número de átomos no universo observável
= número ENORME

Palavras de Cautela

Conclusão:

Lembre das constantes e termos de baixa ordem que estão "escondidos" na notação assintótica.

Em geral um algoritmo que consome tempo $\Theta(n \lg n)$, e com fatores constantes razoáveis, é bem eficiente.

Um algoritmo que consome tempo $\Theta(n^2)$ pode, algumas vezes, ser satisfatório.

Um algoritmo que consome tempo $\Theta(2^n)$ é dificilmente aceitável.

Do ponto de vista de AA, eficiente $=$ polinomial.

Exercício da aula passada

```
f1 (x, y)
    se x = 1 ou y = 1
        devolva 0
    senão
        devolva f1(x − 1, y) + f1(x, y − 1) + xy
```

```
f2 (x, y)
    para i ← 1 até x faça
        t[i, 1] ← 0
    para j ← 2 até y faça
        t[1, j] ← 0
    para i ← 2 até x faça
        para j ← 2 até y faça
            t[i, j] ← t[i − 1, j] + t[i, j − 1] + ij
    devolva t[x, y]
```

Exercício da aula passada

Para que valores de x e y você acha que dá para começar a sentir diferença no tempo consumido por estas funções?

Exercício da aula passada

Para que valores de x e y você acha que dá para começar a sentir diferença no tempo consumido por estas funções?

- Valores menores que 30? 8 respostas.
- Entre 30 e 100? 12 respostas.
- Maiores que 100? 4 respostas.

Exercício da aula passada

Para que valores de x e y você acha que dá para começar a sentir diferença no tempo consumido por estas funções?

- Valores menores que 30? 8 respostas.
- Entre 30 e 100? 12 respostas.
- Maiores que 100? 4 respostas.

Uma frase a se pensar...

$f1$ funciona para x e y maiores que 100.

Exercício da aula passada

Matriz t inicializada com -1 em todas as posições.

```
f3 (x, y)
    se t[x, y] ≠ −1
        devolva t[x, y]
    se x = 1 ou y = 1
        r ← 0
    senão
        r ← f3(x − 1, y) + f3(x, y − 1) + xy
    t[x, y] ← r
    devolva r
```

MEMOIZAÇÃO

Exercício da aula passada

```
f4 (x, y)
    para j ← 1 até y faça
        t[j] ← 0
    para i ← 2 até x faça
        para j ← 2 até y faça
            t[j] ← t[j] + t[j − 1] + ij
    devolva t[y]
```

Mais econômica em relação à memória.

Número de inversões

Você sabe fazer um algoritmo mais rápido para o problema do número de inversões?

Número de inversões

Você sabe fazer um algoritmo mais rápido para o problema do número de inversões?

Note que o número de inversões pode ser $\Theta(n^2)$.
Portanto, para isso, não podemos contar de uma em uma as inversões, como faz o algoritmo que vimos hoje.
Temos que ser mais espertos...

Número de inversões

Você sabe fazer um algoritmo mais rápido para o problema do número de inversões?

Note que o número de inversões pode ser $\Theta(n^2)$.
Portanto, para isso, não podemos contar de uma em uma as inversões, como faz o algoritmo que vimos hoje.
Temos que ser mais espertos...

Idéia: vamos ordenar e contar ao mesmo tempo!

Número de inversões

Você sabe fazer um algoritmo mais rápido para o problema do número de inversões?

Note que o número de inversões pode ser $\Theta(n^2)$.
Portanto, para isso, não podemos contar de uma em uma as inversões, como faz o algoritmo que vimos hoje.
Temos que ser mais espertos...

Idéia: vamos ordenar e contar ao mesmo tempo!

Método: divisão e conquista.

Número de inversões

Você sabe fazer um algoritmo mais rápido para o problema do número de inversões?

Note que o número de inversões pode ser $\Theta(n^2)$.
Portanto, para isso, não podemos contar de uma em uma as inversões, como faz o algoritmo que vimos hoje.
Temos que ser mais espertos...

Idéia: vamos ordenar e contar ao mesmo tempo!

Método: divisão e conquista.

Resultado: um algoritmo $\mathrm{O}(n \lg n)$ para o problema do número de inversões de uma permutação!

Número de inversões

Problema: Dada uma permutação $p[1\,..\,n]$, determinar o número de inversões em p.

Queremos um algoritmo $\mathrm{O}(n \lg n)$ para o problema.

O número de inversões pode ser $\Theta(n^2)$.

Portanto, não podemos contar de uma em uma as inversões, como faz o algoritmo anterior.

Número de inversões

Problema: Dada uma permutação $p[1\,..\,n]$, determinar o número de inversões em p.

Queremos um algoritmo $\mathrm{O}(n \lg n)$ para o problema.

O número de inversões pode ser $\Theta(n^2)$.
Portanto, não podemos contar de uma em uma as inversões, como faz o algoritmo anterior.

Idéia: Vamos ordenar e contar ao mesmo tempo!
A ordenação ajuda a contar várias inversões de uma só vez.

Número de inversões

Problema: Dada uma permutação $p[1\,..\,n]$, determinar o número de inversões em p.

Queremos um algoritmo $\mathrm{O}(n \lg n)$ para o problema.

O número de inversões pode ser $\Theta(n^2)$.
Portanto, não podemos contar de uma em uma as inversões, como faz o algoritmo anterior.

Idéia: Vamos ordenar e contar ao mesmo tempo!
A ordenação ajuda a contar várias inversões de uma só vez.

Que algoritmo de ordenação usaremos?

Número de inversões

Problema: Dada uma permutação $p[1\,..\,n]$, determinar o número de inversões em p.

Queremos um algoritmo $\mathrm{O}(n \lg n)$ para o problema.

O número de inversões pode ser $\Theta(n^2)$.
Portanto, não podemos contar de uma em uma as inversões, como faz o algoritmo anterior.

Idéia: Vamos ordenar e contar ao mesmo tempo!
A ordenação ajuda a contar várias inversões de uma só vez.

Que algoritmo de ordenação usaremos?

Duas opções: o MERGESORT e o HEAPSORT.
Qual deles parece mais adequado?

Número de inversões

Problema: Dada uma permutação $p[1\,..\,n]$, determinar o número de inversões em p.

Queremos um algoritmo $\mathrm{O}(n \lg n)$ para o problema.

O número de inversões pode ser $\Theta(n^2)$.
Portanto, não podemos contar de uma em uma as inversões, como faz o algoritmo anterior.

Idéia: Vamos ordenar e contar ao mesmo tempo!
A ordenação ajuda a contar várias inversões de uma só vez.

Que algoritmo de ordenação usaremos?

Duas opções: o MERGESORT e o HEAPSORT.
Qual deles parece mais adequado?

Resposta: o MERGESORT.

Análise de Algoritmos

CLRS 2.3, 3.2, 4.1 e 4.2

Essas transparências foram adaptadas das transparências do Prof. Paulo Feofiloff e do Prof. José Coelho de Pina.

Número de inversões

Problema: Dada uma permutação $p[1\,..\,n]$, determinar o número de inversões em p.

Queremos um algoritmo $\mathrm{O}(n \lg n)$ para o problema.

O número de inversões pode ser $\Theta(n^2)$.

Portanto, não podemos contar de uma em uma as inversões, como faz o algoritmo anterior.

Número de inversões

Problema: Dada uma permutação $p[1\,..\,n]$, determinar o número de inversões em p.

Queremos um algoritmo $\mathrm{O}(n \lg n)$ para o problema.

O número de inversões pode ser $\Theta(n^2)$.
Portanto, não podemos contar de uma em uma as inversões, como faz o algoritmo anterior.

Idéia: Vamos ordenar e contar ao mesmo tempo!
A ordenação ajuda a contar várias inversões de uma só vez.

Número de inversões

Problema: Dada uma permutação $p[1\,..\,n]$, determinar o número de inversões em p.

Queremos um algoritmo $\mathrm{O}(n \lg n)$ para o problema.

O número de inversões pode ser $\Theta(n^2)$.
Portanto, não podemos contar de uma em uma as inversões, como faz o algoritmo anterior.

Idéia: Vamos ordenar e contar ao mesmo tempo!
A ordenação ajuda a contar várias inversões de uma só vez.

Que algoritmo de ordenação usaremos?

Número de inversões

Problema: Dada uma permutação $p[1\,..\,n]$, determinar o número de inversões em p.

Queremos um algoritmo $\mathrm{O}(n \lg n)$ para o problema.

O número de inversões pode ser $\Theta(n^2)$.
Portanto, não podemos contar de uma em uma as inversões, como faz o algoritmo anterior.

Idéia: Vamos ordenar e contar ao mesmo tempo!
A ordenação ajuda a contar várias inversões de uma só vez.

Que algoritmo de ordenação usaremos?

Duas opções: o MERGESORT e o HEAPSORT.
Qual deles parece mais adequado?

Número de inversões

Problema: Dada uma permutação $p[1\,..\,n]$, determinar o número de inversões em p.

Queremos um algoritmo $\mathrm{O}(n \lg n)$ para o problema.

O número de inversões pode ser $\Theta(n^2)$.
Portanto, não podemos contar de uma em uma as inversões, como faz o algoritmo anterior.

Idéia: Vamos ordenar e contar ao mesmo tempo!
A ordenação ajuda a contar várias inversões de uma só vez.

Que algoritmo de ordenação usaremos?

Duas opções: o MERGESORT e o HEAPSORT.
Qual deles parece mais adequado?

Resposta: o MERGESORT.

Merge-Sort

Rearranja $A[p\,..\,r]$, com $p \leq r$, em ordem crescente.

```
MERGESORT (A, p, r)
1    se p < r
2        então q ← ⌊(p + r)/2⌋
3              MERGESORT (A, p, q)
4              MERGESORT (A, q + 1, r)
5              INTERCALA (A, p, q, r)
```

Método: Divisão e conquista.

Intercalação

Problema: Dados $A[p\,..\,q]$ e $A[q{+}1\,..\,r]$ crescentes, rearranjar $A[p\,..\,r]$ de modo que ele fique em ordem crescente.

Para que valores de q o problema faz sentido?

Entra:

	p				q				r
A	22	33	55	77	99	11	44	66	88

Intercalação

Problema: Dados $A[p\,..\,q]$ e $A[q{+}1\,..\,r]$ crescentes, rearranjar $A[p\,..\,r]$ de modo que ele fique em ordem crescente.

Para que valores de q o problema faz sentido?

Entra:

	p				q				r
A	22	33	55	77	99	11	44	66	88

Sai:

	p				q				r
A	11	22	33	44	55	66	77	88	99

Intercalação

```
INTERCALA (A, p, q, r)
    ▷ B[p..r] é um vetor auxiliar
    para i ← p até q faça
        B[i] ← A[i]
    para j ← q + 1 até r faça
        B[r + q + 1 − j] ← A[j]
    i ← p
    j ← r
    para k ← p até r faça
        se B[i] ≤ B[j]
            então A[k] ← B[i]
                i ← i + 1
            senão A[k] ← B[j]
                j ← j − 1
```

Adaptação do Merge-Sort

Conta o número de inversões de $A[p\,..\,r]$, com $p \leq r$, e rearranja $A[p\,..\,r]$ em ordem crescente.

```
CONTA-E-ORDENA (A, p, r)
    se p ≥ r
        então devolva 0
        senão q ← ⌊(p + r)/2⌋
              c ← CONTA-E-ORDENA (A, p, q) +
                  CONTA-E-ORDENA (A, q + 1, r) +
                  CONTA-E-INTERCALA (A, p, q, r)
              devolva c
```

Método: Divisão e conquista.

Contagem na intercalação

```
CONTA-E-INTERCALA (A, p, q, r)
    para i ← p até q faça
        B[i] ← A[i]
    para j ← q + 1 até r faça
        B[r + q + 1 − j] ← A[j]
    i ← p
    j ← r
    c ← 0                                   ▷ inicializa o contador
    para k ← p até r faça
        se B[i] ≤ B[j]
            então A[k] ← B[i]
                  i ← i + 1
            senão A[k] ← B[j]
                  j ← j − 1
                  c ← c + (q − i + 1)       ▷ conta inversões
    devolva c
```

Simulação

	p				q				r
A	22	33	55	77	99	11	44	66	88

B									

$c = 0$

Simulação

	k								
A									

	i								j
B	22	33	55	77	99	88	66	44	11

$c = 0$

Simulação

		k							
A	11								

	i							j	
B	22	33	55	77	99	88	66	44	11

$$c = 0 + 5 = 5$$

Simulação

			k						
A	11	22							

		i						j	
B	22	33	55	77	99	88	66	44	11

$c = 5$

Simulação

				k					
A	11	22	33						

			i					j	
B	22	33	55	77	99	88	66	44	11

$c = 5$

Simulação

					k				
A	11	22	33	44					

			i				j		
B	22	33	55	77	99	88	66	44	11

$$c = 5 + 3 = 8$$

Simulação

						k			
A	11	22	33	44	55				

				i			j		
B	22	33	55	77	99	88	66	44	11

$c = 8$

Simulação

							k		
A	11	22	33	44	55	66			

				i		j			
B	22	33	55	77	99	88	66	44	11

$$c = 8 + 2 = 10$$

Simulação

								k	
A	11	22	33	44	55	66	77		

					i	j			
B	22	33	55	77	99	88	66	44	11

$c = 10$

Simulação

									k
A	11	22	33	44	55	66	77	88	

$i = j$

B	22	33	55	77	99	88	66	44	11

$c = 10 + 1 = 11$

Simulação

A	11	22	33	44	55	66	77	88	99

					j	i			
B	22	33	55	77	99	88	66	44	11

$c = 11$

Contagem na intercalação

```
CONTA-E-INTERCALA (A, p, q, r)
    para i ← p até q faça
        B[i] ← A[i]
    para j ← q + 1 até r faça
        B[r + q + 1 − j] ← A[j]
    i ← p
    j ← r
    c ← 0                                   ▷ inicializa o contador
    para k ← p até r faça
        se B[i] ≤ B[j]
            então A[k] ← B[i]
                  i ← i + 1
            senão A[k] ← B[j]
                  j ← j − 1
                  c ← c + (q − i + 1)       ▷ conta inversões
    devolva c
```

Consumo de tempo

Quanto tempo consome em função de $n := r - p + 1$?

linha	consumo de todas as execuções da linha
1	$\mathrm{O}(n)$
2	$\mathrm{O}(n)$
3	$\mathrm{O}(n)$
4	$\mathrm{O}(n)$
5–7	$\mathrm{O}(1)$
8	$\mathrm{O}(n)$
9	$\mathrm{O}(n)$
10–14	$\mathrm{O}(n)$
15	$\mathrm{O}(1)$
total	$\mathrm{O}(7n + 2) = \mathrm{O}(n)$

Conclusão

O algoritmo CONTA-E-INTERCALA consome $\mathrm{O}(n)$ unidades de tempo.

Também escreve-se

O algoritmo CONTA-E-INTERCALA consome tempo $\mathrm{O}(n)$.

Análise do Conta-E-Ordena

Seja $T(n)$ o tempo consumido pelo CONTA-E-ORDENA.

Análise do Conta-E-Ordena

Seja $T(n)$ o tempo consumido pelo CONTA-E-ORDENA.

Vale a seguinte recorrência para $T(n)$:

$$T(n) = T(\lceil n/2 \rceil) + T(\lfloor n/2 \rfloor) + \mathrm{O}(n)$$

Análise do Conta-E-Ordena

Seja $T(n)$ o tempo consumido pelo CONTA-E-ORDENA.

Vale a seguinte recorrência para $T(n)$:

$$T(n) = T(\lceil n/2 \rceil) + T(\lfloor n/2 \rfloor) + \mathrm{O}(n)$$

Solução: $T(n) = \mathrm{O}(n \lg n)$.

Prova?

Análise do Conta-E-Ordena

Seja $T(n)$ o tempo consumido pelo CONTA-E-ORDENA.

Vale a seguinte recorrência para $T(n)$:

$$T(n) = T(\lceil n/2 \rceil) + T(\lfloor n/2 \rfloor) + \mathrm{O}(n)$$

Solução: $T(n) = \mathrm{O}(n \lg n)$.

Prova?

Considera-se a recorrência simplificada

$$T(n) = 2\,T(n/2) + n$$

definida apenas para n potência de 2.

Análise do Conta-E-Ordena

Seja $T(n)$ o tempo consumido pelo CONTA-E-ORDENA.

Vale a seguinte recorrência para $T(n)$:

$$T(n) = T(\lceil n/2 \rceil) + T(\lfloor n/2 \rfloor) + \mathrm{O}(n)$$

Solução: $T(n) = \mathrm{O}(n \lg n)$.

Prova?

Considera-se a recorrência simplificada

$$T(n) = 2\,T(n/2) + n$$

definida apenas para n potência de 2.

Prova-se por indução em n que $T(n) = n + n \lg n = \mathrm{O}(n \lg n)$.

Prova

Afirmação: A recorrência

$$T(n) = \begin{cases} 1 & \text{se } n = 1 \\ 2\,T(n/2) + n & \text{se } n \geq 2,\ n \text{ potência de } 2 \end{cases}$$

tem como soluçao $T(n) = n + n \lg n$.

Prova

Afirmação: A recorrência

$$T(n) = \begin{cases} 1 & \text{se } n = 1 \\ 2\,T(n/2) + n & \text{se } n \geq 2,\ n \text{ potência de } 2 \end{cases}$$

tem como soluçao $T(n) = n + n \lg n$.

Prova: Por indução em n.

Base: $n = 1$

$$T(1) = 1 = 1 + 1 \cdot 0 = 1 + 1 \lg 1.$$

Prova

Afirmação: A recorrência

$$T(n) = \begin{cases} 1 & \text{se } n = 1 \\ 2\,T(n/2) + n & \text{se } n \geq 2,\ n \text{ potência de } 2 \end{cases}$$

tem como soluçao $T(n) = n + n \lg n$.

Prova: Por indução em n.

Base: $n = 1$

$$T(1) = 1 = 1 + 1 \cdot 0 = 1 + 1 \lg 1.$$

Passo: $n \geq 2$

$$\begin{aligned} T(n) &= 2\,T(n/2) + n \\ &= 2\,(n/2 + (n/2)\lg(n/2)) + n \qquad \text{por indução} \\ &= 2n + n\lg(n/2) \\ &= 2n + n(\lg n - 1) \\ &= n + n \lg n. \end{aligned}$$

Resolução de recorrências

Mas como descobrimos que $T(n) = n + n \lg n$?

Resolução de recorrências

Mas como descobrimos que $T(n) = n + n \lg n$? No chute!

Resolução de recorrências

Mas como descobrimos que $T(n) = n + n \lg n$? No chute!

Uma maneira de se obter um "chute" de solução de recorrência é desenrolando a recorrência.

Resolução de recorrências

Mas como descobrimos que $T(n) = n + n \lg n$? No chute!

Uma maneira de se obter um "chute" de solução de recorrência é desenrolando a recorrência.

$$
\begin{aligned}
T(n) &= 2\,T(n/2) + n \\
&= 2\,(2\,T(n/2^2) + n/2) + n \;=\; 2^2 T(n/2^2) + 2n \\
&= 2^2\,(2\,T(n/2^3) + n/2^2) + 2n \;=\; 2^3 T(n/2^3) + 3n \\
&= 2^3\,(2\,T(n/2^4) + n/2^3) + 3n \;=\; 2^4 T(n/2^4) + 4n \\
&= \cdots \\
&= 2^k\,T(n/2^k) + kn,
\end{aligned}
$$

onde $k = \lg n$. Disso concluímos que

$$T(n) = n + n \lg n.$$

Análise de Algoritmos

CLRS 4.1 e 4.2

Essas transparências foram adaptadas das transparências do Prof. Paulo Feofiloff e do Prof. José Coelho de Pina.

Mergesort

```
MERGESORT (A, p, d)
1   se p < d
2       então q ← ⌊(p + d)/2⌋
3             MERGESORT (A, p, q)
4             MERGESORT (A, q + 1, d)
5             INTERCALA (A, p, q, d)
```

linha	consumo máximo na linha
1	$\Theta(1)$
2	$\Theta(1)$
3	$T(\lceil n/2 \rceil)$
4	$T(\lfloor n/2 \rfloor)$
5	$\Theta(n)$

$$T(n) = T(\lceil n/2 \rceil) + T(\lfloor n/2 \rfloor) + \Theta(n)$$

Mergesort

$T(n) :=$ consumo de tempo máximo quando $n = d - p + 1$

$$T(n) = T(\lceil n/2 \rceil) + T(\lfloor n/2 \rfloor) + \Theta(n) \tag{1}$$

Solução: $T(n)$ é $\Theta(???)$.

Receita:

- Substitua a notação assintótica por função da classe.
- Restrinja-se a n potência de 2.
- Estipule que na base o valor de é 1.
- Use expansão ou árvore de recorrência para determinar um “chute” de solução.
- Confira se o chute está correto.

Expansão

n potência de 2 e $k = \lg n$

Expansão

n potência de 2 e $k = \lg n$

$$T(n) = 2T(n/2) + n$$

Expansão

n potência de 2 e $k = \lg n$

$$
\begin{aligned}
T(n) &= 2T(n/2) + n \\
&= 2\big(2T(n/2^2) + n/2\big) + n \;=\; 2^2T(n/2^2) + 2n
\end{aligned}
$$

Expansão

n potência de 2 e $k = \lg n$

$$
\begin{aligned}
T(n) &= 2T(n/2) + n \\
&= 2\big(2T(n/2^2) + n/2\big) + n \;=\; 2^2T(n/2^2) + 2n \\
&= 2^2\big(2T(n/2^3) + n/2^2\big) + 2n \;=\; 2^3T(n/2^3) + 3n
\end{aligned}
$$

Expansão

n potência de 2 e $k = \lg n$

$$
\begin{aligned}
T(n) &= 2T(n/2) + n \\
&= 2\big(2T(n/2^2) + n/2\big) + n \;=\; 2^2T(n/2^2) + 2n \\
&= 2^2\big(2T(n/2^3) + n/2^2\big) + 2n \;=\; 2^3T(n/2^3) + 3n \\
&= \ldots \;=\; 2^kT(n/2^k) + kn
\end{aligned}
$$

Expansão

n potência de 2 e $k = \lg n$

$$
\begin{aligned}
T(n) &= 2T(n/2) + n \\
&= 2\big(2T(n/2^2) + n/2\big) + n \;=\; 2^2T(n/2^2) + 2n \\
&= 2^2\big(2T(n/2^3) + n/2^2\big) + 2n \;=\; 2^3T(n/2^3) + 3n \\
&= \ldots \;=\; 2^kT(n/2^k) + kn \\
&= n + n\lg n \;=\; \Theta(n\lg n)
\end{aligned}
$$

Conclusão: O MERGESORT consume $\Theta(n \lg n)$ unidades de tempo.

Exemplos

- $T(n) = T(\lfloor n/2 \rfloor) + \Theta(1)$
- $T(n) = T(n-1) + \Theta(n)$
- $T(n) = 3T(\lfloor n/2 \rfloor) + \Theta(n)$
- $T(n) = 2T(\lfloor n/2 \rfloor) + \Theta(n^2)$

Exemplos

- $T(n) = T(\lfloor n/2 \rfloor) + \Theta(1)$ $\qquad \Theta(\lg n)$
- $T(n) = T(n-1) + \Theta(n)$ $\qquad \Theta(n^2)$
- $T(n) = 3T(\lfloor n/2 \rfloor) + \Theta(n)$ $\qquad \Theta(n^{\lg 3})$
- $T(n) = 2T(\lfloor n/2 \rfloor) + \Theta(n^2)$ $\qquad \Theta(n^2)$

Segmento de soma máxima

Um segmento de um vetor $A[1\,..\,n]$ é qualquer subvetor da forma $A[e\,..\,d]$.

Problema: Dado um vetor $A[1\,..\,n]$ de números inteiros, determinar um segmento $A[e\,..\,d]$ de soma máxima.

Entra:

	1									n
A	31	-41	59	26	-53	58	97	-93	-23	84

Segmento de soma máxima

Um segmento de um vetor $A[1\,..\,n]$ é qualquer subvetor da forma $A[e\,..\,d]$.

Problema: Dado um vetor $A[1\,..\,n]$ de números inteiros, determinar um segmento $A[e\,..\,d]$ de soma máxima.

Entra:

	1									n
A	31	-41	59	26	-53	58	97	-93	-23	84

Sai:

	1		3				7			n
A	31	-41	59	26	-53	58	97	-93	-23	84

$A[e\,..\,d] = A[3\,..\,7]$ é segmento de soma máxima.
$A[3\,..\,7]$ tem soma 187.

Algoritmo café-com-leite

Determina um segmento de soma máxima de $A[1\,..\,n]$.

```
SEG-MAX-3 (A, n)
    somamax ← 0
    e ← 0    d ← −1    ▷ A[e..d] é vazio
    para i ← 1 até n faça
        para f ← i até n faça
            soma ← 0
            para k ← i até f faça
                soma ← soma + A[k]
            se soma > somamax então
                somamax ← soma    e ← i    d ← f
    devolva e, d e somamax
```

Consumo de tempo

Se a execução de cada linha de código consome 1 unidade de tempo o consumo total é:

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

Algoritmo arroz-com-feijão

Determina um segmento de soma máxima de $A[1\,..\,n]$.

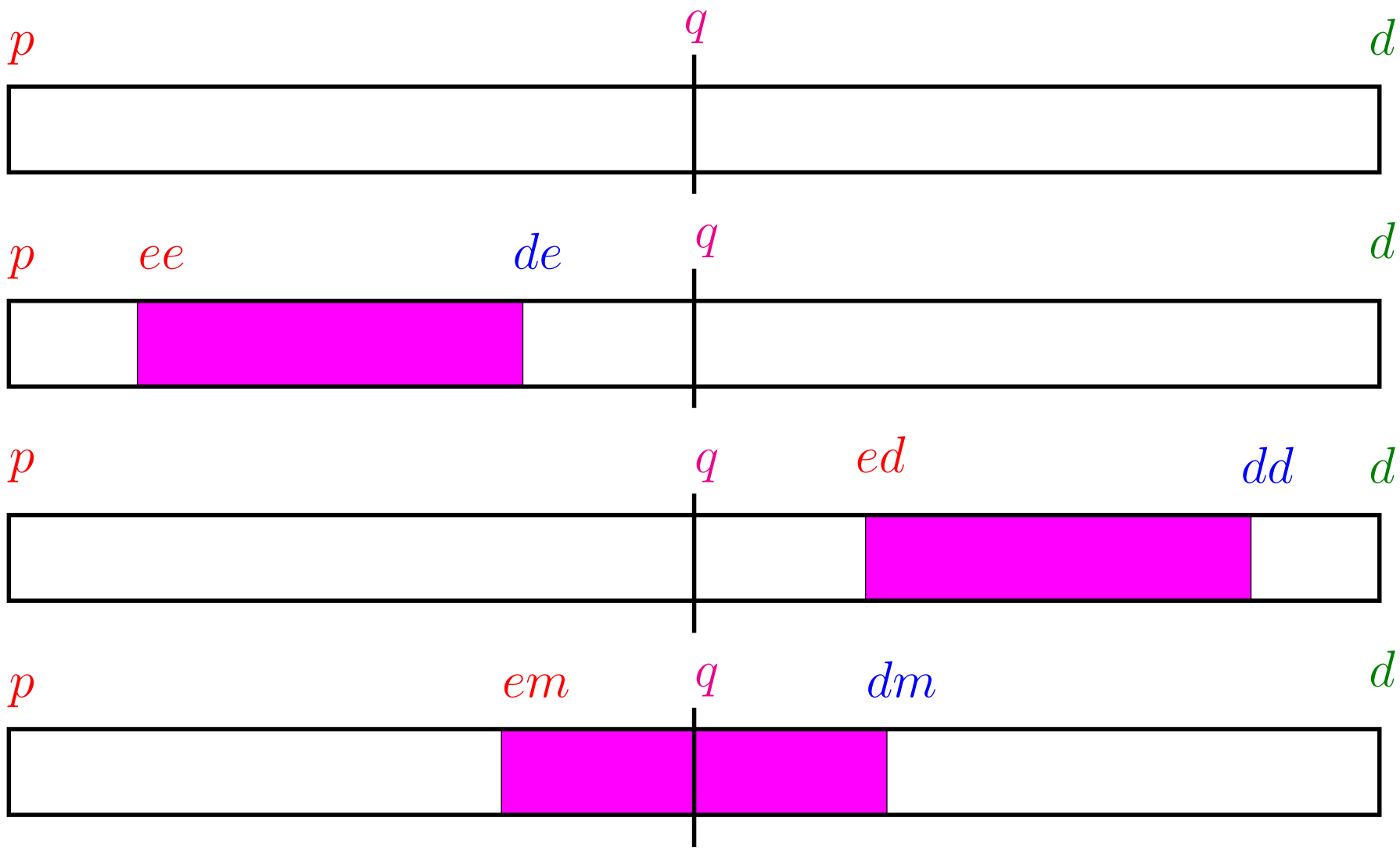
```
SEG-MAX-2 (A, n)
    somamax ← 0
    e ← 0    d ← −1    ▷ A[e..d] é vazio
    para i ← 1 até n faça
        soma ← 0
        para f ← i até n faça
            soma ← soma + A[f]
            se soma > somamax então
                somamax ← soma    e ← i    d ← f
    devolva e, d e somamax
```

Consumo de tempo

Se a execução de cada linha de código consome 1 unidade de tempo o consumo total é:

linha	todas as execuções da linha		
1-2	$=$	2	$= \Theta(1)$
3	$=$	$n+1$	$= \Theta(n)$
4	$=$	n	$= \Theta(n)$
5	$=$	$(n+1)+n+\cdots+2$	$= \Theta(n^2)$
6	$=$	$n+(n-1)+\cdots+1$	$= \Theta(n^2)$
7	$=$	$n+(n-1)+\cdots+1$	$= \Theta(n^2)$
8	$\leq$	$n+(n-1)+\cdots+1$	$= \mathrm{O}(n^2)$
9	$=$	1	$= \Theta(1)$
total	$=$	$\Theta(3n^2+2n+2)+\mathrm{O}(n^2)$	$= \Theta(n^2)$

Solução de divisão-e-conquista



Algoritmo de divisão-e-conquista

Setermina soma máxima de um seg. de $A[p\,..\,d]$.

```
SEG-MAX-DC (A, p, d)
    se p = d então devolva max(0, A[p])
    q ← ⌊(p + d)/2⌋
    maxesq ← SEG-MAX-DC(A, p, q)
    maxdir ← SEG-MAX-DC(A, q + 1, d)
    max2esq ← soma ← A[q]
    para i ← q − 1 decrescendo até p faça
        soma ← soma + A[i]
        max2esq ← max(max2esq, soma)
    max2dir ← soma ← A[q + 1]
    para f ← q + 2 até d faça
        soma ← soma + A[f]
        max2dir ← max(max2dir, soma)
    maxcruz ← max2esq + max2dir
    devolva max(maxesq, maxcruz, maxdir)
```

Algoritmo de divisão-e-conquista

Convença-se que o algoritmo anterior funciona. Ou seja, que ele de fato devolve a resposta correta.

Analise o consumo de tempo do algoritmo.

Análise de Algoritmos

KT 5.5 e CLRS 28.2

Essas transparências foram adaptadas das transparências do Prof. Paulo Feofiloff e do Prof. José Coelho de Pina.

Segmento de soma máxima

Um segmento de um vetor $A[1\,..\,n]$ é qualquer subvetor da forma $A[e\,..\,d]$.

Problema: Dado um vetor $A[1\,..\,n]$ de números inteiros, determinar um segmento $A[e\,..\,d]$ de soma máxima.

Entra:

	1									n
A	31	-41	59	26	-53	58	97	-93	-23	84

Segmento de soma máxima

Um segmento de um vetor $A[1\,..\,n]$ é qualquer subvetor da forma $A[e\,..\,d]$.

Problema: Dado um vetor $A[1\,..\,n]$ de números inteiros, determinar um segmento $A[e\,..\,d]$ de soma máxima.

Entra:

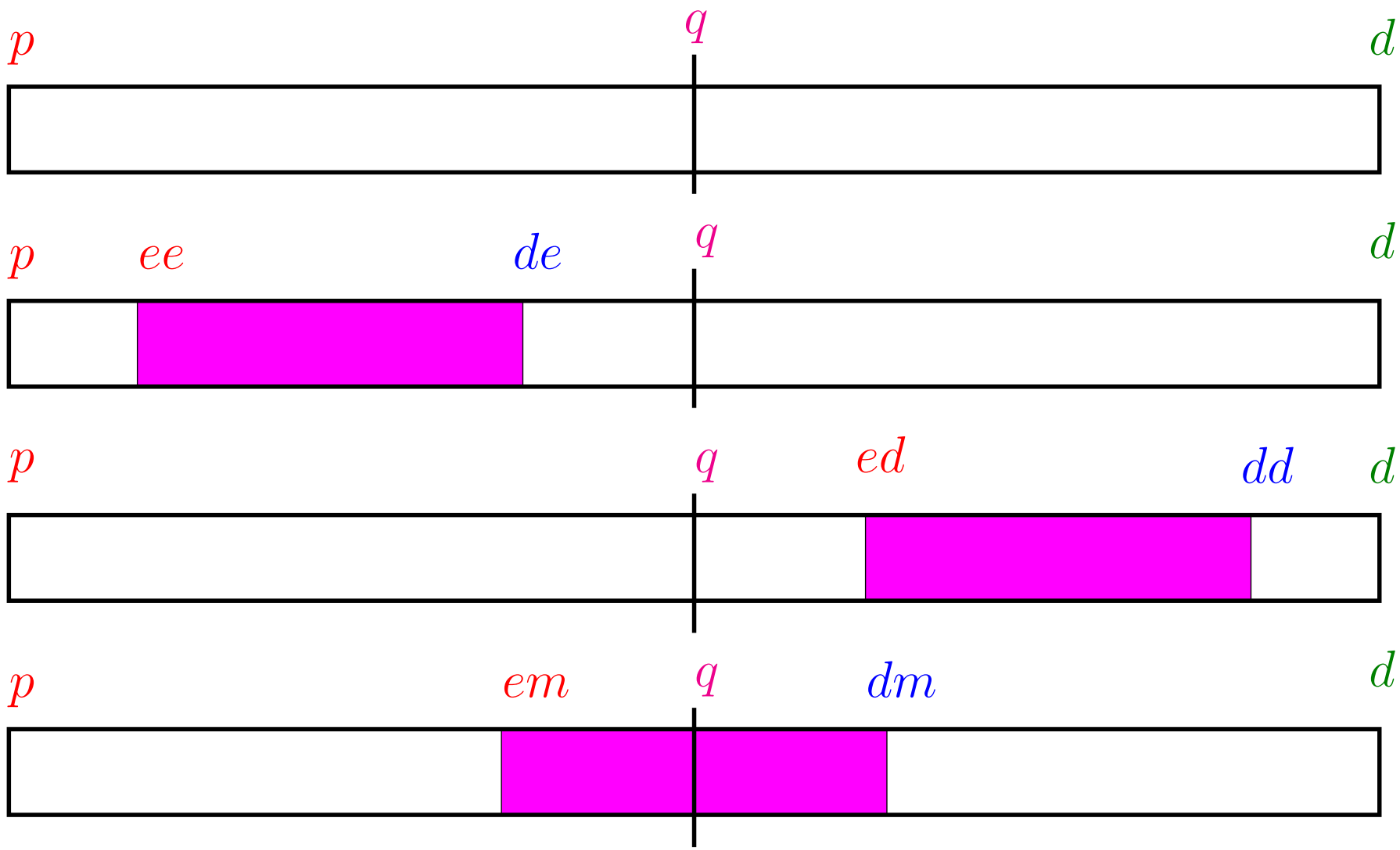
	1									n
A	31	-41	59	26	-53	58	97	-93	-23	84

Sai:

	1		3				7			n
A	31	-41	59	26	-53	58	97	-93	-23	84

$A[e\,..\,d] = A[3\,..\,7]$ é segmento de soma máxima.
$A[3\,..\,7]$ tem soma 187.

Solução de divisão-e-conquista



Algoritmo de divisão-e-conquista

Setermina soma máxima de um seg. de $A[p\,..\,d]$.

```
SEG-MAX-DC (A, p, d)
    se p = d então devolva max(0, A[p])
    q ← ⌊(p + d)/2⌋
    maxesq ← SEG-MAX-DC(A, p, q)
    maxdir ← SEG-MAX-DC(A, q + 1, d)
    max2esq ← soma ← A[q]
    para i ← q − 1 decrescendo até p faça
        soma ← soma + A[i]
        max2esq ← max(max2esq, soma)
    max2dir ← soma ← A[q + 1]
    para f ← q + 2 até d faça
        soma ← soma + A[f]
        max2dir ← max(max2dir, soma)
    maxcruz ← max2esq + max2dir
    devolva max(maxesq, maxcruz, maxdir)
```

Correção

Verifique que:

- $maxesq$ é a soma máxima de um segmento de $A[p\,..\,q]$;
- $maxdir$ é a soma máxima de um segmento de $A[q+1\,..\,d]$; e
- $maxcruz$ é a soma máxima de um segmento da forma $A[i\,..\,f]$ com $i \leq q$ e $q+1 \leq f$.

Conclua que o algoritmo devolve a soma máxima de um segmento de $A[p\,..\,d]$.

Consumo de tempo

Se a execução de cada linha de código consome 1 unidade de tempo o consumo total é:

linha	todas as execuções da linha		
1-2	$=$	2	$= \Theta(1)$
3	$=$	$T(\lceil \frac{n}{2} \rceil)$	$= T(\lceil \frac{n}{2} \rceil)$
4	$=$	$T(\lfloor \frac{n}{2} \rfloor)$	$= T(\lfloor \frac{n}{2} \rfloor)$
5	$=$	1	$= \Theta(1)$
6	$=$	$\lceil \frac{n}{2} \rceil + 1$	$= \Theta(n)$
7-8	$=$	$\lceil \frac{n}{2} \rceil$	$= \Theta(n)$
9	$=$	1	$= \Theta(1)$
10	$=$	$\lfloor \frac{n}{2} \rfloor + 1$	$= \Theta(n)$
11-12	$=$	$\lfloor \frac{n}{2} \rfloor$	$= \Theta(n)$
13-14	$=$	2	$= \Theta(1)$
total	$=$	$T(\lfloor \frac{n}{2} \rfloor) + T(\lfloor \frac{n}{2} \rfloor) + \Theta(n)$	

Consumo de tempo

$T(n) :=$ consumo de tempo quando $n = d - p + 1$

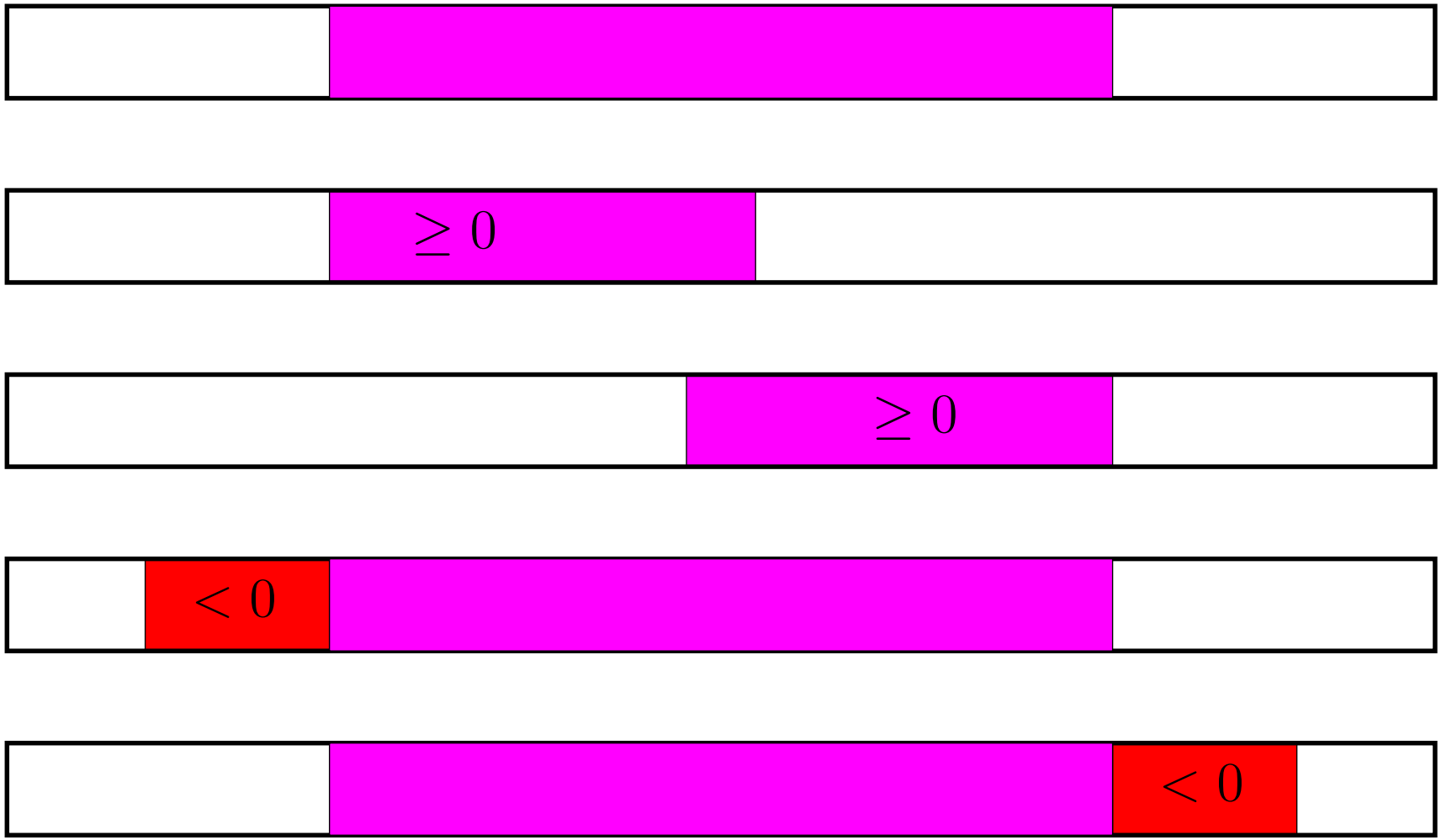
Na análise do consumo de tempo do SEG-MAX-DC chegamos a (já manjada) recorrência com Θ do lado direito:

$$T(n) = T(\lceil n/2 \rceil) + T(\lfloor n/2 \rfloor) + \Theta(n)$$

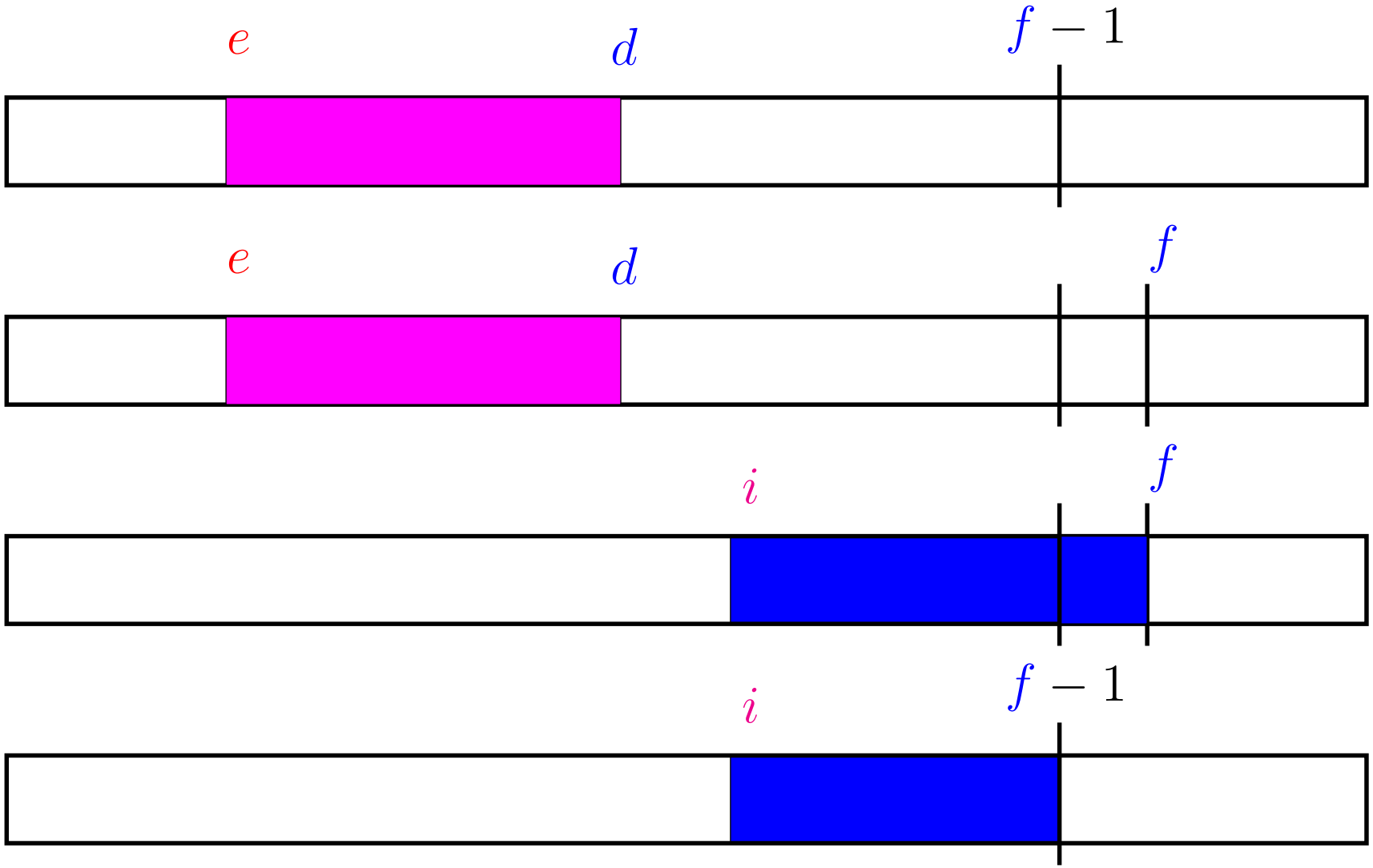
Solução assintótica: $T(n)$ é $\Theta(n \lg n)$.

Cara da solução

Solução



Solução indutiva



Algoritmo linear

Determina um segmento de soma máxima de $A[1\,..\,n]$ (por Jay Kadane).

```
SEG-MAX-1 (A, n)
    somamax ← 0
    e ← 0    d ← −1    ▷ A[e..d] é vazio
    i ← 1
    soma ← 0
    para f ← 1 até n faça
        se soma + A[f] < 0
            então i ← f + 1    soma ← 0
            senão soma ← soma + A[f]
        se soma > somamax então
            somamax ← soma    e ← i    d ← f
    devolva e, d e somamax
```

Correção

Verifique que:

- $A[e\,..\,d]$ é um segmento de soma máxima em $A[1\,..\,f-1]$.
- $somamax$ é a soma de $A[e\,..\,d]$.
- $A[i\,..\,f-1]$ é um segmento de soma máxima que termina em $f-1$.
- $soma$ é a soma de $A[i\,..\,f-1]$.

Conclua que o algoritmo devolve a soma máxima de um segmento de $A[1\,..\,n]$.

Consumo de tempo

Se a execução de cada linha de código consome 1 unidade de tempo o consumo total é:

linha	todas as execuções da linha		
1-2	$=$	2	$= \Theta(1)$
3-4	$=$	2	$= \Theta(1)$
5	$=$	$n+1$	$= \Theta(n)$
6	$=$	n	$= \Theta(n)$
7-8	$=$	n	$= \Theta(n)$
9	$=$	n	$= \Theta(n)$
10	$\leq$	n	$= \mathrm{O}(n)$
11	$=$	1	$= \Theta(1)$
total	$=$	$\Theta(4n+3) + \mathrm{O}(n)$	$= \Theta(n)$

Conclusões

O consumo de tempo do algoritmo SEG-MAX-3 é $\Theta(n^3)$.

O consumo de tempo do algoritmo SEG-MAX-2 é $\Theta(n^2)$.

O consumo de tempo do algoritmo SEG-MAX-DC é $\Theta(n \lg n)$.

O consumo de tempo do algoritmo SEG-MAX-1 é $\Theta(n)$.

Técnicas

- Evitar recálculos. Usar espaço para armazenar resultados a fim de evitar recalculá-los (SEG-MAX-2, SEG-MAX-1, programação dinâmica).
- Divisão-e-conquista. Os algoritmos Mergesort e SEG-MAX-2 utilizam uma forma conhecida dessa técnica.
- Algoritmos incrementais/varredura. Como estender a solução de um subproblema a uma solução do problema (SEG-MAX-1).
- Delimitação inferior. Bons projetistas de algoritmos só dormem em paz quando sabem que seus algoritmos são o melhor possível (SEG-MAX-1).

Multiplicação de inteiros gigantescos

$n :=$ número de algarismos.

Problema: Dados dois números inteiros $X[1\,..\,n]$ e $Y[1\,..\,n]$ calcular o produto $X \cdot Y$.

Entra: Exemplo com $n = 12$

	12											1
X	9	2	3	4	5	5	4	5	6	2	9	8

Y	0	6	3	2	8	4	9	9	3	8	4	4

Multiplicação de inteiros gigantescos

$n :=$ número de algarismos.

Problema: Dados dois números inteiros $X[1\,..\,n]$ e $Y[1\,..\,n]$ calcular o produto $X \cdot Y$.

Entra: Exemplo com $n = 12$

	12											1
X	9	2	3	4	5	5	4	5	6	2	9	8

Y	0	6	3	2	8	4	9	9	3	8	4	4

Sai:

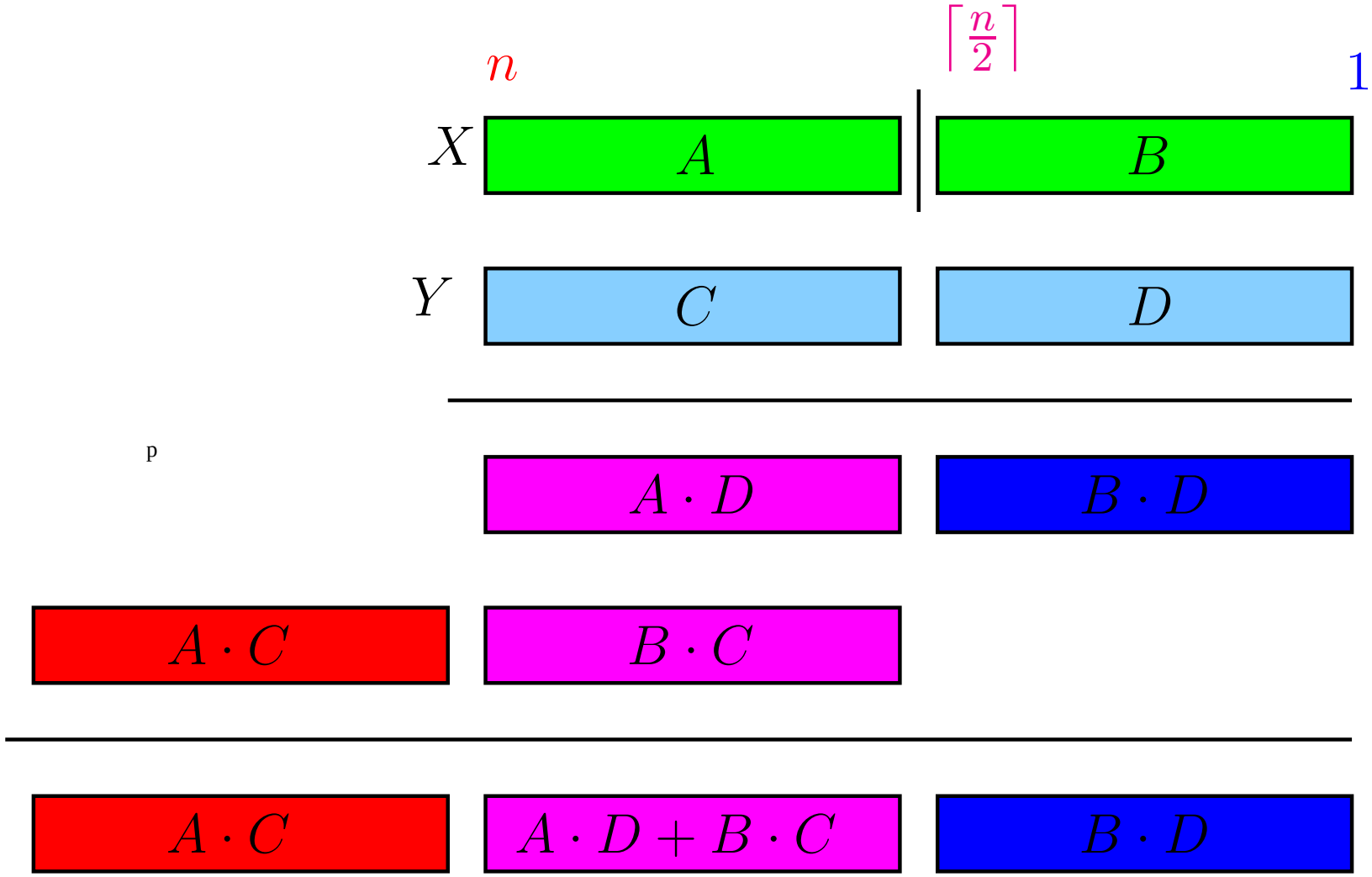
23										$X \cdot Y$												1
5	8	4	4	0	8	7	2	8	6	7	0	2	7	1	4	1	0	2	9	5	1	2

Algoritmo do ensino fundamental

```
                                    * * * * * * * *
                                  × * * * * * * * *
                                  ---------------
                                  * * * * * * * * *
                              * * * * * * * * *
                          * * * * * * * * *
                      * * * * * * * * *
                  * * * * * * * * *
              * * * * * * * * *
          * * * * * * * * *
      * * * * * * * * *
  * * * * * * * * *
  -----------------
* * * * * * * * * * * * * * * * *
```

O algoritmo do ensino fundamental é $\Theta(n^2)$.

Divisão e conquista



$$X \cdot Y = A \cdot C \times 10^{n} + (A \cdot D + B \cdot C) \times 10^{\lceil n/2 \rceil} + B \cdot D$$

Exemplo

			4		1		4		1
X	3	1	4	1	Y	5	9	3	6

Exemplo

		4		1				4		1	
X		3	1	4	1		Y	5	9	3	6

A	3	1		B	4	1		C	5	9		D	3	6

Exemplo

	4			1
X	3	1	4	1

	4			1
Y	5	9	3	6

A	3	1

B	4	1

C	5	9

D	3	6

$$X \cdot Y = A \cdot C \times 10^4 + (A \cdot D + B \cdot C) \times 10^2 + B \cdot D$$

$A \cdot C = 1829$ $\qquad$ $(A \cdot D + B \cdot C) = 1116 + 2419 = 3535$

$B \cdot D = 1476$

$A \cdot C$	1	8	2	9	0	0	0	0
$(A \cdot D + B \cdot C)$			3	5	3	5	0	0
$B \cdot D$					1	4	7	6
$X \cdot Y =$	1	8	6	4	4	9	7	6

Algoritmo de Multi-DC

Algoritmo recebe inteiros $X[1\,..\,n]$ e $Y[1\,..\,n]$ e devolve $X \cdot Y$.

MULT (X, Y, n)

1 **se** $n = 1$ **devolva** $X \cdot Y$
2 $q \leftarrow \lceil n/2 \rceil$
3 $A \leftarrow X[q+1\,..\,n] \qquad B \leftarrow X[1\,..\,q]$
4 $C \leftarrow Y[q+1\,..\,n] \qquad D \leftarrow Y[1\,..\,q]$
5 $E \leftarrow$ MULT$(A, C, \lfloor n/2 \rfloor)$
6 $F \leftarrow$ MULT$(B, D, \lceil n/2 \rceil)$
7 $G \leftarrow$ MULT$(A, D, \lceil n/2 \rceil)$
8 $H \leftarrow$ MULT$(B, C, \lceil n/2 \rceil)$
9 $R \leftarrow E \times 10^n + (G + H) \times 10^{\lceil n/2 \rceil} + F$
10 **devolva** R

$T(n) =$ consumo de tempo do algoritmo para multiplicar dois inteiros com n algarismos.

Consumo de tempo

linha		todas as execuções da linha
1	$=$	$\Theta(1)$
2	$=$	$\Theta(1)$
3	$=$	$\Theta(n)$
4	$=$	$\Theta(n)$
5	$=$	$T(\lfloor n/2 \rfloor)$
6	$=$	$T(\lceil n/2 \rceil)$
7	$=$	$T(\lceil n/2 \rceil)$
8	$=$	$T(\lceil n/2 \rceil)$
9	$=$	$\Theta(n)$
10	$=$	$\Theta(n)$
total	$=$	$T(\lfloor n/2 \rfloor) + 3\,T(\lceil n/2 \rceil) + \Theta(n)$

Consumo de tempo

Sabemos que

$$T(n) \;=\; T(\lfloor n/2 \rfloor) + 3\,T(\lceil n/2 \rceil) + \Theta(n)$$

está na mesma classe Θ que a solução de

$$\begin{aligned} T'(1) &= 1 \\ T'(n) &= 4T'(n/2) + n \quad \text{para } n = 2, 2^2, 2^3, \ldots \end{aligned}$$

n	1	2	4	8	16	32	64	128	256	512
$T'(n)$	1	6	28	120	496	2016	8128	32640	130816	523776

Conclusões

$T'(n)$ é $\Theta(n^2)$.

$T(n)$ é $\Theta(n^2)$.

O consumo de tempo do algoritmo MULT é $\Theta(n^2)$.

Tanto trabalho por nada . . .
Será?!?

Pensar pequeno

Olhar para números com 2 algarismos (n=2).

Suponha $X = a\,b$ e $Y = c\,d$.
Se cada multiplicação custa R$ 1,00 e
cada soma custa R$ 0,01, quanto custa $X \cdot Y$?

Pensar pequeno

Olhar para números com 2 algarismos (n=2).

Suponha $X = a\,b$ e $Y = c\,d$.
Se cada multiplicação custa R$ 1,00 e
cada soma custa R$ 0,01, quanto custa $X \cdot Y$?
Eis $X \cdot Y$ por R$ 4,03:

$$
\begin{array}{lccc}
X & & a & b \\
Y & & c & d \\
\hline
 & & ad & bd \\
 & ac & bc & \\
\hline
X \cdot Y & ac & ad + bc & bd
\end{array}
$$

$$X \cdot Y = ac \times 10^2 + (ad + bc) \times 10^1 + bd$$

Pensar pequeno

Olhar para números com 2 algarismos (n=2).

Suponha $X = a\,b$ e $Y = c\,d$.
Se cada multiplicação custa R$ 1,00 e
cada soma custa R$ 0,01, quanto custa $X \cdot Y$?
Eis $X \cdot Y$ por R$ 4,03:

$$
\begin{array}{lccc}
X & & a & b \\
Y & & c & d \\
\hline
 & & ad & bd \\
 & ac & bc & \\
\hline
X \cdot Y & ac & ad + bc & bd
\end{array}
$$

$$X \cdot Y = ac \times 10^2 + (ad + bc) \times 10^1 + bd$$

Solução mais barata?

Pensar pequeno

Olhar para números com 2 algarismos (n=2).

Suponha $X = a\,b$ e $Y = c\,d$.
Se cada multiplicação custa R$ 1,00 e
cada soma custa R$ 0,01, quanto custa $X \cdot Y$?
Eis $X \cdot Y$ por R$ 4,03:

$$
\begin{array}{lccc}
X & & a & b \\
Y & & c & d \\
\hline
 & & ad & bd \\
 & ac & bc & \\
\hline
X \cdot Y & ac & ad + bc & bd
\end{array}
$$

$$X \cdot Y = ac \times 10^2 + (ad + bc) \times 10^1 + bd$$

Solução mais barata?
Gauss faz por R$ 3,06!

$X \cdot Y$ por apenas R$ 3,06

X		a	b
Y		c	d
		ad	bd
	ac	bc	
$X \cdot Y$	ac	$ad + bc$	bd

$X \cdot Y$ por apenas R$ 3,06

X		a	b
Y		c	d
		ad	bd
	ac	bc	
$X \cdot Y$	ac	$ad + bc$	bd

$$(a+b)(c+d) = ac + ad + bc + bd \Rightarrow$$
$$ad + bc = (a+b)(c+d) - ac - bd$$

$$g = (a+b)(c+d) \qquad e = ac \qquad f = bd \qquad h = g - e - f$$

$$X \cdot Y \text{ (por R\$ 3,06)} = e \times 10^2 + h \times 10^1 + f$$

Exemplo

$$X = 2133 \quad Y = 2312 \qquad X \cdot Y = ?$$

$$ac = ? \qquad bd = ? \qquad (a+b)(c+d) = ?$$

Exemplo

$$X = 2133 \quad Y = 2312 \qquad X \cdot Y = ?$$
$$ac = ? \quad bd = ? \quad (a+b)(c+d) = ?$$

$$X = 21 \quad Y = 23 \qquad X \cdot Y = ?$$
$$ac = ? \quad bd = ? \quad (a+b)(c+d) = ?$$

Exemplo

$$X = 2133 \quad Y = 2312 \qquad X \cdot Y = ?$$
$$ac = ? \qquad bd = ? \qquad (a+b)(c+d) = ?$$

$$X = 21 \quad Y = 23 \qquad X \cdot Y = ?$$
$$ac = ? \qquad bd = ? \qquad (a+b)(c+d) = ?$$

$$X = 2 \quad Y = 2 \quad X \cdot Y = 4$$

Exemplo

$$X = 2133 \quad Y = 2312 \qquad X \cdot Y = ?$$
$$ac = ? \quad bd = ? \quad (a+b)(c+d) = ?$$

$$X = 21 \quad Y = 23 \qquad X \cdot Y = ?$$
$$ac = 4 \quad bd = ? \quad (a+b)(c+d) = ?$$

Exemplo

$$X = 2133 \quad Y = 2312 \qquad X \cdot Y = ?$$
$$ac = ? \qquad bd = ? \qquad (a+b)(c+d) = ?$$

$$X = 21 \quad Y = 23 \qquad X \cdot Y = ?$$
$$ac = 4 \qquad bd = ? \qquad (a+b)(c+d) = ?$$

$$X = 1 \quad Y = 3 \quad X \cdot Y = 3$$

Exemplo

$$X = 2133 \quad Y = 2312 \qquad X \cdot Y = ?$$
$$ac = ? \quad bd = ? \quad (a+b)(c+d) = ?$$

$$X = 21 \quad Y = 23 \qquad X \cdot Y = ?$$
$$ac = 4 \quad bd = 3 \quad (a+b)(c+d) = ?$$

Exemplo

$$X = 2133 \quad Y = 2312 \qquad X \cdot Y = ?$$
$$ac = ? \qquad bd = ? \qquad (a+b)(c+d) = ?$$

$$X = 21 \quad Y = 23 \qquad X \cdot Y = ?$$
$$ac = 4 \qquad bd = 3 \qquad (a+b)(c+d) = ?$$

$$X = 3 \quad Y = 5 \quad X \cdot Y = 15$$

Exemplo

$$X = 2133 \quad Y = 2312 \qquad X \cdot Y = ?$$
$$ac = ? \quad bd = ? \quad (a+b)(c+d) = ?$$

$$X = 21 \quad Y = 23 \qquad X \cdot Y = 483$$
$$ac = 4 \quad bd = 3 \quad (a+b)(c+d) = 15$$

Exemplo

$$X = 2133 \quad Y = 2312 \qquad X \cdot Y = ?$$

$$ac = 483 \quad bd = ? \qquad (a+b)(c+d) = ?$$

Exemplo

$$X = 2133 \quad Y = 2312 \qquad X \cdot Y = ?$$
$$ac = 483 \quad bd = ? \qquad (a+b)(c+d) = ?$$

$$X = 33 \quad Y = 12 \qquad X \cdot Y = ?$$
$$ac = ? \quad bd = ? \qquad (a+b)(c+d) = ?$$

Exemplo

$$X = 2133 \quad Y = 2312 \qquad X \cdot Y = ?$$
$$ac = 483 \quad bd = ? \qquad (a+b)(c+d) = ?$$

$$X = 33 \quad Y = 12 \qquad X \cdot Y = 396$$
$$ac = 3 \quad bd = 6 \qquad (a+b)(c+d) = 18$$

Exemplo

$$X = 2133 \quad Y = 2312 \qquad X \cdot Y = ?$$

$$ac = 483 \quad bd = 396 \quad (a+b)(c+d) = ?$$

Exemplo

$$X = 2133 \quad Y = 2312 \qquad X \cdot Y = ?$$
$$ac = 483 \quad bd = 396 \quad (a+b)(c+d) = ?$$

$$X = 54 \quad Y = 35 \qquad X \cdot Y = ?$$
$$ac = ? \quad bd = ? \quad (a+b)(c+d) = ?$$

Exemplo

$$X = 2133 \quad Y = 2312 \qquad X \cdot Y = ?$$
$$ac = 483 \quad bd = 396 \quad (a+b)(c+d) = ?$$

$$X = 54 \quad Y = 35 \qquad X \cdot Y = 1890$$
$$ac = 15 \quad bd = 20 \quad (a+b)(c+d) = 72$$

Exemplo

$$X = 2133 \quad Y = 2312 \qquad X \cdot Y = ?$$

$$ac = 483 \quad bd = 396 \quad (a+b)(c+d) = 1890$$

Exemplo

$$X = 2133 \quad Y = 2312 \qquad X \cdot Y = 4931496$$

$$ac = 483 \quad bd = 396 \quad (a+b)(c+d) = 1890$$

Algoritmo Multi

Algoritmo recebe inteiros $X[1\,..\,n]$ e $Y[1\,..\,n]$ e devolve $X \cdot Y$ (Karatsuba e Ofman).

KARATSUBA (X, Y, n)

se $n \leq 3$ **devolva** $X \cdot Y$
$q \leftarrow \lceil n/2 \rceil$
$A \leftarrow X[q+1\,..\,n] \qquad B \leftarrow X[1\,..\,q]$
$C \leftarrow Y[q+1\,..\,n] \qquad D \leftarrow Y[1\,..\,q]$
$E \leftarrow$ KARATSUBA$(A, C, \lfloor n/2 \rfloor)$
$F \leftarrow$ KARATSUBA$(B, D, \lceil n/2 \rceil)$
$G \leftarrow$ KARATSUBA$(A + B, C + D, \lceil n/2 \rceil + 1)$
$H \leftarrow G - F - E$
$R \leftarrow E \times 10^n + H \times 10^{\lceil n/2 \rceil} + F$
devolva R

$T(n) =$ consumo de tempo do algoritmo para multiplicar dois inteiros com n algarismos.

Consumo de tempo

linha		todas as execuções da linha
1	$=$	$\Theta(1)$
2	$=$	$\Theta(1)$
3	$=$	$\Theta(n)$
4	$=$	$\Theta(n)$
5	$=$	$T(\lfloor n/2 \rfloor)$
6	$=$	$T(\lceil n/2 \rceil)$
7	$=$	$T(\lceil n/2 \rceil + 1)$
8	$=$	$\Theta(n)$
9	$=$	$\Theta(n)$
10	$=$	$\Theta(n)$
total	$=$	$T(\lfloor n/2 \rfloor) + T(\lceil n/2 \rceil) + T(\lceil n/2 \rceil + 1) + \Theta(n)$

Consumo de tempo

Sabemos que

$$T(n) = T(\lfloor n/2 \rfloor) + T(\lceil n/2 \rceil) + T(\lceil n/2 \rceil + 1) + \Theta(n)$$

está na mesma classe Θ que a solução de

$$\begin{aligned} T'(1) &= 1 \\ T'(n) &= 3T'(n/2) + n \quad \text{para } n = 2, 2^2, 2^3, \ldots \end{aligned}$$

n	1	2	4	8	16	32	64	128	256	512
$T'(n)$	1	5	19	65	211	665	2059	6305	19171	58025

Recorrência

Considere a recorrência

$$
\begin{aligned}
R(1) &= 1 \\
R(2) &= 1 \\
R(3) &= 1 \\
R(n) &= 3R(\lceil \tfrac{n}{2} \rceil + 1) + n \quad \text{para } n = 4, 5, 6 \ldots
\end{aligned}
$$

n	1	2	3	4	5	6	7	8	9	10
$T(n)$	1	1	1	7	14	15	29	36	45	53
$R(n)$	1	1	1	7	26	27	85	86	90	91

Recorrência

Considere a recorrência

$$
\begin{aligned}
R(1) &= 1 \\
R(2) &= 1 \\
R(3) &= 1 \\
R(n) &= 3R(\lceil \tfrac{n}{2} \rceil + 1) + n \quad \text{para } n = 4, 5, 6 \ldots
\end{aligned}
$$

n	1	2	3	4	5	6	7	8	9	10
$T(n)$	1	1	1	7	14	15	29	36	45	53
$R(n)$	1	1	1	7	26	27	85	86	90	91

Vamos mostra que $R(n)$ é $\mathrm{O}(n^{\lg 3})$.
Isto implica que $T(n)$ é $\mathrm{O}(n^{\lg 3})$.

Solução assintótica da recorrência

Vou mostrar que $R(n) \leq 31\,(n-3)^{\lg 3} - 6n$ para $n = 4, 5, 6, \ldots$

n	1	2	3	4	5	6	7	8	9	10
$R(n)$	1	1	1	7	26	27	85	86	90	91
$31(n-3)^{\lg 3} - 6n$	$*$	$*$	$*$	7	63	119	237	324	473	5910

Solução assintótica da recorrência

Vou mostrar que $R(n) \leq 31\,(n-3)^{\lg 3} - 6n$ para $n = 4, 5, 6, \ldots$

n	1	2	3	4	5	6	7	8	9	10
$R(n)$	1	1	1	7	26	27	85	86	90	91
$31(n-3)^{\lg 3} - 6n$	$*$	$*$	$*$	7	63	119	237	324	473	5910

Prova:

Se $n = 4$, então $R(n) = 7 = 31(n-3)^{\lg 3} - 6n$.

Solução assintótica da recorrência

Prova: (continuação) Se $n \geq 5$ vale que

$$
\begin{aligned}
R(n) &= 3R(\lceil n/2 \rceil + 1) + n \\
&\overset{\text{hi}}{\leq} 3(31(\lceil n/2 \rceil + 1 - 3)^{\lg 3} - 6(\lceil n/2 \rceil + 1)) + n \\
&\leq 3(31(\frac{(n+1)}{2} - 2)^{\lg 3} - 6(\frac{n}{2} + 1)) + n \\
&= 3(31(\frac{(n-3)}{2})^{\lg 3} - 3n - 6) + n \\
&= 3(31\frac{(n-3)^{\lg 3}}{2^{\lg 3}} - 3n - 6) + n \\
&= 3 \cdot 31\frac{(n-3)^{\lg 3}}{3} - 9n - 18 + n \\
&= 31(n-3)^{\lg 3} - 6n - 2n - 18 \\
&< 31(n-3)^{\lg 3} - 6n = \Theta(n^{\lg 3})
\end{aligned}
$$

Conclusões

$R(n)$ é $\Theta(n^{\lg 3})$.

Logo $T(n)$ é $\Theta(n^{\lg 3})$.

O consumo de tempo do algoritmo KARATSUBA é $\Theta(n^{\lg 3})$ ($1{,}584 < \lg 3 < 1{,}585$).

Mais conclusões

Consumo de tempo de
algoritmos para multiplicação de inteiros:

Jardim de infância	$\Theta(n\,10^n)$
Ensino fundamental	$\Theta(n^2)$
Karatsuba e Ofman'60	$\mathrm{O}(n^{1.585})$
Toom e Cook'63	$\mathrm{O}(n^{1.465})$
(divisão e conquista; generaliza o acima)	
Schönhage e Strassen'71	$\mathrm{O}(n \lg n \lg \lg n)$
(FFT em aneis de tamanho específico)	
Fürer'07	$\mathrm{O}(n \lg n\, 2^{\mathrm{O}(\log^* n)})$

Ambiente experimental

A plataforma utilizada nos experimentos é um PC rodando Linux Debian ?.? com um processador Pentium II de 233 MHz e 128MB de memória RAM .

Os códigos estão compilados com o gcc versão 2.7.2.1 e opção de compilação -O2.

As implementações comparadas neste experimento são as do algoritmo do ensino fundamental e do algoritmo KARATSUBA.

O programa foi escrito por Carl Burch:
`http://www-2.cs.cmu.edu/~cburch/251/karat/`.

Resultados experimentais

n	Ensino Fund.	KARATSUBA
4	0.005662	0.005815
8	0.010141	0.010600
16	0.020406	0.023643
32	0.051744	0.060335
64	0.155788	0.165563
128	0.532198	0.470810
256	1.941748	1.369863
512	7.352941	4.032258

Tempos em 10^3 segundos.

Multiplicação de matrizes

Problema: Dadas duas matrizes $X[1..n, 1..n]$ e $Y[1..n, 1..n]$ calcular o produto $X \cdot Y$.

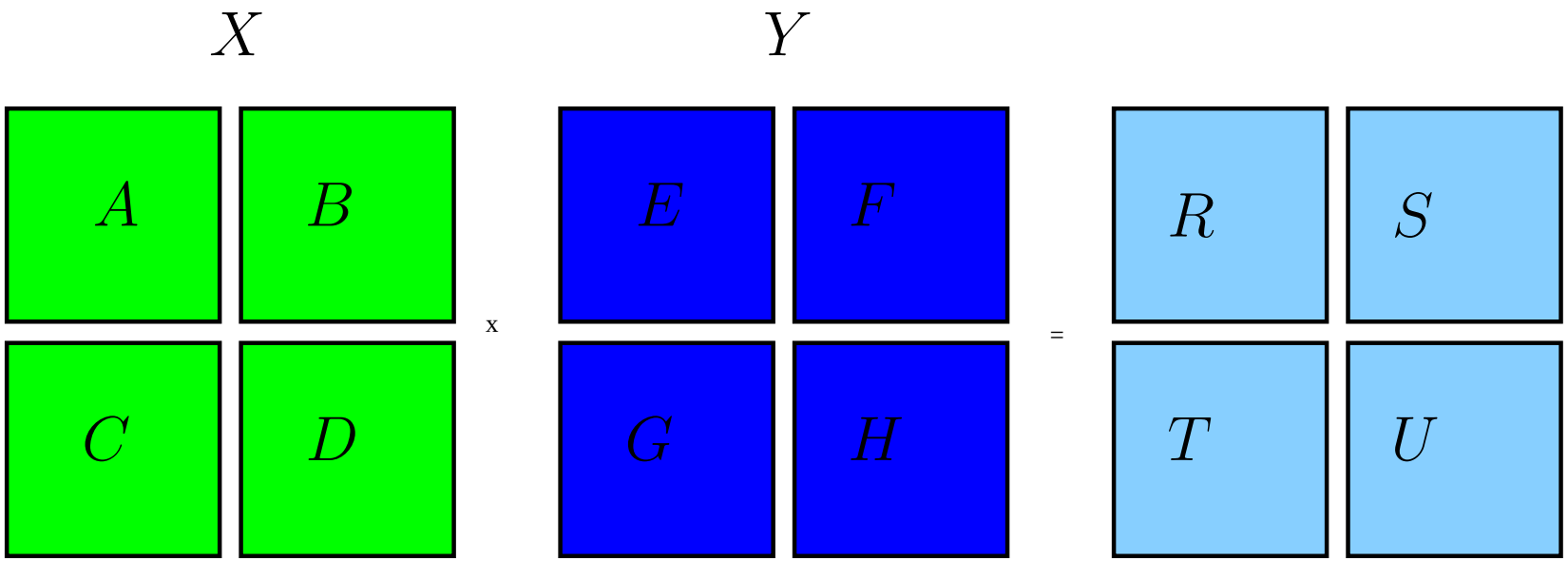
Os algoritmo tradicional de multiplicação de matrizes consome tempo $\Theta(n^3)$.

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix} \times \begin{pmatrix} e & f \\ g & h \end{pmatrix} = \begin{pmatrix} r & s \\ t & u \end{pmatrix}$$

$$\begin{aligned} r &= ae + bg \\ s &= af + bh \\ t &= ce + dg \\ u &= cf + dh \end{aligned} \tag{1}$$

Solução custa R$ 8,04

Divisão e conquista



$$
\begin{aligned}
R &= AE + BG \\
S &= AF + BH \\
T &= CE + DG \\
U &= CF + DH
\end{aligned}
\tag{2}
$$

Algoritmo de Multi-Mat

Algoritmo recebe inteiros $X[1\,..\,n]$ e $Y[1\,..\,n]$ e devolve $X \cdot Y$.

MULTI-M (X, Y, n)

1 **se** $n = 1$ **devolva** $X \cdot Y$
2 $(A, B, C, D) \leftarrow$ PARTICIONE(X, n)
3 $(E, F, G, H) \leftarrow$ PARTICIONE(Y, n)
4 $R \leftarrow$ MULTI-M$(A, E, n/2) +$ MULTI-M$(B, G, n/2)$
5 $S \leftarrow$ MULTI-M$(A, F, n/2) +$ MULTI-M$(B, H, n/2)$
6 $T \leftarrow$ MULTI-M$(C, E, n/2) +$ MULTI-M$(D, G, n/2)$
7 $U \leftarrow$ MULTI-M$(C, F, n/2) +$ MULTI-M$(D, H, n/2)$
8 $P \leftarrow$ CONSTRÓI-MAT(R, S, T, U)
9 **devolva** P

$T(n) =$ consumo de tempo do algoritmo para multiplicar duas matrizes de n linhas e n colunas.

Consumo de tempo

linha		todas as execuções da linha
1	$=$	$\Theta(1)$
2	$=$	$\Theta(n^2)$
3	$=$	$\Theta(n^2)$
4	$=$	$T(n/2) + T(n/2)$
5	$=$	$T(n/2) + T(n/2)$
6	$=$	$T(n/2) + T(n/2)$
7	$=$	$T(n/2) + T(n/2)$
8	$=$	$\Theta(n^2)$
9	$=$	$\Theta(n^2)$
total	$=$	$8\,T(n/2) + \Theta(n^2)$

Consumo de tempo

As dicas no nosso estudo de recorrências sugeri que a solução da recorrência

$$T(n) \quad = \quad 8\,T(n/2) + \Theta(n^2)$$

está na mesma classe Θ que a solução de

$$\begin{aligned} T'(1) &= 1 \\ T'(n) &= 8T'(n/2) + n^2 \quad \text{para } n = 2, 2^2, 2^3, \ldots \end{aligned}$$

n	1	2	4	8	16	32	64	128	256
$T'(n)$	1	12	112	960	7936	64512	520192	4177920	33488896

Solução assintótica da recorrência

Considere a recorrência

$$\begin{aligned} R(1) &= 1 \\ R(n) &= 8\,R(\lceil \tfrac{n}{2} \rceil) + n^2 \quad \text{para } n = 2, 3, 4, \ldots \end{aligned}$$

Verifique por indução que $R(n) \leq 20(n-1)^3 - 2n^2$ para $n = 2, 3, 4 \ldots$.

n	1	2	3	4	5	6	7	8
$R(n)$	1	12	105	112	865	876	945	960
$20\,(n-1)^3 - 2n^2$	-2	12	142	508	1230	2428	4222	6732

Conclusões

$R(n)$ é $\Theta(n^3)$.

Conclusão anterior+Exercício $\Rightarrow$
$T(n)$ é $\Theta(n^3)$.

O consumo de tempo do algoritmo MULTI-M é
$\Theta(n^3)$.

Strassen: $X \cdot Y$ por apenas R$ 7,18

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix} \times \begin{pmatrix} e & f \\ g & h \end{pmatrix} = \begin{pmatrix} r & s \\ t & u \end{pmatrix}$$

Strassen: $X \cdot Y$ por apenas R$ 7,18

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix} \times \begin{pmatrix} e & f \\ g & h \end{pmatrix} = \begin{pmatrix} r & s \\ t & u \end{pmatrix}$$

$$\begin{aligned}
p_1 &= a(f-h) = af - ah \\
p_2 &= (a+b)h = ah + bh \\
p_3 &= (c+d)e = ce + de \\
p_4 &= d(g-e) = dg - de \\
p_5 &= (a+d)(e+h) = ae + ah + de + dh \\
p_6 &= (b-d)(g+h) = bg + bh - dg - dh \\
p_7 &= (a-c)(e+f) = ae + af - ce - cfd
\end{aligned} \tag{4}$$

Strassen: $X \cdot Y$ por apenas R$ 7,18

$$p_1 = a(f - h) = af - ah$$

$$p_2 = (a + b)h = ah + bh$$

$$p_3 = (c + d)e = ce + de$$

$$p_4 = d(g - e) = dg - de$$

$$p_5 = (a + d)(e + h) = ae + ah + de + dh$$

$$p_6 = (b - d)(g + h) = bg + bh - dg - dh$$

$$p_7 = (a - c)(e + f) = ae + af - ce - cfd$$

$$r = p_5 + p_4 - p_2 + p_6 = ae + bg$$

$$s = p_1 + p_2 = af + bh$$

$$t = p_3 + p_4 = ce + dg$$

$$u = p_5 + p_1 - p_3 - p_7 = cf + dh$$

Algoritmo de Strassen

STRASSEN (X, Y, n)

se $n = 1$ **devolva** $X \cdot Y$
$(A, B, C, D) \leftarrow$ PARTICIONE(X, n)
$(E, F, G, H) \leftarrow$ PARTICIONE(Y, n)
$P_1 \leftarrow$ STRASSEN$(A, F - H, n/2)$
$P_2 \leftarrow$ STRASSEN$(A + B, H, n/2)$
$P_3 \leftarrow$ STRASSEN$(C + D, E, n/2)$
$P_4 \leftarrow$ STRASSEN$(D, G - E, n/2)$
$P_5 \leftarrow$ STRASSEN$(A + D, E + H, n/2)$
$P_6 \leftarrow$ STRASSEN$(B - D, G + H, n/2)$
$P_7 \leftarrow$ STRASSEN$(A - C, E + F, n/2)$
$R \leftarrow P_5 + P_4 - P_2 + P_6$
$S \leftarrow P_1 + P_2$
$T \leftarrow P_3 + P_4$
$U \leftarrow P_5 + P_1 - P_3 - P_7$
devolva $P \leftarrow$ CONSTRÓI-MAT(R, S, T, U)

Consumo de tempo

linha		todas as execuções da linha
1	$=$	$\Theta(1)$
2-3	$=$	$\Theta(n^2)$
4-10	$=$	$7, T(n/2) + \Theta(n^2)$
11-14	$=$	$\Theta(n^2)$
15	$=$	$\Theta(n^2)$
total	$=$	$7\,T(n/2) + \Theta(n^2)$

Consumo de tempo

As dicas no nosso estudo de recorrências sugeri que a solução da recorrência

$$T(n) \quad = \quad 7\,T(n/2) + \Theta(n^2)$$

está na mesma classe Θ que a solução de

$$\begin{aligned} T'(1) &= 1 \\ T'(n) &= 7T'(n/2) + n^2 \quad \text{para } n = 2, 2^2, 2^3, \ldots \end{aligned}$$

n	1	2	4	8	16	32	64	128	256
$T'(n)$	1	11	93	715	5261	37851	269053	1899755	13363821

Solução assintótica da recorrência

Considere a recorrência

$$\begin{aligned} R(1) &= 1 \\ R(n) &= 7\,R(\lceil \tfrac{n}{2} \rceil) + n^2 \quad \text{para } n = 2, 3, 4, \ldots \end{aligned}$$

Verifique por indução que $R(n) \leq 19(n-1)^{\lg 7} - 2n^2$ para $n = 2, 3, 4 \ldots$.

$2{,}80 < \lg 7 < 2{,}81$

n	1	2	3	4	5	6	7	8
$R(n)$	1	11	86	93	627	638	700	715
$19\,(n-1)^{\lg 7} - 2n^2$	-1	11	115	327	881	1657	2790	4337

Conclusões

$R(n)$ é $\Theta(n^{\lg 7})$.

$T(n)$ é $\Theta(n^{\lg 7})$.

O consumo de tempo do algoritmo STRASSEN é $\Theta(n^{\lg 7})$ ($2{,}80 < \lg 7 < 2{,}81$).

Mais conclusões

Consumo de tempo de algoritmos para multiplicação de matrizes:

Ensino fundamental	$\Theta(n^3)$
Strassen	$\Theta(n^{2.81})$
...	...
Coppersmith e Winograd	$\Theta(n^{2.38})$

Análise de Algoritmos

Slides de Paulo Feofiloff

[com erros do coelho e agora também da cris]

Análise probabilística

CLRS 5.1, C.2, C.3

Máximo

Problema: Encontrar o elemento máximo de um vetor $A[1\,..\,n]$ de números inteiros positivos distintos.

```
MAX (A, n)
1    max ← 0
2    para i ← 1 até n faça
3         se A[i] > max
4              então max ← A[i]
5    devolva max
```

Quantas vezes a linha 4 é executada?

Máximo

Problema: Encontrar o elemento máximo de um vetor $A[1\,..\,n]$ de números inteiros positivos distintos.

```
MAX (A, n)
1    max ← 0
2    para i ← 1 até n faça
3         se A[i] > max
4              então max ← A[i]
5    devolva max
```

Quantas vezes a linha 4 é executada?
Melhor caso, pior caso, caso médio?

Máximo

Problema: Encontrar o elemento máximo de um vetor $A[1\,..\,n]$ de números inteiros positivos distintos.

```
MAX (A, n)
1    max ← 0
2    para i ← 1 até n faça
3         se A[i] > max
4              então max ← A[i]
5    devolva max
```

Quantas vezes a linha 4 é executada?
Melhor caso, pior caso, caso médio?

Suponha que $A[1\,..\,n]$ é permutação aleatória uniforme de $1, \ldots, n$

Cada permutação tem probabilidade $1/n!$.

Um pouco de probabilidade

$(S, \Pr)$ espaço de probabilidade
$S =$ conjunto finito (eventos elementares)
$\Pr\{\} =$ (distribuição de probabilidades) função de S em $[0, 1]$ tal que

p1. $\Pr\{s\} \geq 0$;

p2. $\Pr\{S\} = 1$; e

p3. $R, T \subseteq S, R \cap T = \emptyset \Rightarrow \Pr\{R \cup T\} = \Pr\{R\} + \Pr\{T\}$.

$\Pr\{U\}$ é abreviação de $\displaystyle\sum_{u \in U} \Pr\{u\}$.

Um pouco de probabilidade

$(S, \Pr)$ espaço de probabilidade
$S =$ conjunto finito (eventos elementares)
$\Pr\{\} =$ (distribuição de probabilidades) função de S em $[0, 1]$ tal que

p1. $\Pr\{s\} \geq 0$;

p2. $\Pr\{S\} = 1$; e

p3. $R, T \subseteq S, R \cap T = \emptyset \Rightarrow \Pr\{R \cup T\} = \Pr\{R\} + \Pr\{T\}$.

$$\Pr\{U\} \text{ é abreviação de } \sum_{u \in U} \Pr\{u\}.$$

No problema do máximo:

- S é o conjunto das permutações dos números em $A[1\,..\,n]$;
- na distribuição uniforme, para cada $s \in S$, $\Pr\{s\} = 1/n!$.

Mais um pouco de probabilidade

Um evento é um subconjunto de S.

Mais um pouco de probabilidade

Um evento é um subconjunto de S.

No problema do máximo, eventos são subconjuntos de permutações de $A[1\,..\,n]$.

Exemplo.

$$U := \{\text{permutações de } A[1\,..\,n] \text{ em que } A[n] \text{ é máximo}\}$$

é um evento de S.

Mais um pouco de probabilidade

Um evento é um subconjunto de S.

No problema do máximo, eventos são subconjuntos de permutações de $A[1\,..\,n]$.

Exemplo.

$$U := \{\text{permutações de } A[1\,..\,n] \text{ em que } A[n] \text{ é máximo}\}$$

é um evento de S.

Se $\Pr\{\}$ é distribuição uniforme, então

$$\Pr\{U\} = ???.$$

Mais um pouco de probabilidade

Um evento é um subconjunto de S.

No problema do máximo, eventos são subconjuntos de permutações de $A[1\,..\,n]$.

Exemplo.

$$U := \{\text{permutações de } A[1\,..\,n] \text{ em que } A[n] \text{ é máximo}\}$$

é um evento de S.

Se $\Pr\{\}$ é distribuição uniforme, então

$$\Pr\{U\} = 1/n.$$

Mais um pouco de probabilidade

Uma variável aleatória é uma função númerica definida sobre os eventos elementares.

Mais um pouco de probabilidade

Uma variável aleatória é uma função númerica definida sobre os eventos elementares.

Exemplo de variável aleatória

$$X(A) := \text{número de execuções da linha 4 em MAX}(A, n)$$

Mais um pouco de probabilidade

Uma variável aleatória é uma função númerica definida sobre os eventos elementares.

Exemplo de variável aleatória

$$X(A) := \text{número de execuções da linha 4 em MAX}(A, n)$$

"$X = k$" é uma abreviação de $\{s \in S : X(s) = k\}$

Esperança $\mathrm{E}[X]$ de uma variável aleatória X

$$\mathrm{E}[X] = \sum_{k \in X(S)} k \cdot \Pr\{X = k\} = \sum_{s \in S} X(s) \cdot \Pr\{s\}$$

Mais um pouco de probabilidade

Uma variável aleatória é uma função númerica definida sobre os eventos elementares.

Exemplo de variável aleatória

$$X(A) := \text{número de execuções da linha 4 em MAX}(A, n)$$

"$X = k$" é uma abreviação de $\{s \in S : X(s) = k\}$

Esperança $\mathrm{E}[X]$ de uma variável aleatória X

$$\mathrm{E}[X] = \sum_{k \in X(S)} k \cdot \Pr\{X = k\} = \sum_{s \in S} X(s) \cdot \Pr\{s\}$$

Linearidade da esperança: $\mathrm{E}[\alpha X + Y] = \alpha \mathrm{E}[X] + \mathrm{E}[Y]$

De volta ao máximo

Problema: Encontrar o elemento máximo de um vetor $A[1\,..\,n]$ de números inteiros distintos.

```
MAX (A, n)
1    max ← 0
2    para i ← 1 até n faça
3         se A[i] > max
4              então max ← A[i]
5    devolva max
```

Quantas vezes a linha 4 é executada no caso médio?

Suponha que $A[1\,..\,n]$ é permutação aleatória uniforme de $1, \ldots, n$

Cada permutação tem probabilidade $1/n!$.

Exemplos

$A[1\,..\,2]$	linha 4
1,2	2
2,1	1
$\mathrm{E}[X]$	3/2

$A[1\,..\,3]$	linha 4
1,2,3	3
1,3,2	2
2,1,3	2
2,3,1	2
3,1,2	1
3,2,1	1
$\mathrm{E}[X]$	11/6

Mais um exemplo

$A[1\,..\,4]$	linha 4
1,2,3,4	4
1,2,4,3	3
1,3,2,4	3
1,3,4,2	3
1,4,2,3	2
1,4,3,2	2
2,1,3,4	3
2,1,4,3	2
2,3,1,4	3
2,3,4,1	3
2,4,1,3	2
2,4,3,1	2

$A[1\,..\,4]$	linha 14
3,1,2,4	2
3,1,4,2	2
3,2,1,4	2
3,2,4,1	2
3,4,1,2	2
3,4,2,1	2
4,1,2,3	1
4,1,3,2	1
4,2,1,3	1
4,2,3,1	1
4,3,1,2	1
4,3,2,1	1

$\mathrm{E}[X]$ 50/24

Variáveis aleatórias

X = número total de execuções da linha 4

Variáveis aleatórias

$$X = \text{número total de execuções da linha 4}$$

$$X_i = \begin{cases} 1 \text{ se “} max \leftarrow A[i] \text{” é executado} \\ 0 \text{ caso contrário} \end{cases}$$

$$\begin{aligned} X &= \text{número total de execuções da linha 4} \\ &= X_1 + \cdots + X_n \end{aligned}$$

Variáveis aleatórias

$$X = \text{número total de execuções da linha 4}$$

$$X_i = \begin{cases} 1 \text{ se “}max \leftarrow A[i]\text{” é executado} \\ 0 \text{ caso contrário} \end{cases}$$

$$\begin{aligned} X &= \text{número total de execuções da linha 4} \\ &= X_1 + \cdots + X_n \end{aligned}$$

Esperanças:

$$\begin{aligned} \mathrm{E}[X_i] &= \text{probabilidade de que } A[i] \text{ seja} \\ &\quad\ \text{máximo em } A[1\,..\,i] \\ &= 1/i \end{aligned}$$

Esperança

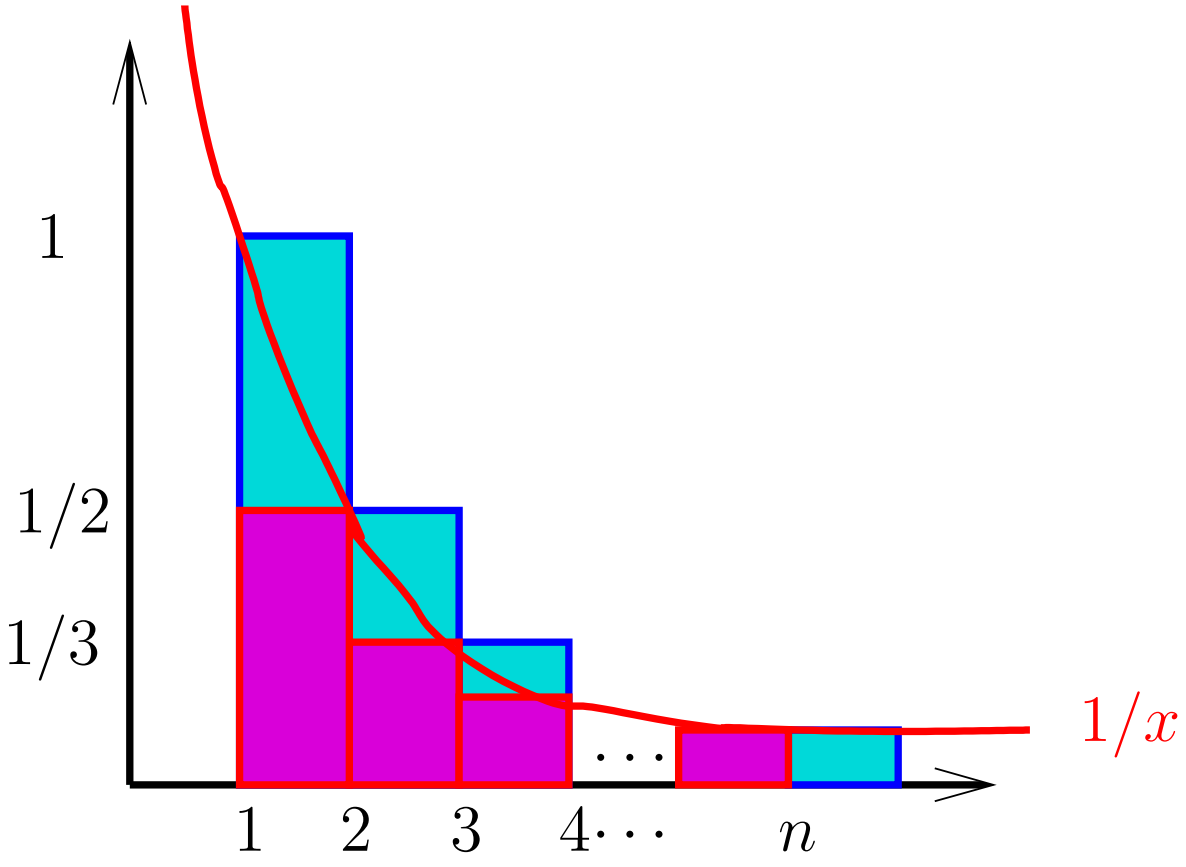
$$
\begin{aligned}
\mathrm{E}[X] &= \mathrm{E}[X_1 + \cdots + X_n] \\
&= \mathrm{E}[X_1] + \cdots + \mathrm{E}[X_n] \\
&= 1/1 + \cdots + 1/n \\
&< 1 + \ln n \\
&= \Theta(\lg n)
\end{aligned}
$$

$$2.92 < \frac{1}{1} + \cdots + \frac{1}{10} < 2.93 < 3.30 < 1 + \ln 10$$

$$5.18 < \frac{1}{1} + \cdots + \frac{1}{100} < 5.19 < 6.60 < 1 + \ln 100$$

$$9.78 < \frac{1}{1} + \cdots + \frac{1}{10000} < 9.79 < 10.21 < 1 + \ln 10000$$

Série harmônica



$$\ln n = \int_1^n \frac{dx}{x} < H_n = 1 + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \cdots + \frac{1}{n}$$

$$< 1 + \int_1^n \frac{dx}{x} = 1 + \ln n.$$

Experimentos

Para cada valor de $n = 252, 512, 1024, \ldots$ foram geradas 10, 100 ou 200 amostras de seqüencias de inteiros através do trecho de código

```
for (i = 0; i < n; i++){
  v[i]=(int)((double)INT_MAX*rand()/(RAND_MAX+1
}
```

onde `rand()` é a função geradora de números (pseudo-)aleatórios da biblioteca do `C`.

A coluna $\mathrm{E}[\hat{X}]$ nas tabelas a seguir mostra o número médio de vezes que a linha 4 do algoritmo MAX foi executada para cada valor de n e cada amostra de seqüências.

Experimentos (10)

n	$E[\hat{X}]$	$1 + \ln n$
256	7.20	6.55
512	6.90	7.24
1024	7.30	7.93
2048	7.10	8.62
4096	10.20	9.32
8192	9.00	10.01
16384	10.80	10.70
32768	11.00	11.40
65536	12.50	12.09
131072	12.60	12.78
262144	13.20	13.48
524288	13.20	14.17
1048576	12.80	14.86
2097152	13.90	15.56
4194304	14.90	16.25
8388608	17.90	16.94

Experimentos (100)

n	$E[\hat{X}]$	$1 + \ln n$
256	5.92	6.55
512	6.98	7.24
1024	7.55	7.93
2048	8.39	8.62
4096	8.97	9.32
8192	9.26	10.01
16384	10.44	10.70
32768	11.32	11.40
65536	11.66	12.09
131072	12.38	12.78
262144	13.17	13.48
524288	13.56	14.17
1048576	14.54	14.86
2097152	15.10	15.56
4194304	15.61	16.25
8388608	16.56	16.94

Experimentos (200)

n	$E[\hat{X}]$	$1 + \ln n$
256	6.12	6.55
512	6.86	7.24
1024	7.38	7.93
2048	7.96	8.62
4096	8.87	9.32
8192	9.41	10.01
16384	10.28	10.70
32768	10.92	11.40
65536	11.31	12.09
131072	12.37	12.78
262144	12.92	13.48
524288	13.98	14.17
1048576	14.19	14.86
2097152	15.62	15.56
4194304	15.74	16.25
8388608	17.06	16.94

Quicksort

CLRS 7

Partição

Problema: Rearranjar um dado vetor $A[p\,..\,d]$ e devolver um índice q tal que $p \leq q \leq d$ e

$$A[p\,..\,q-1] \leq A[q] < A[q+1\,..\,d]$$

Entra:

	p									d
A	99	33	55	77	11	22	88	66	33	44

Partição

Problema: Rearranjar um dado vetor $A[p\,..\,d]$ e devolver um índice q tal que $p \leq q \leq d$ e

$$A[p\,..\,q-1] \leq A[q] < A[q+1\,..\,d]$$

Entra:

	p									d
A	99	33	55	77	11	22	88	66	33	44

Sai:

	p				q					d
A	33	11	22	33	44	55	99	66	77	88

Particione

	p									d
A	99	33	55	77	11	22	88	66	33	44

Particione

	i	j								x
A	99	33	55	77	11	22	88	66	33	44

Particione

	i	j								x
A	99	33	55	77	11	22	88	66	33	44

Particione

	i		j							x
A	99	33	55	77	11	22	88	66	33	44

	i		j							x
A	33	99	55	77	11	22	88	66	33	44

Particione

	i		j							x
A	99	33	55	77	11	22	88	66	33	44
	i		j							x
A	33	99	55	77	11	22	88	66	33	44
	i			j						x
A	33	99	55	77	11	22	88	66	33	44

Particione

	i		j							x
A	99	33	55	77	11	22	88	66	33	44
	i		j							x
A	33	99	55	77	11	22	88	66	33	44
	i				j					x
A	33	99	55	77	11	22	88	66	33	44

Particione

	i		j							x
A	99	33	55	77	11	22	88	66	33	44
	i		j							x
A	33	99	55	77	11	22	88	66	33	44
		i				j				x
A	33	11	55	77	99	22	88	66	33	44

Particione

	i		j							x
A	99	33	55	77	11	22	88	66	33	44
	i		j							x
A	33	99	55	77	11	22	88	66	33	44
		i				j				x
A	33	11	55	77	99	22	88	66	33	44
			i				j			x
A	33	11	22	77	99	55	88	66	33	44

Particione

	i		j							x
A	99	33	55	77	11	22	88	66	33	44
	i		j							x
A	33	99	55	77	11	22	88	66	33	44
		i				j				x
A	33	11	55	77	99	22	88	66	33	44
			i				j			x
A	33	11	22	77	99	55	88	66	33	44
			i					j		x
A	33	11	22	77	99	55	88	66	33	44

Particione

	i		j							x
A	99	33	55	77	11	22	88	66	33	44
		i		j						x
A	33	99	55	77	11	22	88	66	33	44
			i				j			x
A	33	11	55	77	99	22	88	66	33	44
				i				j		x
A	33	11	22	77	99	55	88	66	33	44
				i					j	x
A	33	11	22	77	99	55	88	66	33	44

Particione

	i		j							x
A	99	33	55	77	11	22	88	66	33	44
	i		j							x
A	33	99	55	77	11	22	88	66	33	44
		i				j				x
A	33	11	55	77	99	22	88	66	33	44
			i				j			x
A	33	11	22	77	99	55	88	66	33	44
				i						j
A	33	11	22	33	99	55	88	66	77	44

Particione

	i		j							x
A	99	33	55	77	11	22	88	66	33	44
	i		j							x
A	33	99	55	77	11	22	88	66	33	44
		i				j				x
A	33	11	55	77	99	22	88	66	33	44
			i				j			x
A	33	11	22	77	99	55	88	66	33	44
				i						j
A	33	11	22	33	99	55	88	66	77	44
	p				q					d
A	33	11	22	33	44	55	88	66	77	99

Particione

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

PARTICIONE (A, p, d)

1 $x \leftarrow A[d]$ $\quad\triangleright$ x é o "pivô"
2 $i \leftarrow p{-}1$
3 **para** $j \leftarrow p$ **até** $d - 1$ **faça**
4 **se** $A[j] \le x$
5 **então** $i \leftarrow i + 1$
6 $A[i] \leftrightarrow A[j]$
7 $A[i{+}1] \leftrightarrow A[d]$
8 **devolva** $i + 1$

Invariantes: no começo de cada iteração de 3–6,
(i0) $A[p\,..\,i] \le x$ (i1) $A[i{+}1\,..\,j{-}1] > x$ (i2) $A[d] = x$

Consumo de tempo

Quanto tempo consome em função de $n := d - p + 1$?

linha	consumo de todas as execuções da linha
1-2	$= 2\,\Theta(1)$
3	$= \Theta(n)$
4	$= \Theta(n)$
5-6	$= 2\,\mathrm{O}(n)$
7-8	$= 2\,\Theta(1)$
total	$= \Theta(2n+4) + \mathrm{O}(2n) \qquad = \Theta(n)$

Conclusão:

O algoritmo PARTICIONE consome tempo $\Theta(n)$.

Quicksort

Rearranja $A[p\,..\,d]$ em ordem crescente.

```
QUICKSORT (A, p, d)
1    se p < d
2        então q ← PARTICIONE (A, p, d)
3              QUICKSORT (A, p, q − 1)
4              QUICKSORT (A, q + 1, d)
```

	p									d
A	99	33	55	77	11	22	88	66	33	44

Quicksort

Rearranja $A[p\,..\,d]$ em ordem crescente.

QUICKSORT (A, p, d)

1 **se** $p < d$

2 **então** $q \leftarrow$ PARTICIONE (A, p, d)

3 QUICKSORT $(A, p, q-1)$

4 QUICKSORT $(A, q+1, d)$

	p				q					d
A	33	11	22	33	44	55	88	66	77	99

No começo da linha 3,

$$A[p\,..\,q{-}1] \leq A[q] \leq A[q{+}1\,..\,d]$$

Quicksort

Rearranja $A[p\,..\,d]$ em ordem crescente.

QUICKSORT (A, p, d)

1 **se** $p < d$

2 **então** $q \leftarrow$ PARTICIONE (A, p, d)

3 QUICKSORT $(A, p, q-1)$

4 QUICKSORT $(A, q+1, d)$

	p				q					d
A	11	22	33	33	44	55	88	66	77	99

Quicksort

Rearranja $A[p\,..\,d]$ em ordem crescente.

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

	p				q					d
A	11	22	33	33	44	55	66	77	88	99

Quicksort

Rearranja $A[p\,..\,d]$ em ordem crescente.

QUICKSORT (A, p, d)
1 **se** $p < d$
2 **então** $q \leftarrow$ PARTICIONE (A, p, d)
3 QUICKSORT $(A, p, q-1)$
4 QUICKSORT $(A, q+1, d)$

No começo da linha 3,

$$A[p\,..\,q{-}1] \leq A[q] \leq A[q{+}1\,..\,d]$$

Consumo de tempo?

Quicksort

Rearranja $A[p\,..\,d]$ em ordem crescente.

```
QUICKSORT (A, p, d)
1    se p < d
2         então q ← PARTICIONE (A, p, d)
3               QUICKSORT (A, p, q − 1)
4               QUICKSORT (A, q + 1, d)
```

No começo da linha 3,

$$A[p\,..\,q{-}1] \leq A[q] \leq A[q{+}1\,..\,d]$$

Consumo de tempo?

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

$$n := d - p + 1$$

Consumo de tempo

Quanto tempo consome em função de $n := d - p + 1$?

linha		consumo de todas as execuções da linha
1	$=$	?
2	$=$	?
3	$=$	?
4	$=$	?
total	$=$	????

Consumo de tempo

Quanto tempo consome em função de $n := d - p + 1$?

linha		consumo de todas as execuções da linha
1	$=$	$\Theta(1)$
2	$=$	$\Theta(n)$
3	$=$	$T(k)$
4	$=$	$T(n-k-1)$
total	$=$	$T(k) + T(n-k-1) + \Theta(n+1)$

$$0 \leq k := q - p \leq n - 1$$

Recorrência

$T(n) :=$ consumo de tempo máximo quando $n = d - p + 1$

$$T(n) = T(k) + T(n - k - 1) + \Theta(n)$$

Recorrência

$T(n) :=$ consumo de tempo máximo quando $n = d - p + 1$

$$T(n) = T(k) + T(n - k - 1) + \Theta(n)$$

Recorrência grosseira:

$$T(n) = T(0) + T(n - 1) + \Theta(n)$$

$T(n)$ é $\Theta(???)$.

Recorrência

$T(n) :=$ consumo de tempo máximo quando $n = d - p + 1$

$$T(n) = T(k) + T(n - k - 1) + \Theta(n)$$

Recorrência grosseira:

$$T(n) = T(0) + T(n - 1) + \Theta(n)$$

$T(n)$ é $\Theta(n^2)$.

Demonstração: ... Exercício!

Recorrência cuidadosa

$T(n) :=$ consumo de tempo máximo quando $n = d - p + 1$

$$T(n) = \max_{0 \le k \le n-1} \{T(k) + T(n-k-1)\} + \Theta(n)$$

Recorrência cuidadosa

$T(n) :=$ consumo de tempo máximo quando $n = d - p + 1$

$$T(n) = \max_{0 \le k \le n-1} \{T(k) + T(n-k-1)\} + \Theta(n)$$

Versão simplificada:

$$T(0) = 1$$

$$T(1) = 1$$

$$T(n) = \max_{0 \le k \le n-1} \{T(k) + T(n-k-1)\} + n \quad \text{para } n = 2, 3, 4, \ldots$$

n	0	1	2	3	4	5
$T(n)$	1	1	$2+2$	$5+3$	$9+4$	$14+5$

Recorrência cuidadosa

$T(n) :=$ consumo de tempo máximo quando $n = d - p + 1$

$$T(n) = \max_{0 \le k \le n-1} \{T(k) + T(n-k-1)\} + \Theta(n)$$

Versão simplificada:

$$T(0) = 1$$

$$T(1) = 1$$

$$T(n) = \max_{0 \le k \le n-1} \{T(k) + T(n-k-1)\} + n \quad \text{para } n = 2, 3, 4, \ldots$$

n	0	1	2	3	4	5
$T(n)$	1	1	$2+2$	$5+3$	$9+4$	$14+5$

Vamos mostrar que $T(n) \le n^2 + 1$ para $n \ge 0$.

Demonstração

Prova: Trivial para $n \leq 1$. Se $n \geq 2$ então

$$
\begin{aligned}
T(n) &= \max_{0 \leq k \leq n-1} \left\{ T(k) + T(n-k-1) \right\} + n \\
&\overset{\text{hi}}{\leq} \max_{0 \leq k \leq n-1} \left\{ k^2 + 1 + (n-k-1)^2 + 1 \right\} + n \\
&= \cdots \\
&= n^2 - n + 3 \\
&\leq n^2 + 1 \,.
\end{aligned}
$$

Prove que $T(n) \geq \frac{1}{2}\, n^2$ para $n \geq 1$.

Algumas conclusões

$T(n)$ é $\Theta(n^2)$.

O consumo de tempo do QUICKSORT no pior caso é $\mathrm{O}(n^2)$.

O consumo de tempo do QUICKSORT é $\mathrm{O}(n^2)$.

Quicksort no melhor caso

$M(n) :=$ consumo de tempo mínimo quando $n = d - p + 1$

$$M(n) = \min_{0 \leq k \leq n-1} \{M(k) + M(n-k-1)\} + \Theta(n)$$

Quicksort no melhor caso

$M(n) :=$ consumo de tempo mínimo quando $n = d - p + 1$

$$M(n) = \min_{0 \le k \le n-1} \{M(k) + M(n-k-1)\} + \Theta(n)$$

Versão simplificada:

$$M(0) = 1$$

$$M(1) = 1$$

$$M(n) = \min_{0 \le k \le n-1} \{M(k) + M(n-k-1)\} + n \quad \text{para } n = 2, 3, 4, \ldots$$

Quicksort no melhor caso

$M(n) :=$ consumo de tempo mínimo quando $n = d - p + 1$

$$M(n) = \min_{0 \le k \le n-1} \{M(k) + M(n-k-1)\} + \Theta(n)$$

Versão simplificada:

$$M(0) = 1$$

$$M(1) = 1$$

$$M(n) = \min_{0 \le k \le n-1} \{M(k) + M(n-k-1)\} + n \quad \text{para } n = 2, 3, 4, \ldots$$

Mostre que $M(n) \ge (n+1)\lg(n+1)$ para $n \ge 1$.

Quicksort no melhor caso

$M(n) :=$ consumo de tempo mínimo quando $n = d - p + 1$

$$M(n) = \min_{0 \le k \le n-1} \{M(k) + M(n-k-1)\} + \Theta(n)$$

Versão simplificada:

$$M(0) = 1$$

$$M(1) = 1$$

$$M(n) = \min_{0 \le k \le n-1} \{M(k) + M(n-k-1)\} + n \quad \text{para } n = 2, 3, 4, \ldots$$

Mostre que $M(n) \ge (n+1)\lg(n+1)$ para $n \ge 1$.

Isto implica que no melhor caso o QUICKSORT é $\Omega(n \lg n)$,

Quicksort no melhor caso

$M(n) :=$ consumo de tempo mínimo quando $n = d - p + 1$

$$M(n) = \min_{0 \le k \le n-1} \{M(k) + M(n-k-1)\} + \Theta(n)$$

Versão simplificada:

$$M(0) = 1$$

$$M(1) = 1$$

$$M(n) = \min_{0 \le k \le n-1} \{M(k) + M(n-k-1)\} + n \quad \text{para } n = 2, 3, 4, \ldots$$

Mostre que $M(n) \ge (n+1)\lg(n+1)$ para $n \ge 1$.

Isto implica que no melhor caso o QUICKSORT é $\Omega(n \lg n)$, que é o mesmo que dizer que o QUICKSORT é $\Omega(n \lg n)$.

Mais algumas conclusões

$M(n)$ é $\Theta(n \lg n)$.

O consumo de tempo do QUICKSORT no melhor caso é $\Omega(n \log n)$.

Na verdade . . .

O consumo de tempo do QUICKSORT no melhor caso é $\Theta(n \log n)$.

Análise de caso médio do Quicksort

Apesar do consumo de tempo de pior caso do QUICKSORT ser $\Theta(n^2)$, sua performance na prática é comparável (e em geral melhor) a de outros algoritmos cujo consumo de tempo no pior caso é $\mathrm{O}(n \lg n)$.

Por que isso acontece?

Exercício

Considere a recorrência

$$T(1) = 1$$

$$T(n) = T(\lceil n/3 \rceil) + T(\lfloor 2n/3 \rfloor) + n$$

para $n = 2, 3, 4, \ldots$

Solução assintótica: $T(n)$ é $\mathrm{O}(???)$, $T(n)$ é $\Theta(???)$

Exercício

Considere a recorrência

$$T(1) = 1$$

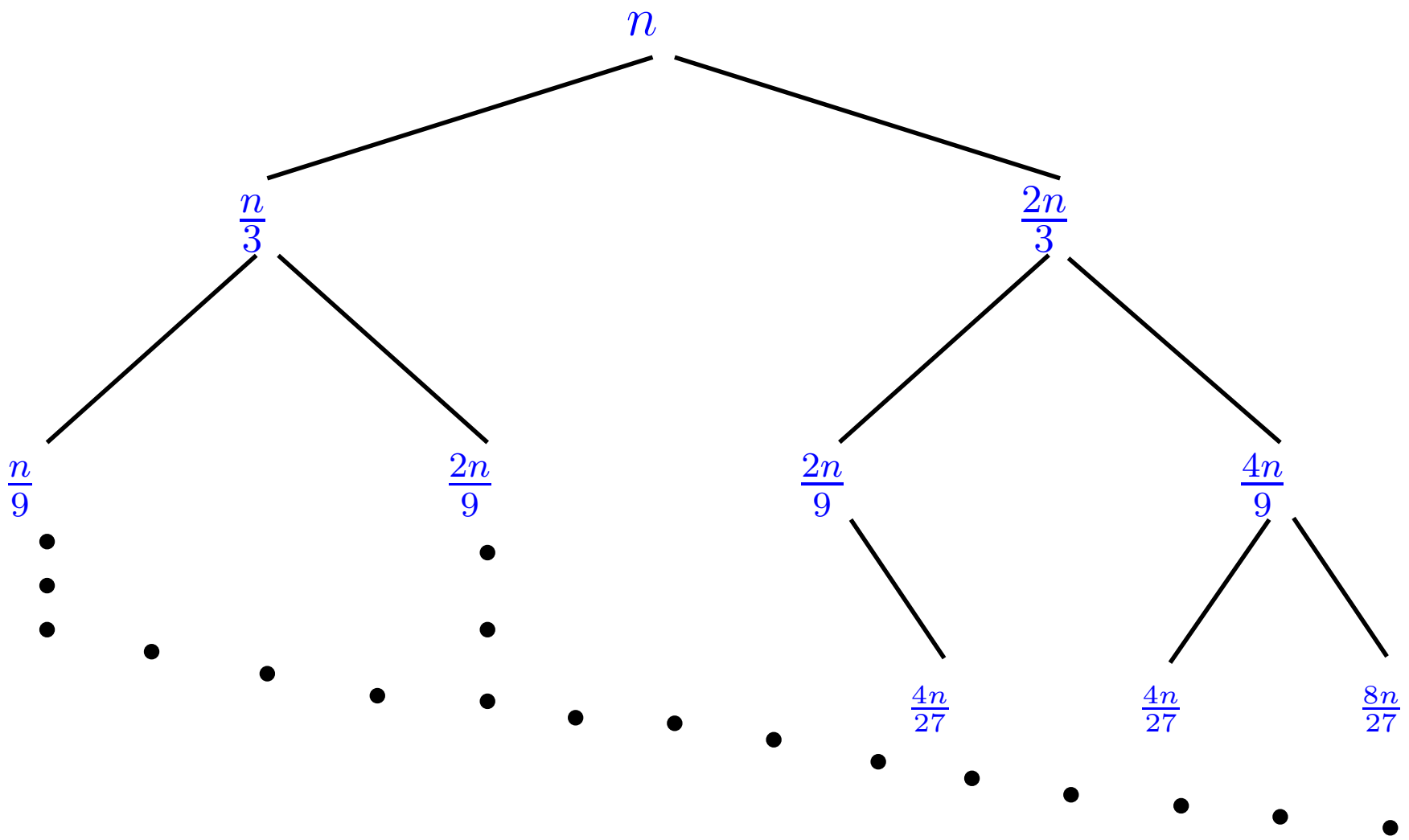
$$T(n) = T(\lceil n/3 \rceil) + T(\lfloor 2n/3 \rfloor) + n$$

para $n = 2, 3, 4, \ldots$

Solução assintótica: $T(n)$ é $\mathrm{O}(???)$, $T(n)$ é $\Theta(???)$

Vamos olhar a árvore da recorrência.

Árvore da recorrência



total de níveis $\leq \log_{3/2} n$

Árvore da recorrência

soma em cada horizontal $\leq n$

número de "níveis" $\leq \log_{3/2} n$

$T(n)$ = a soma de tudo

$$T(n) \;\leq\; n \log_{3/2} n \;+\; \underbrace{1 + \cdots + 1}_{\log_{3/2} n}$$

$T(n)$ é $\mathrm{O}(n \lg n)$.

De volta a recorrência

$$T(1) = 1$$

$$T(n) = T(\lceil n/3 \rceil) + T(\lfloor 2n/3 \rfloor) + n \text{ para } n = 2, 3, 4, \ldots$$

n	$T(n)$
1	1
2	$1 + 1 + 2 = 4$
3	$1 + 4 + 3 = 8$
4	$4 + 4 + 4 = 12$

De volta a recorrência

$$T(1) = 1$$

$$T(n) = T(\lceil n/3 \rceil) + T(\lfloor 2n/3 \rfloor) + n \quad \text{para } n = 2, 3, 4, \ldots$$

n	$T(n)$
1	1
2	$1 + 1 + 2 = 4$
3	$1 + 4 + 3 = 8$
4	$4 + 4 + 4 = 12$

Vamos mostrar que $T(n) \leq 20\, n \lg n$ para $n = 2, 3, 4, 5, 6, \ldots$

De volta a recorrência

$T(1) = 1$

$T(n) = T(\lceil n/3 \rceil) + T(\lfloor 2n/3 \rfloor) + n$ para $n = 2, 3, 4, \ldots$

n	$T(n)$
1	1
2	$1 + 1 + 2 = 4$
3	$1 + 4 + 3 = 8$
4	$4 + 4 + 4 = 12$

Vamos mostrar que $T(n) \leq 20\, n \lg n$ para $n = 2, 3, 4, 5, 6, \ldots$
Para $n = 2$ temos $T(2) = 4 < 20 \cdot 2 \cdot \lg 2$.
Para $n = 3$ temos $T(3) = 8 < 20 \cdot 3 \cdot \lg 3$.
Suponha agora que $n > 3$. Então...

Continuação da prova

$$
\begin{aligned}
T(n) &= T(\lceil \frac{n}{3} \rceil) + T(\lfloor \frac{2n}{3} \rfloor) + n \\
&\overset{\text{hi}}{\leq} 20\lceil \frac{n}{3} \rceil \lg \lceil \frac{n}{3} \rceil + 20\lfloor \frac{2n}{3} \rfloor \lg \lfloor \frac{2n}{3} \rfloor + n \\
&\leq 20\frac{n+2}{3} \lceil \lg \frac{n}{3} \rceil + 20\frac{2n}{3} \lg \frac{2n}{3} + n \\
&< 20\frac{n+2}{3} (\lg \frac{n}{3} + 1) + 20\frac{2n}{3} \lg \frac{2n}{3} + n \\
&= 20\frac{n+2}{3} \lg \frac{2n}{3} + 20\frac{2n}{3} \lg \frac{2n}{3} + n \\
&= 20\frac{n}{3} \lg \frac{2n}{3} + 20\frac{2}{3} \lg \frac{2n}{3} + 20\frac{2n}{3} \lg \frac{2n}{3} + n
\end{aligned}
$$

Continuação da continuação da prova

$$
\begin{aligned}
&< 20n \lg \frac{2n}{3} + 14 \lg \frac{2n}{3} + n \\
&= 20n \lg n + 20n \lg \frac{2}{3} + 14 \lg n + 14 \lg \frac{2}{3} + n \\
&< 20n \lg n + 20n(-0.58) + 14 \lg n + 14(-0.58) + n \\
&< 20n \lg n - 11n + 14 \lg n - 8 + n \\
&= 20n \lg n - 10n + 14 \lg n - 8 \\
&< 20n \lg n - 10n + 7n - 8 \\
&< 20n \lg n
\end{aligned}
$$

Iiiéééééssss!

De volta à intuição

Certifique-se que a conclusão seria a mesma qualquer que fosse a proporção fixa que tomássemos. Por exemplo, resolva o seguinte...

De volta à intuição

Certifique-se que a conclusão seria a mesma qualquer que fosse a proporção fixa que tomássemos. Por exemplo, resolva o seguinte...

Exercício: Considere a recorrência

$$T(1) = 1$$

$$T(n) = T(\lceil n/10 \rceil) + T(\lfloor 9n/10 \rfloor) + n$$

para $n = 2, 3, 4, \ldots$ e mostre que $T(n)$ é $\mathrm{O}(n \lg n)$.

De volta à intuição

Certifique-se que a conclusão seria a mesma qualquer que fosse a proporção fixa que tomássemos. Por exemplo, resolva o seguinte...

Exercício: Considere a recorrência

$$T(1) = 1$$

$$T(n) = T(\lceil n/10 \rceil) + T(\lfloor 9n/10 \rfloor) + n$$

para $n = 2, 3, 4, \ldots$ e mostre que $T(n)$ é $\mathrm{O}(n \lg n)$.

Note que, se o QUICKSORT fizer uma “boa” partição a cada, digamos, 5 níveis da recursão, o efeito geral é o mesmo, assintoticamente, que ter feito uma boa partição em todos os níveis.

De volta ao caso médio

Considere que $A[1\,..\,n]$ é uma permutação escolhida uniformemente dentre todas as permutações de 1 a n.

De volta ao caso médio

Considere que $A[1\,..\,n]$ é uma permutação escolhida uniformemente dentre todas as permutações de 1 a n.

Seja $X(A)$ o número de vezes que a linha 4 do PARTICIONE é executada para uma chamada de QUICKSORT$(A, 1, n)$.

Observe que X é uma variável aleatória.

De volta ao caso médio

Considere que $A[1\,..\,n]$ é uma permutação escolhida uniformemente dentre todas as permutações de 1 a n.

Seja $X(A)$ o número de vezes que a linha 4 do PARTICIONE é executada para uma chamada de QUICKSORT$(A, 1, n)$.

Observe que X é uma variável aleatória.

Uma maneira de estimarmos o consumo de tempo médio do QUICKSORT é calcularmos $\mathrm{E}[X]$.

De volta ao caso médio

Considere que $A[1\,..\,n]$ é uma permutação escolhida uniformemente dentre todas as permutações de 1 a n.

Seja $X(A)$ o número de vezes que a linha 4 do PARTICIONE é executada para uma chamada de QUICKSORT$(A, 1, n)$.

Observe que X é uma variável aleatória.

Uma maneira de estimarmos o consumo de tempo médio do QUICKSORT é calcularmos $\mathrm{E}[X]$.

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

Quem serão essas variáveis aleatórias binárias?

Exemplo

1	3	6	2	5	7	4

1	3	2	4	5	7	6

1	2	3	4	5	6	7

	1	2	3	4	5	6	7
1		1	0	1	0	0	0
2	1		1	1	0	0	0
3	0	1		1	0	0	0
4	1	1	1		1	1	1
5	0	0	0	1		1	0
6	0	0	0	1	1		1
7	0	0	0	1	0	1	

De volta ao caso médio

Continuamos na aula que vem...

Quicksort aleatorizado

CLRS 7.4

Quicksort aleatorizado

PARTICIONE-ALEA(A, p, d)

1 $\quad i \leftarrow$ RANDOM (p, d)

2 $\quad A[i] \leftrightarrow A[d]$

3 $\quad$ **devolva** PARTICIONE (A, p, d)

QUICKSORT-ALE (A, p, d)

1 $\quad$ **se** $p < d$

2 $\qquad$ **então** $q \leftarrow$ PARTICIONE-ALEA (A, p, d)

3 $\qquad\qquad$ QUICKSORT-ALE $(A, p, q-1)$

4 $\qquad\qquad$ 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 $p \le q \le d$ e
$A[p\,..\,q{-}1] \le A[q] < A[q{+}1\,..\,d]$

```
PARTICIONE (A, p, d)
    x ← A[d]            ▷ x é o "pivô"
    i ← p−1
    para j ← p até d − 1 faça
        se A[j] ≤ x
            então i ← i + 1
                  A[i] ↔ A[j]
    A[i+1] ↔ 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?

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

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

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

Consumo de tempo esperado

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

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

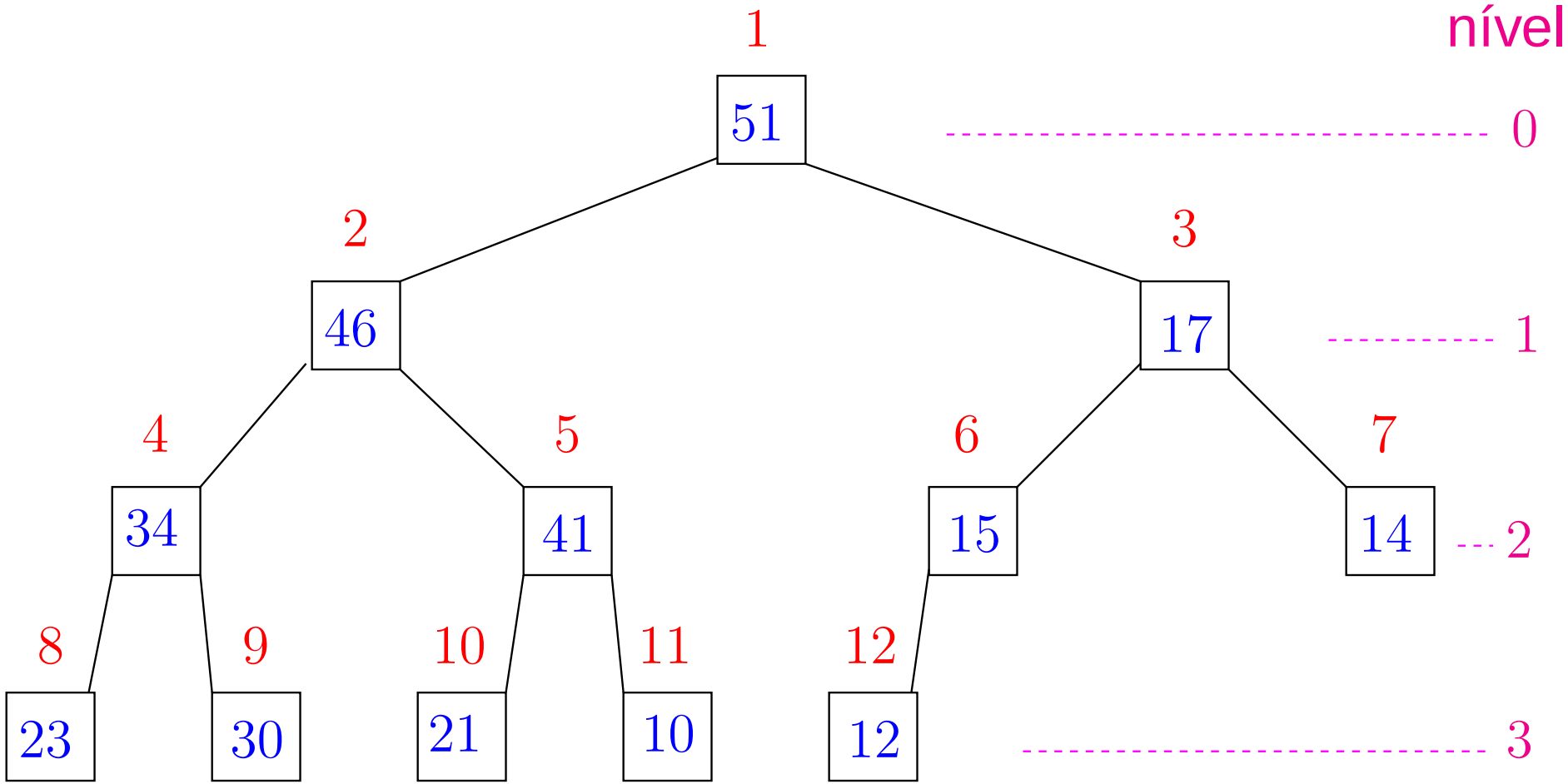
De uma forma mais geral, $A[j\,..\,m]$ é um heap se

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

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



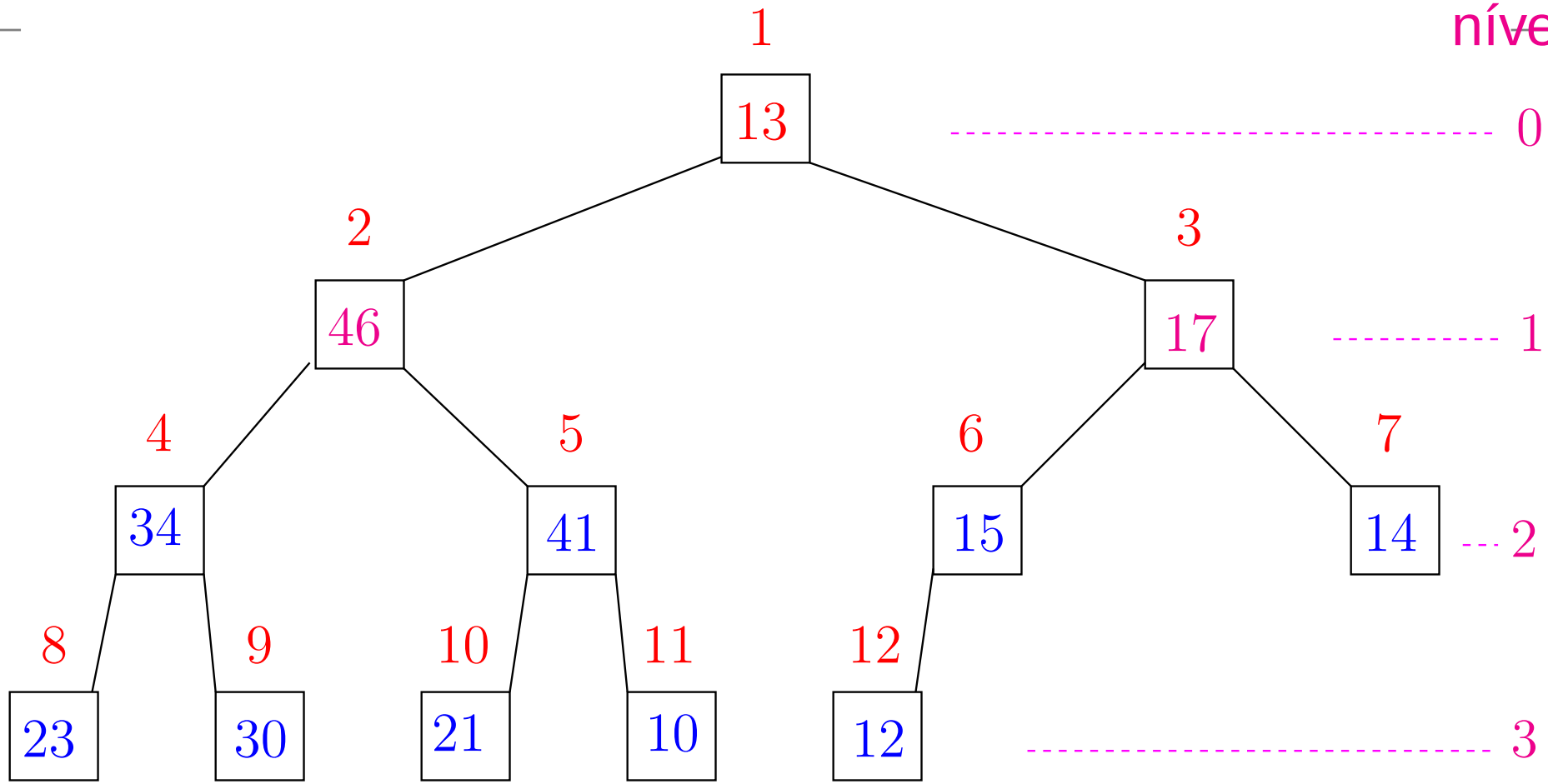
1	2	3	4	5	6	7	8	9	10	11	12
51	46	17	34	41	15	14	23	30	21	10	12

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.

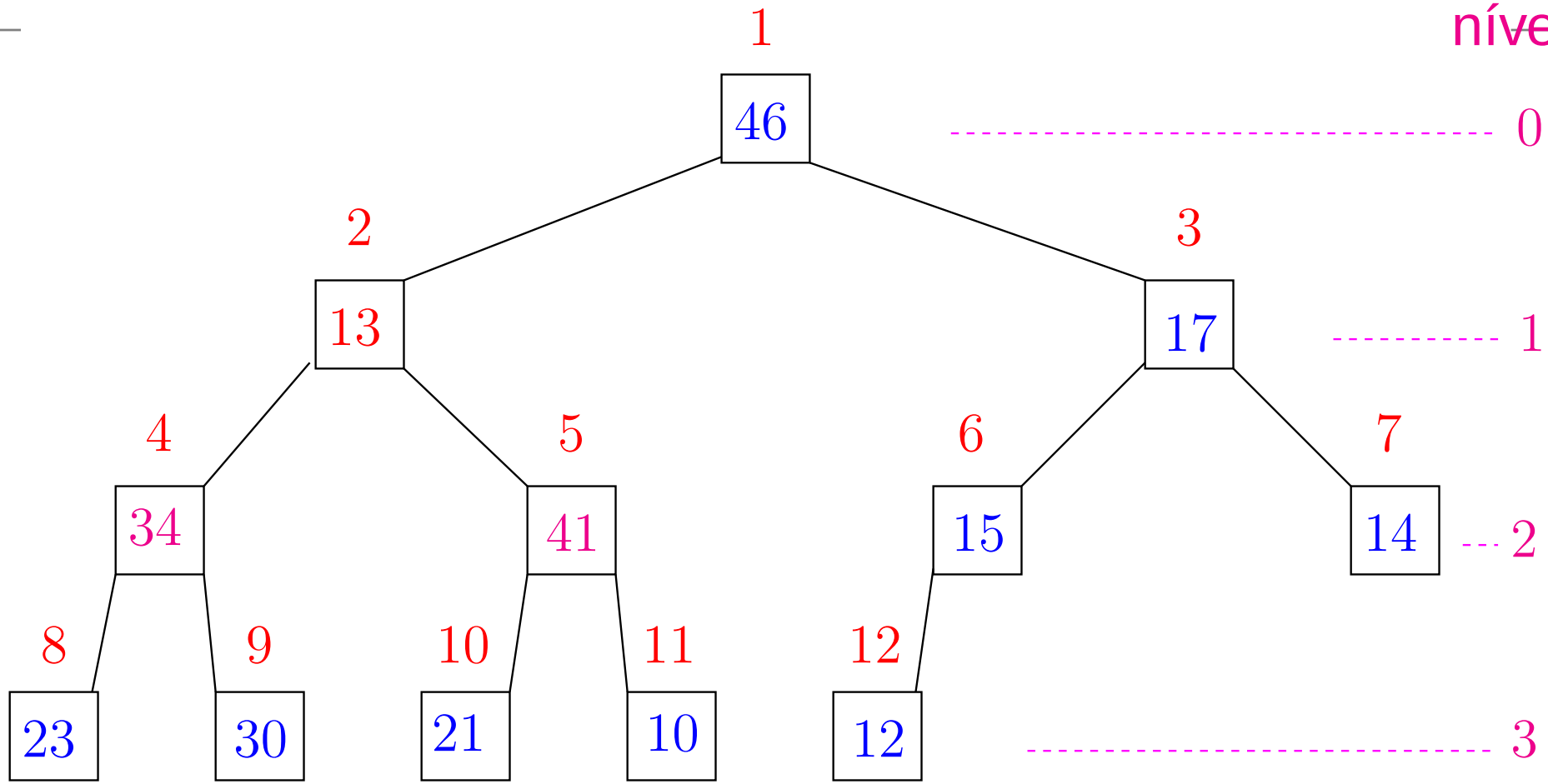
```
DESCE-HEAP (A, m, i)
    e ← 2i
    d ← 2i + 1
    se e ≤ m e A[e] > A[i]
        então maior ← e
        senão maior ← i
    se d ≤ m e A[d] > A[maior]
        então maior ← d
    se maior ≠ i
        então A[i] ↔ A[maior]
            DESCE-HEAP (A, m, maior)
```

Simulação



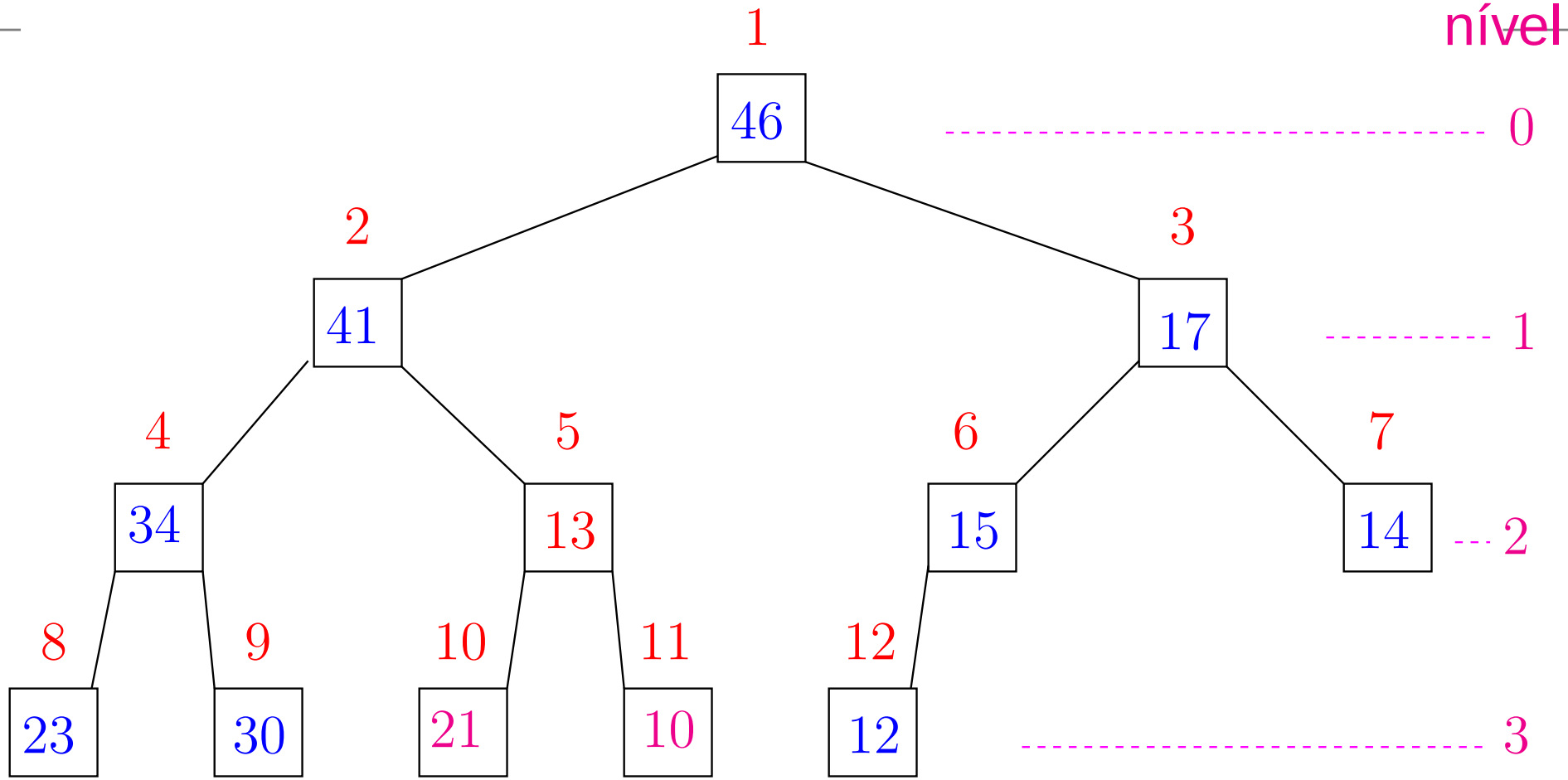
1	2	3	4	5	6	7	8	9	10	11	12
13	46	17	34	41	15	14	23	30	21	10	12

Simulação



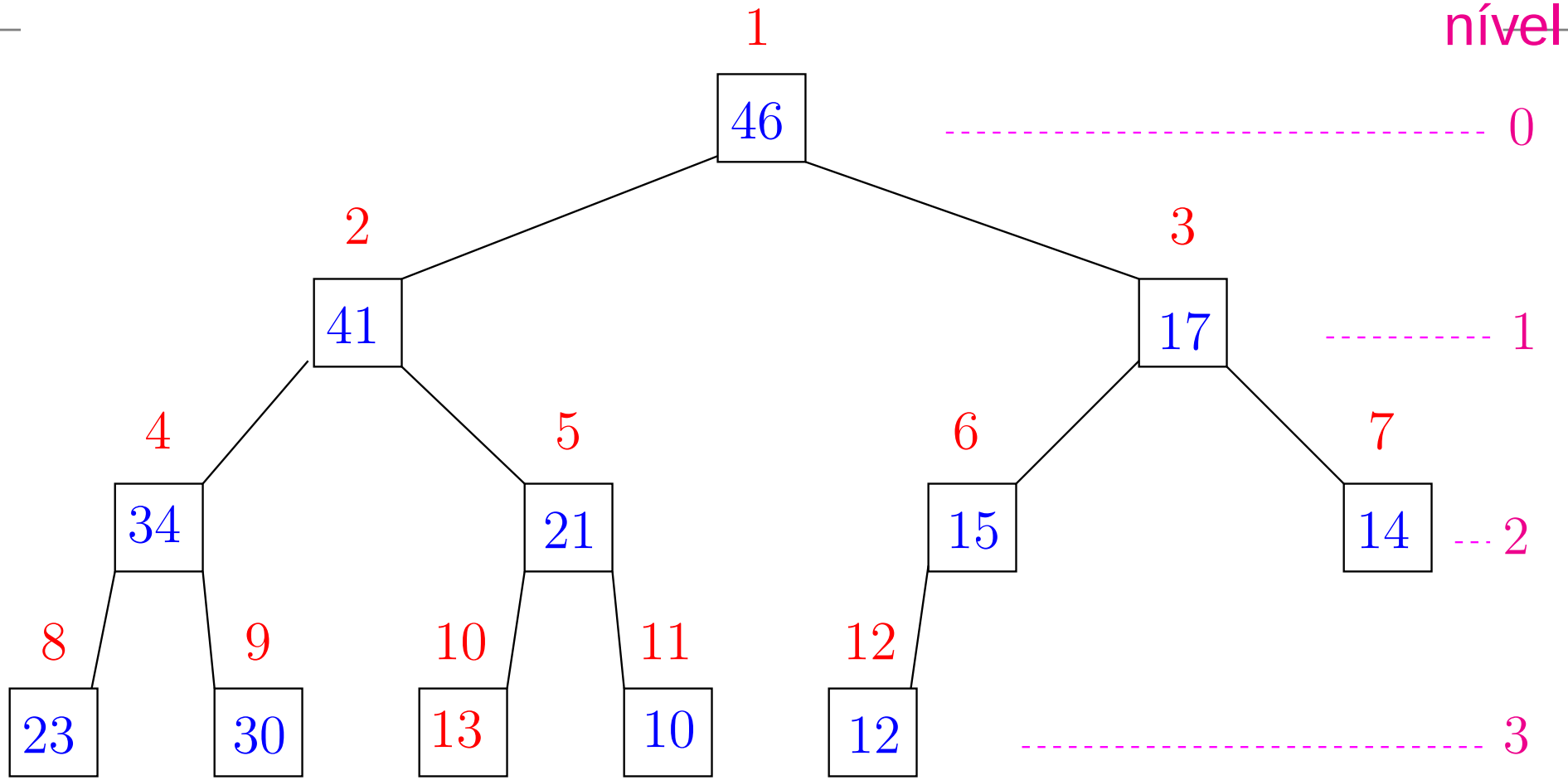
1	2	3	4	5	6	7	8	9	10	11	12
46	13	17	34	41	15	14	23	30	21	10	12

Simulação



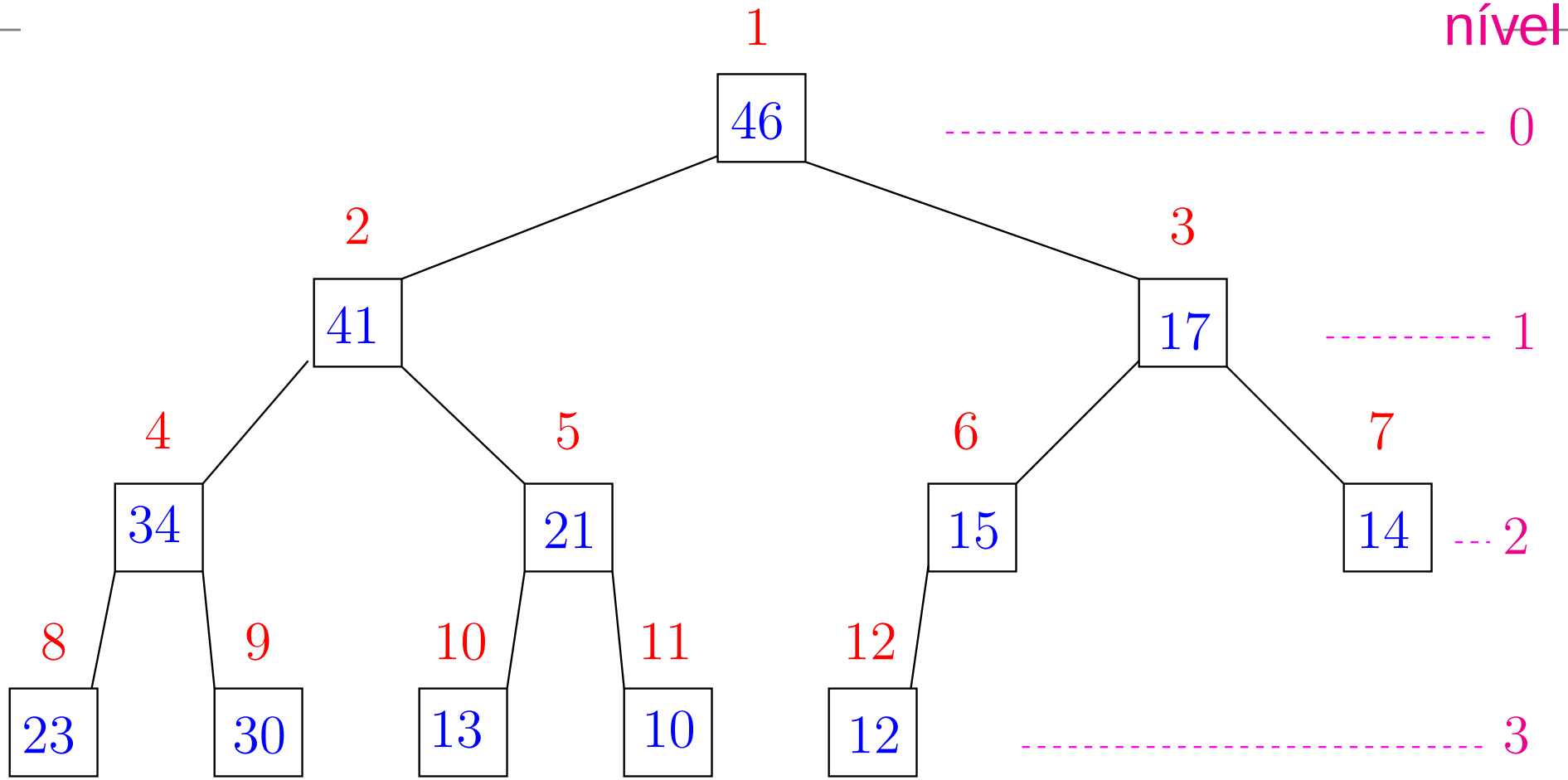
1	2	3	4	5	6	7	8	9	10	11	12
46	41	17	34	13	15	14	23	30	21	10	12

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:

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

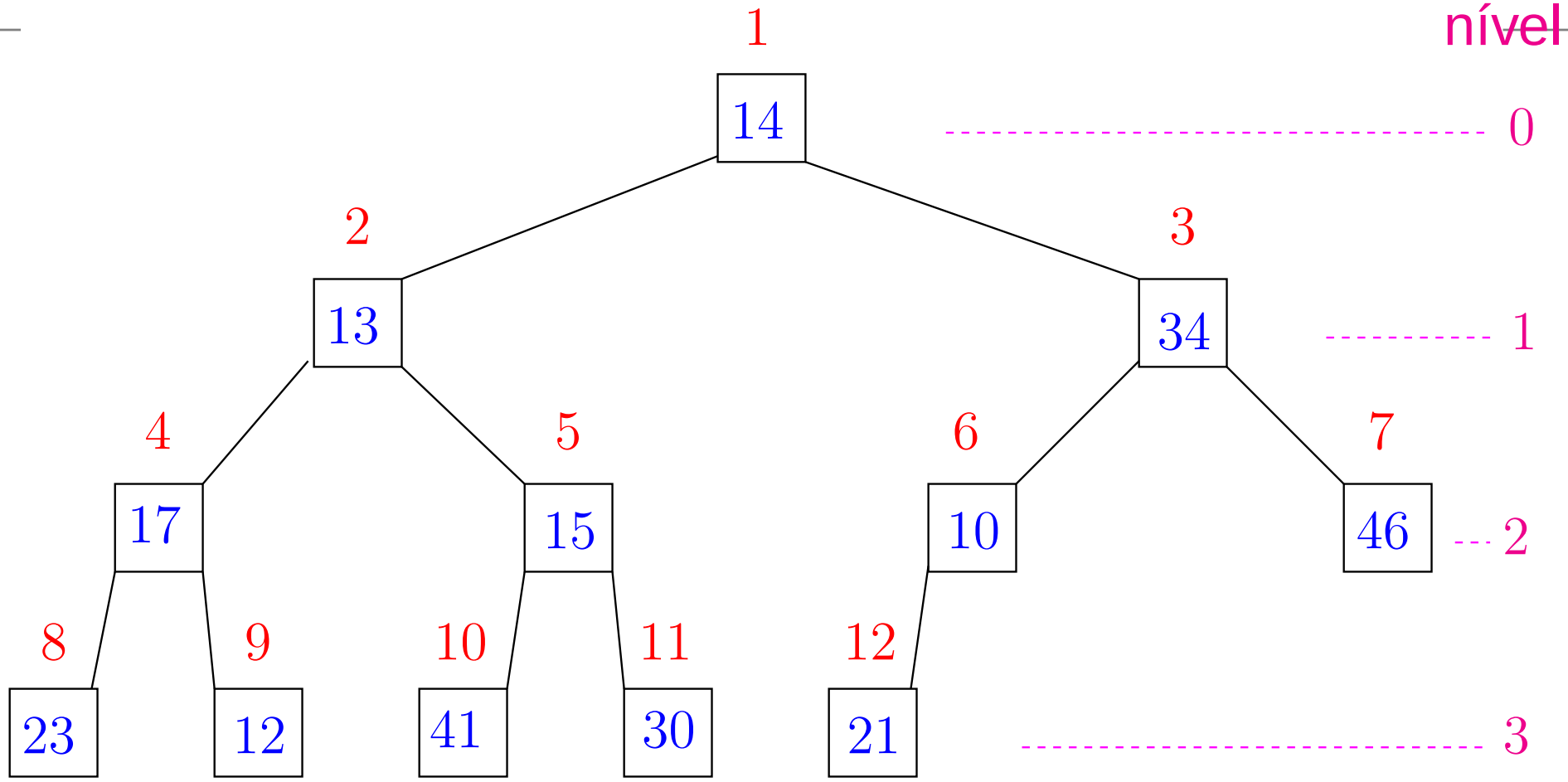
pois altura de $maior$ é $h-1$.

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

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

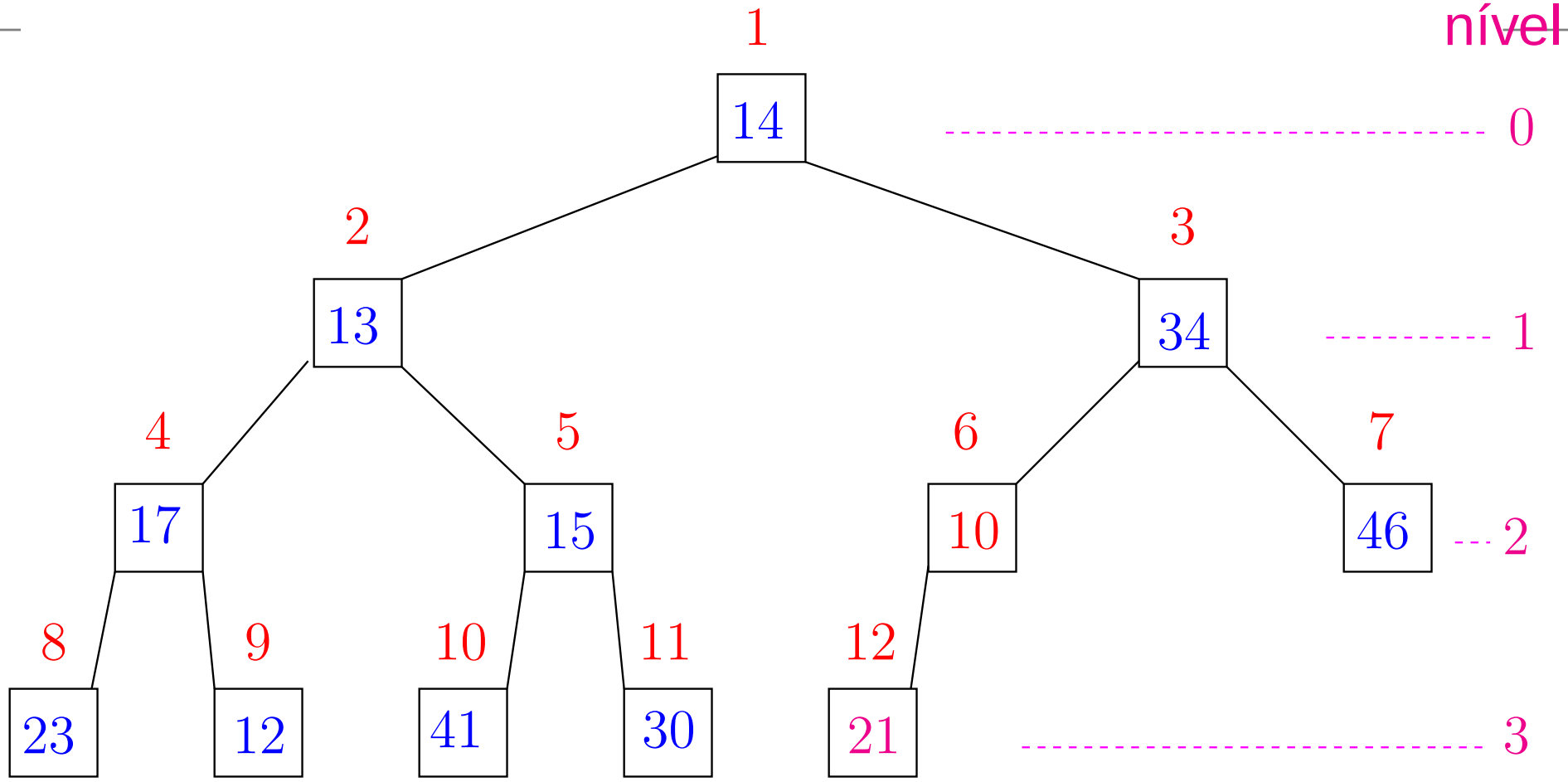
O consumo de tempo do algoritmo DESCE-HEAP é $\mathrm{O}(\lg m)$ (ou melhor ainda, $\mathrm{O}(h)$).

Construção de um heap



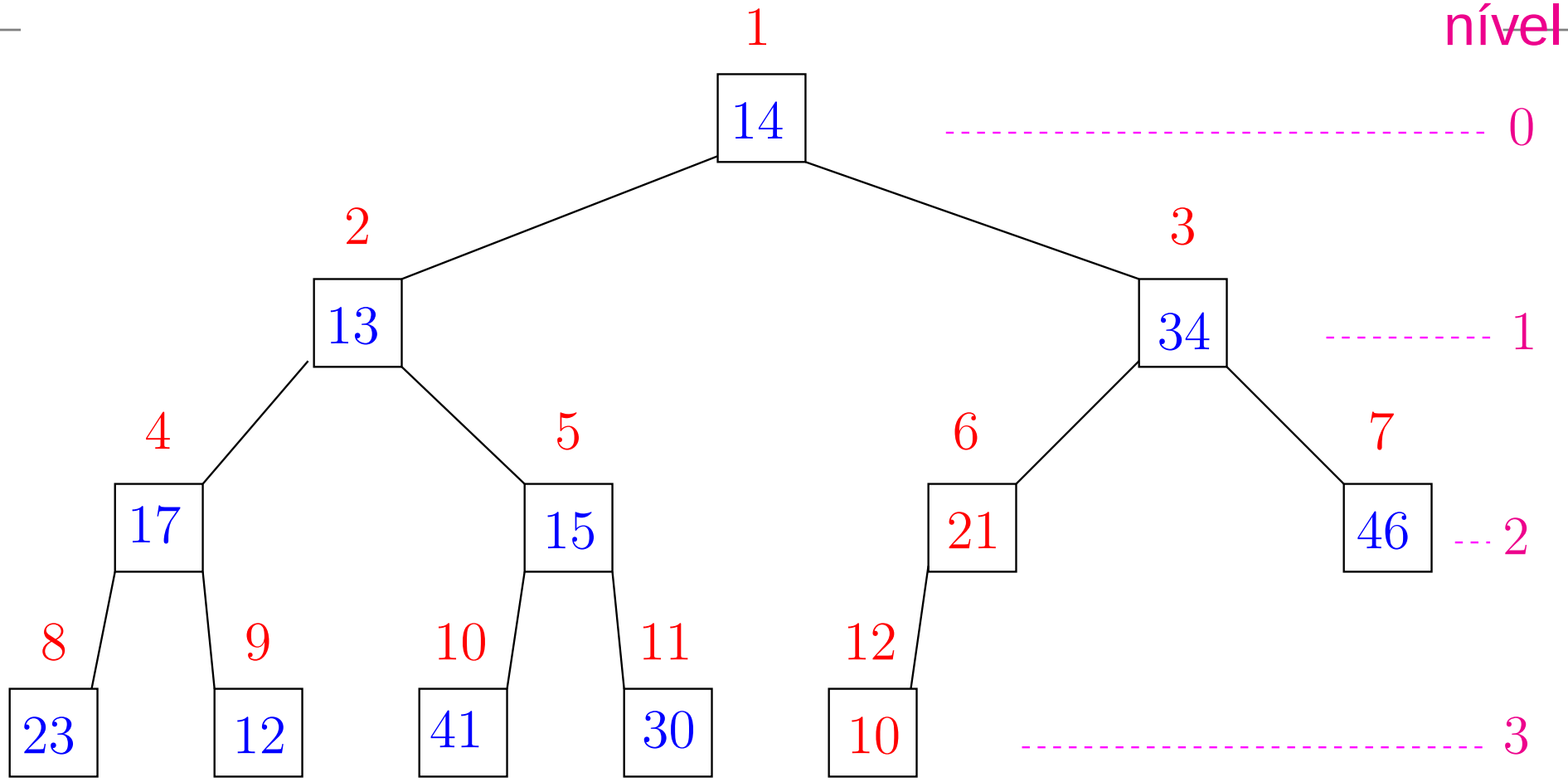
1	2	3	4	5	6	7	8	9	10	11	12
14	13	34	17	15	10	46	23	12	41	30	21

Construção de um heap



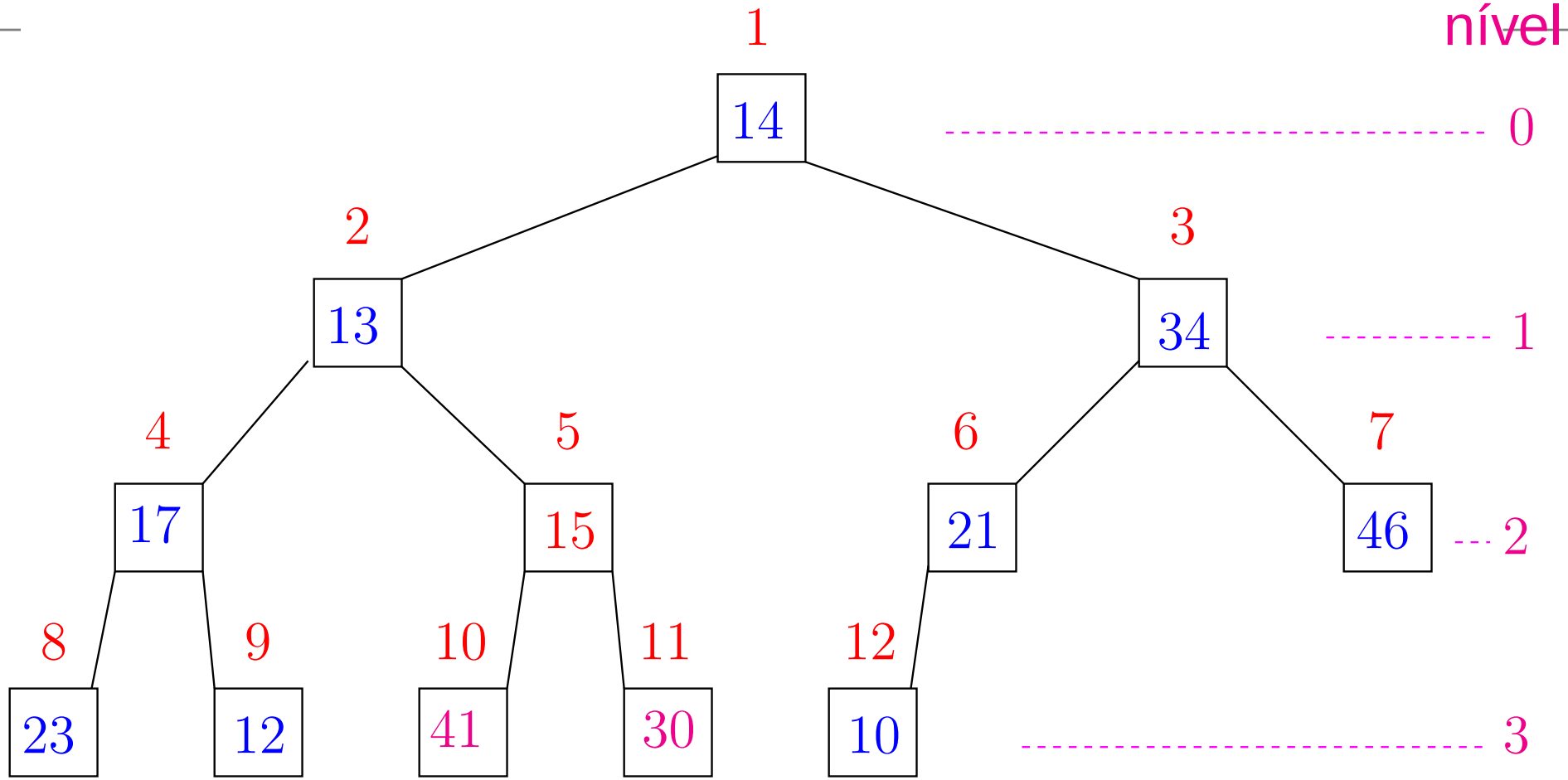
1	2	3	4	5	6	7	8	9	10	11	12
14	13	34	17	15	10	46	23	12	41	30	21

Construção de um heap



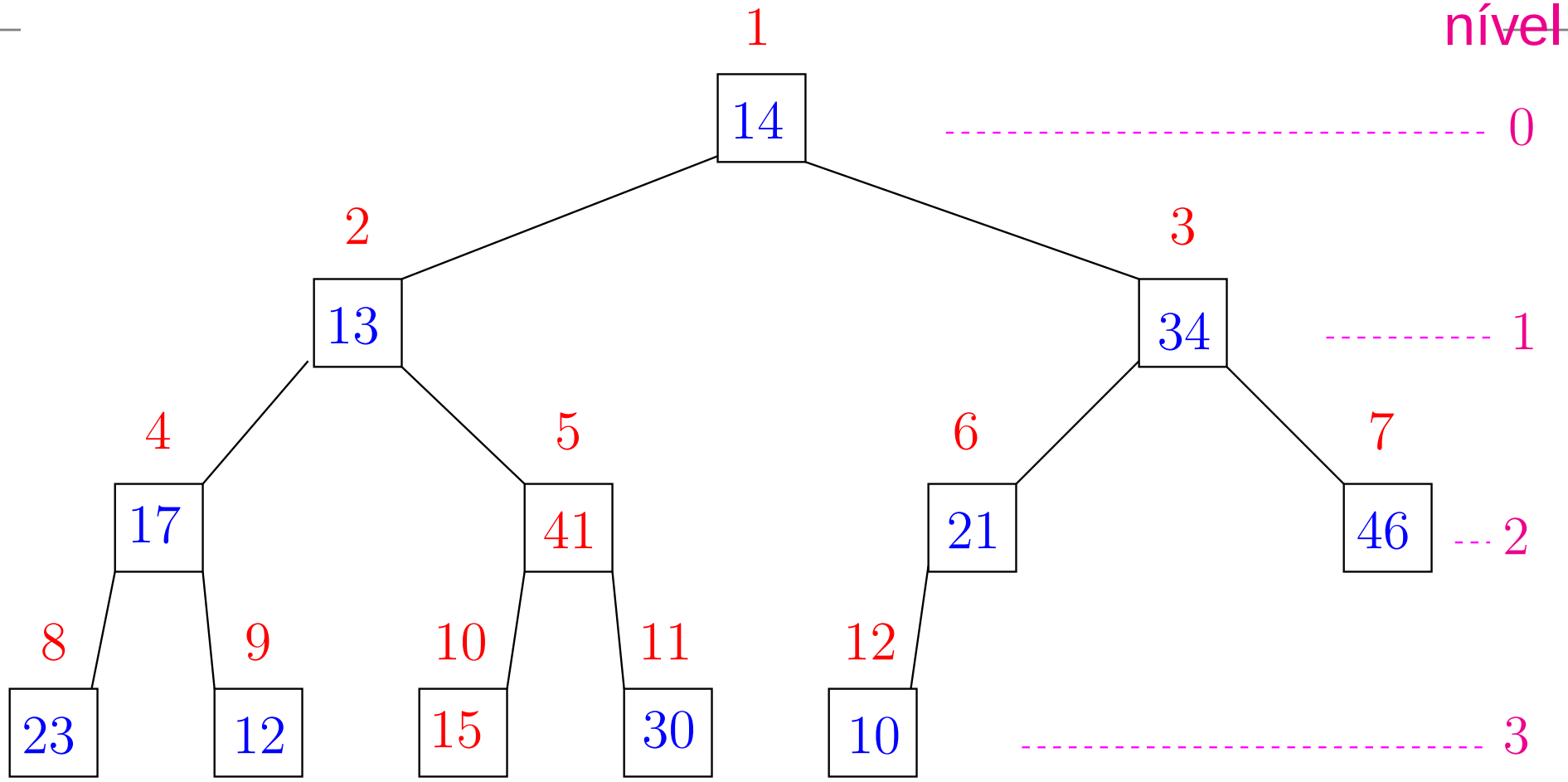
1	2	3	4	5	6	7	8	9	10	11	12
14	13	34	17	15	21	46	23	12	41	30	10

Construção de um heap



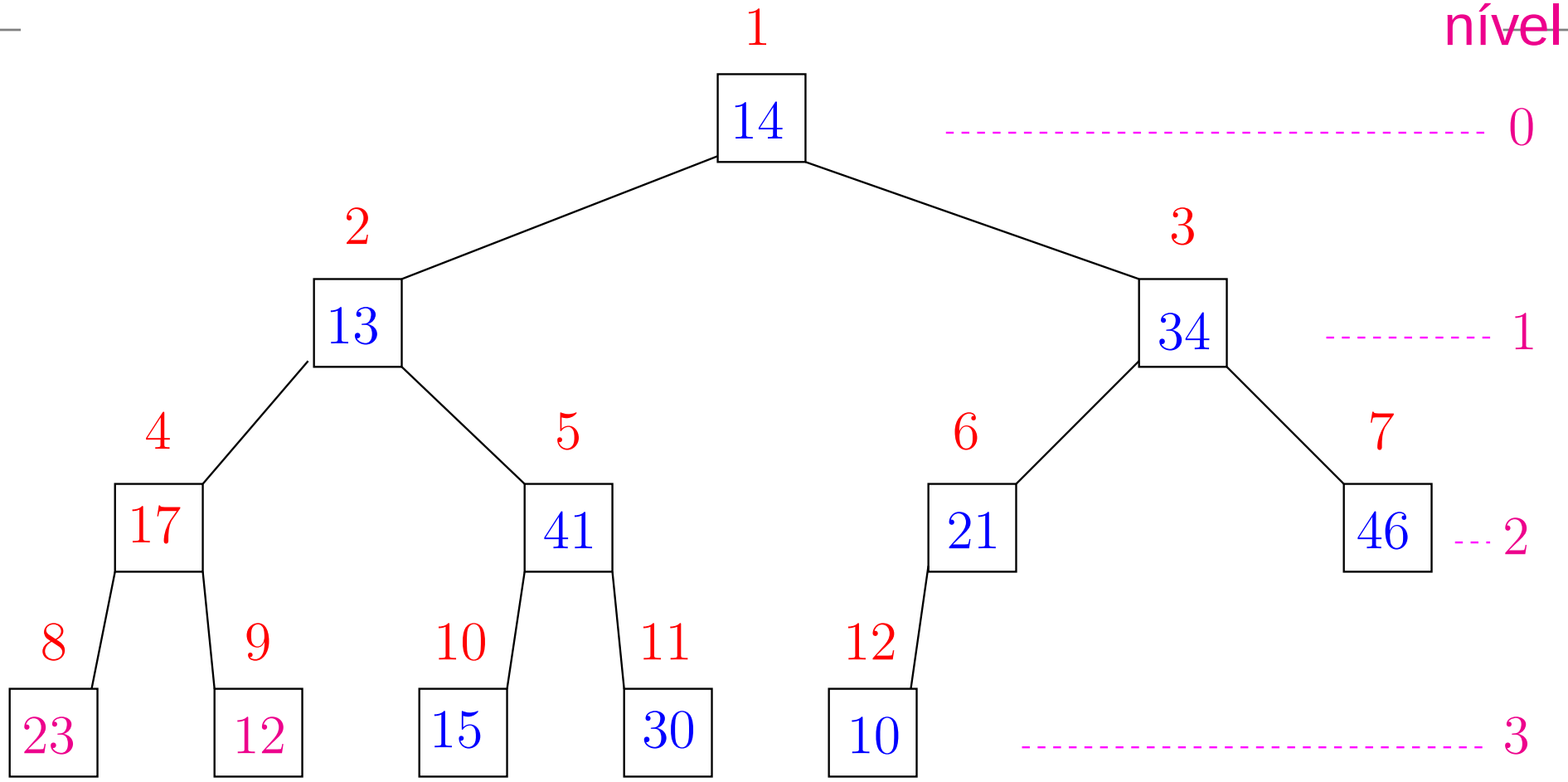
1	2	3	4	5	6	7	8	9	10	11	12
14	13	34	17	15	21	46	23	12	41	30	10

Construção de um heap



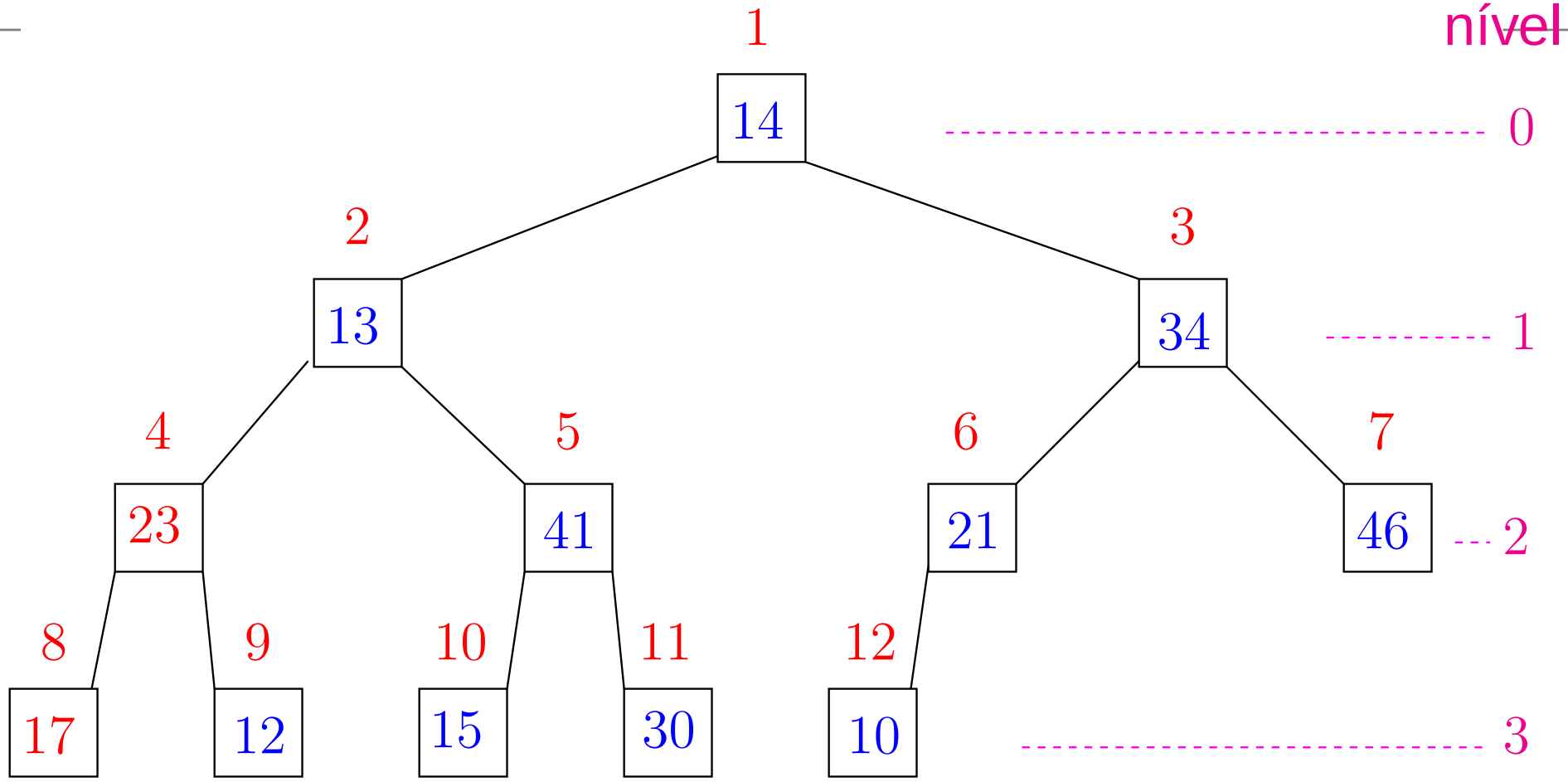
1	2	3	4	5	6	7	8	9	10	11	12
14	13	34	17	41	21	46	23	12	15	30	10

Construção de um heap



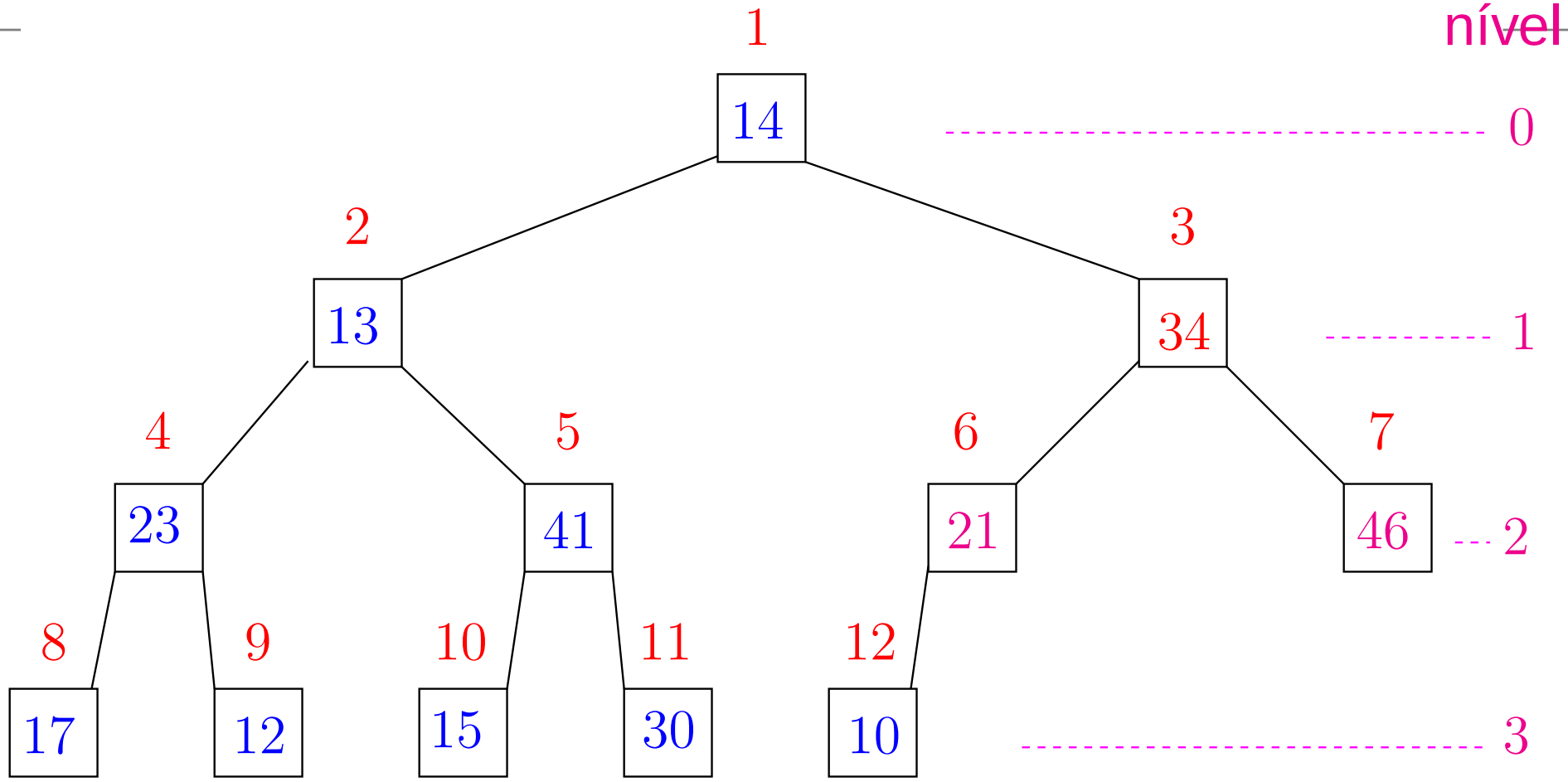
1	2	3	4	5	6	7	8	9	10	11	12
14	13	34	17	41	21	46	23	12	15	30	10

Construção de um heap



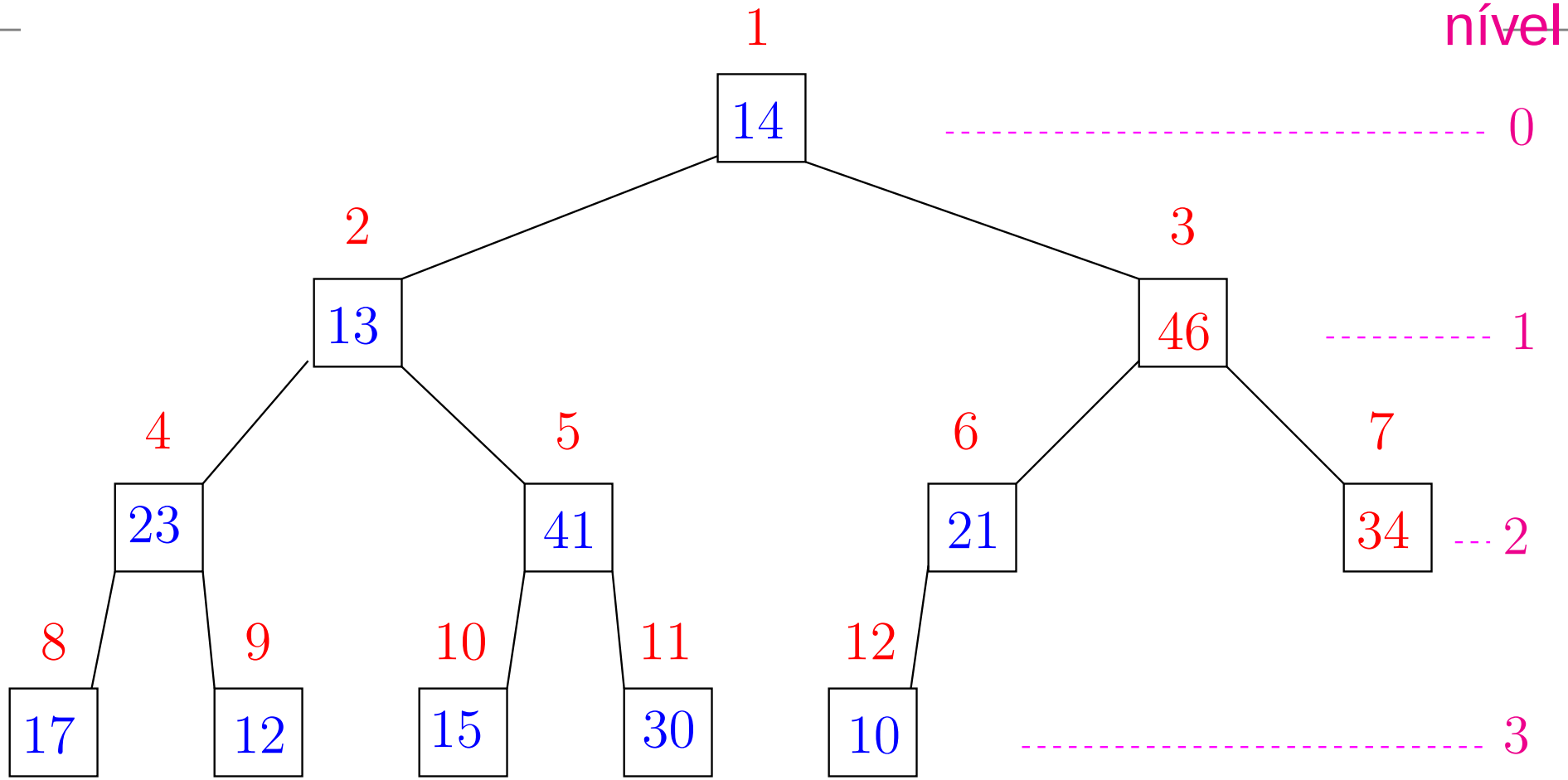
1	2	3	4	5	6	7	8	9	10	11	12
14	13	34	23	41	21	46	17	12	15	30	10

Construção de um heap



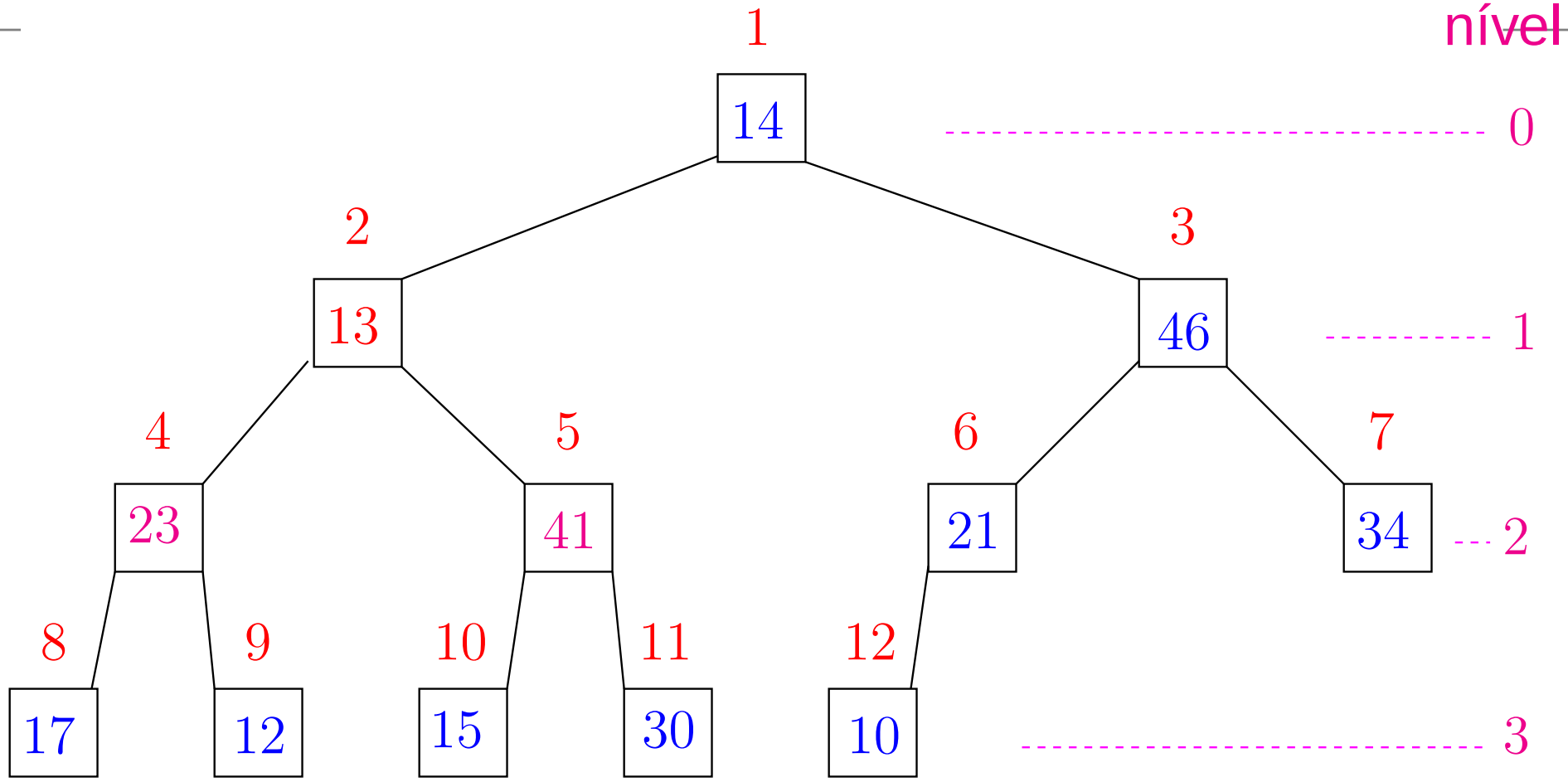
1	2	3	4	5	6	7	8	9	10	11	12
14	13	34	23	41	21	46	17	12	15	30	10

Construção de um heap



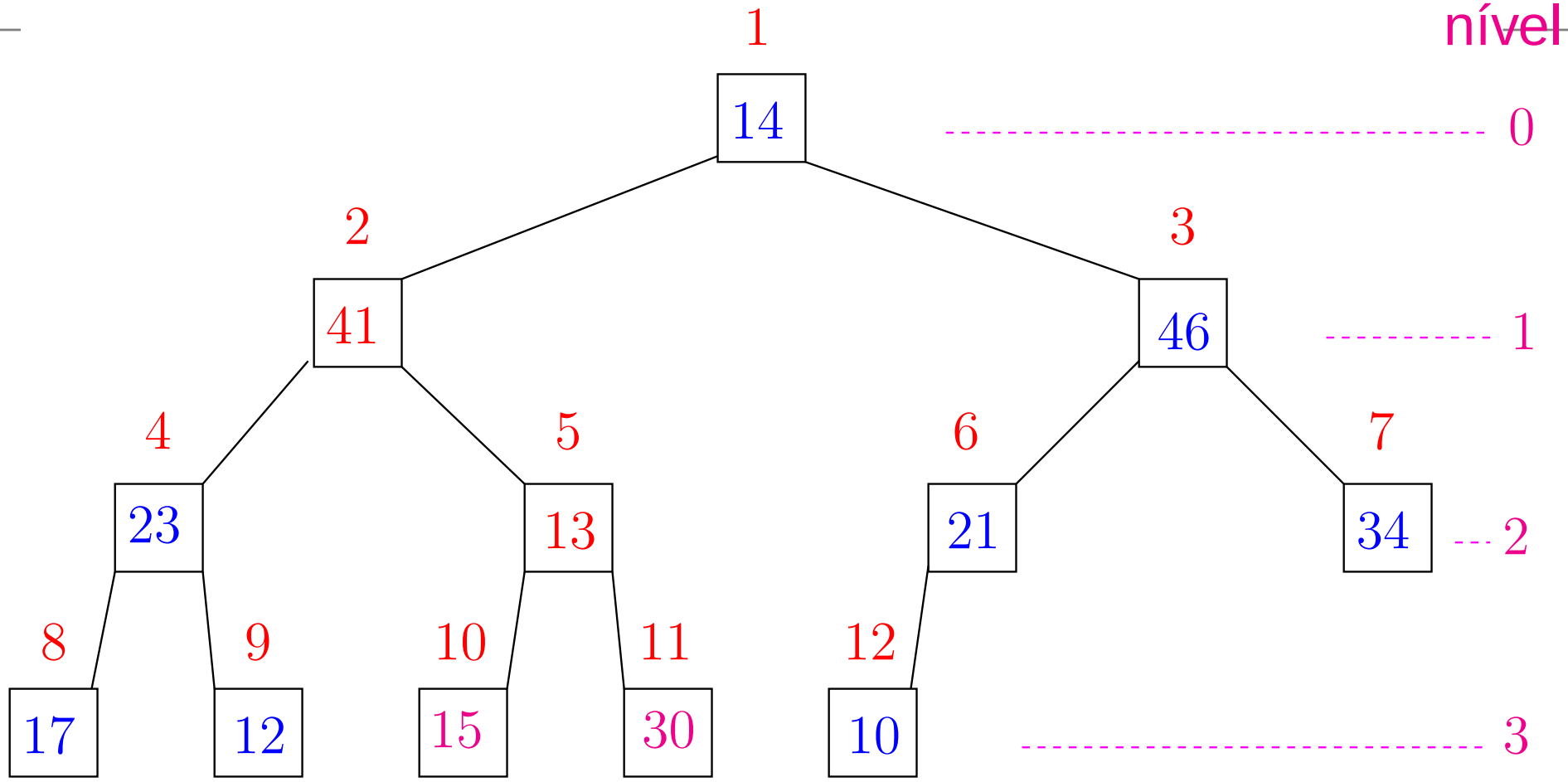
1	2	3	4	5	6	7	8	9	10	11	12
14	13	46	23	41	21	34	17	12	15	30	10

Construção de um heap



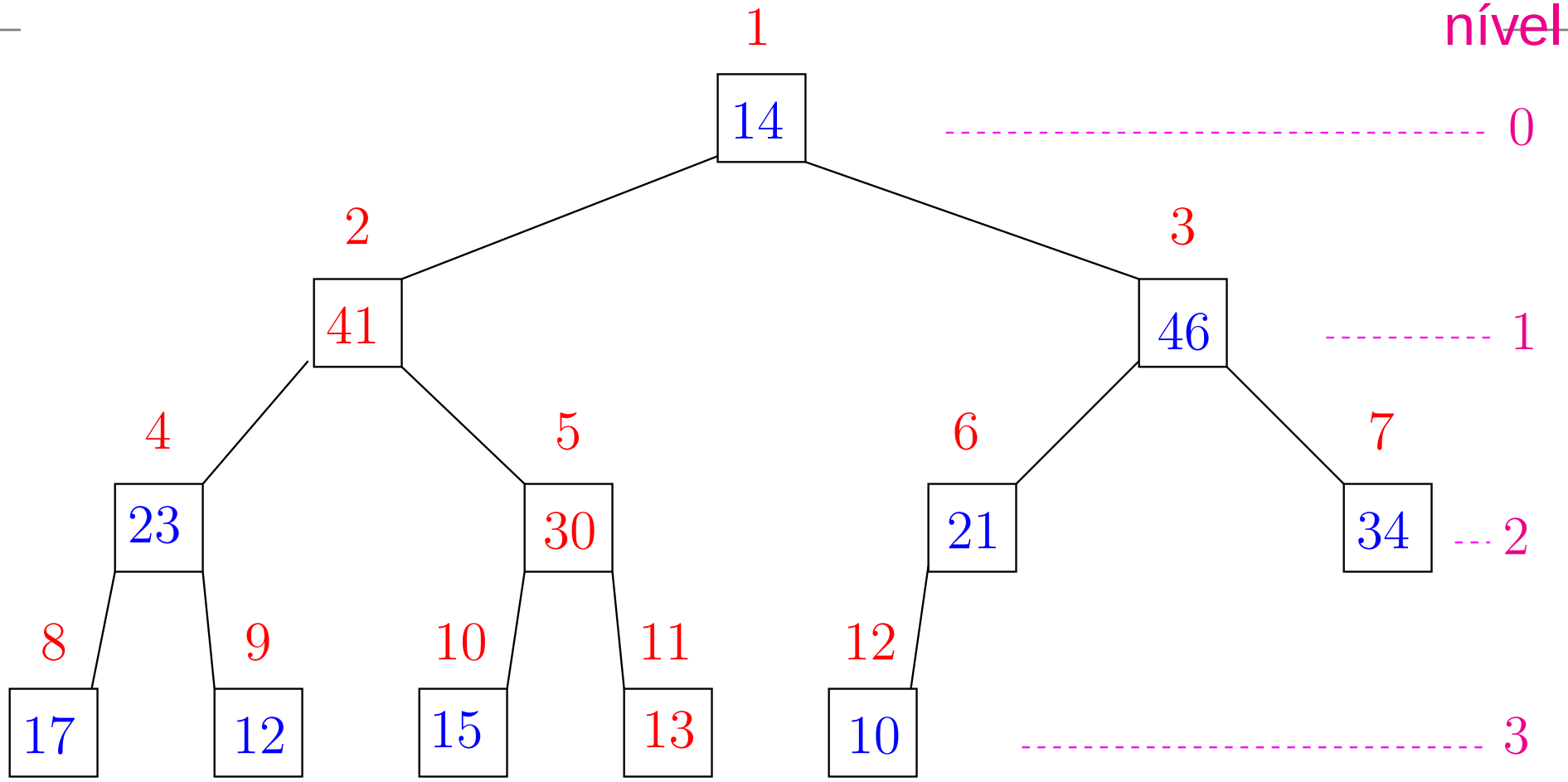
1	2	3	4	5	6	7	8	9	10	11	12
14	13	46	23	41	21	34	17	12	15	30	10

Construção de um heap



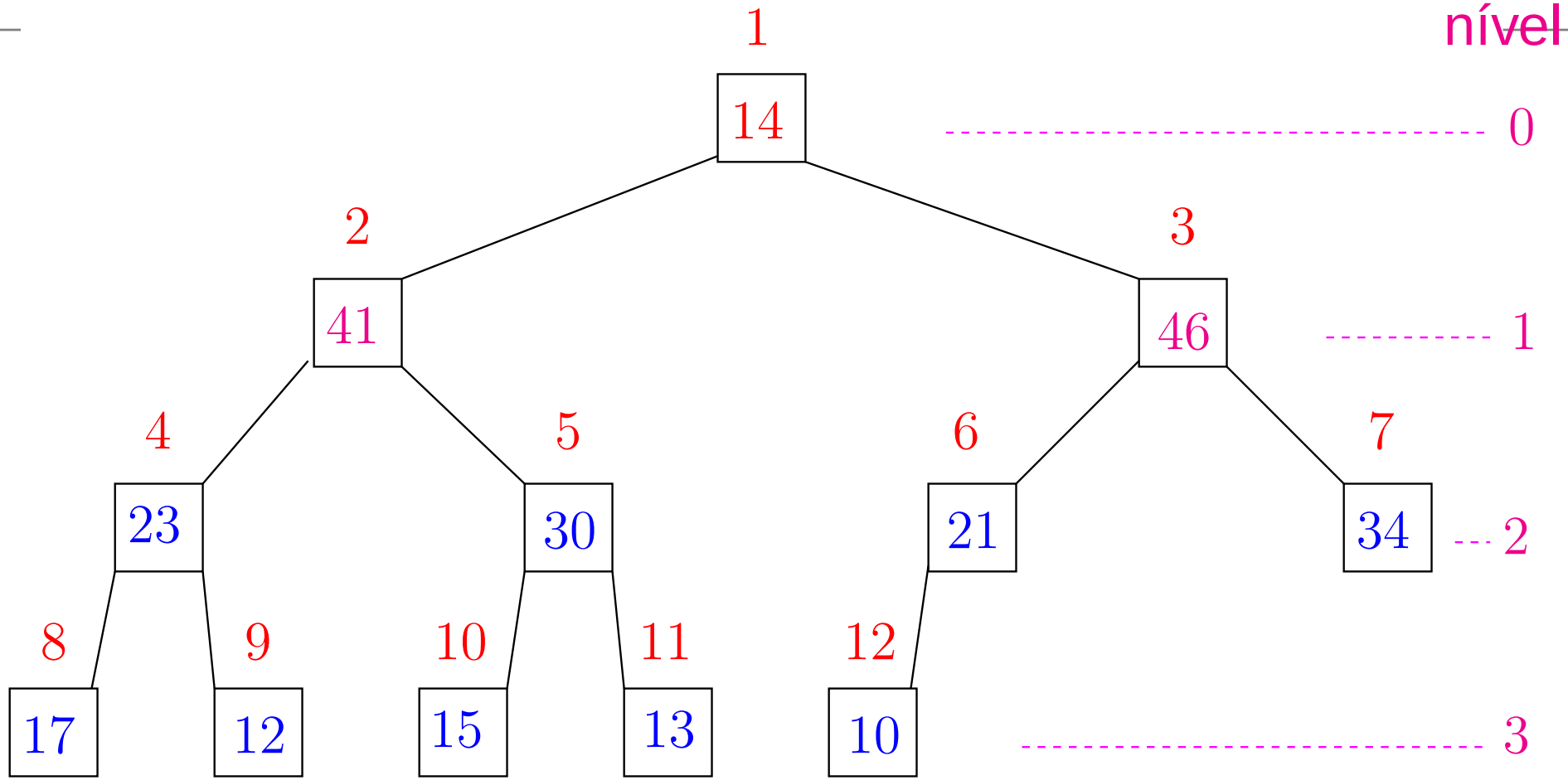
1	2	3	4	5	6	7	8	9	10	11	12
14	41	46	23	13	21	34	17	12	15	30	10

Construção de um heap



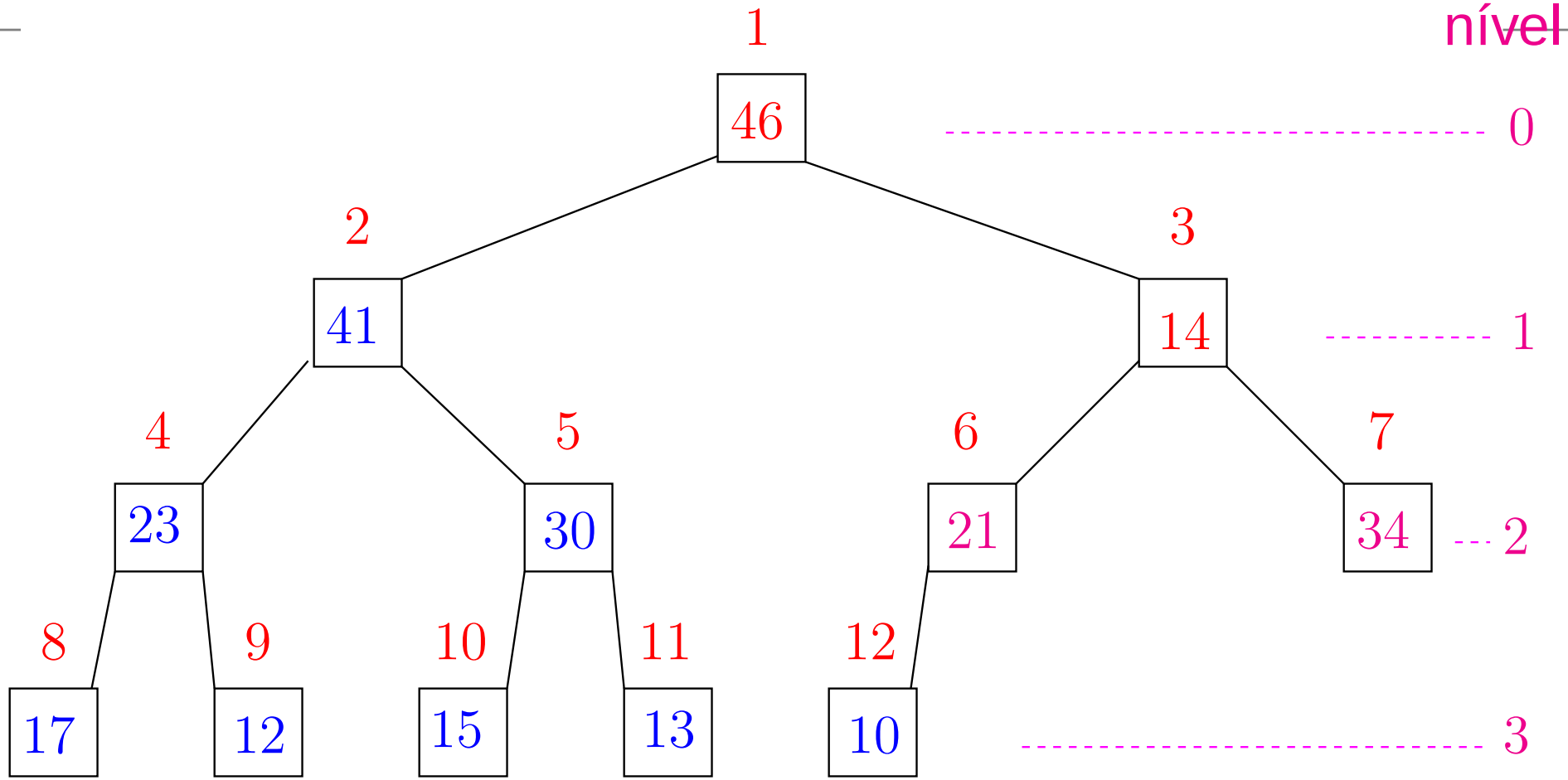
1	2	3	4	5	6	7	8	9	10	11	12
14	41	46	23	30	21	34	17	12	15	13	10

Construção de um heap



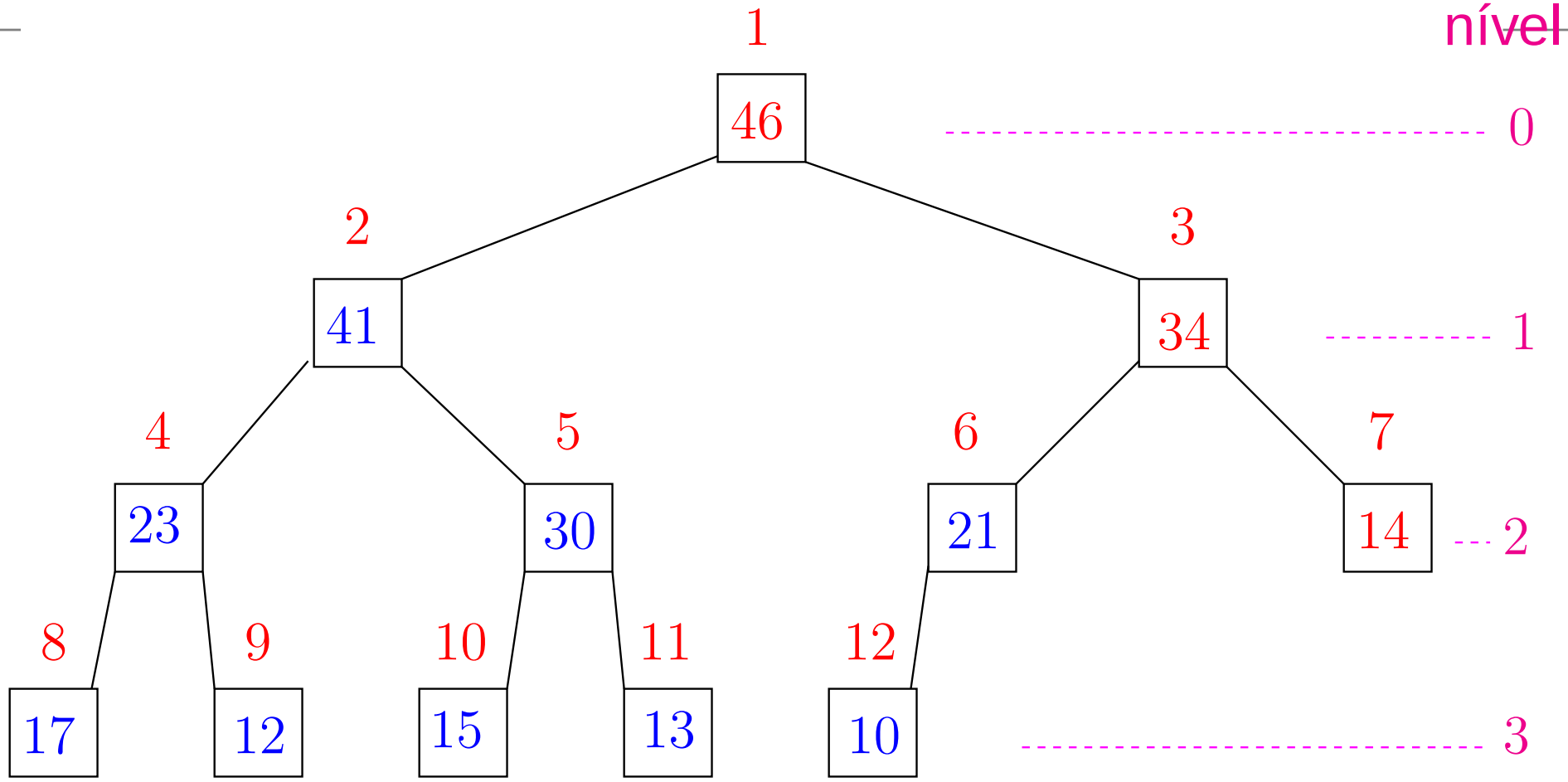
1	2	3	4	5	6	7	8	9	10	11	12
14	41	46	23	30	21	34	17	12	15	13	10

Construção de um heap



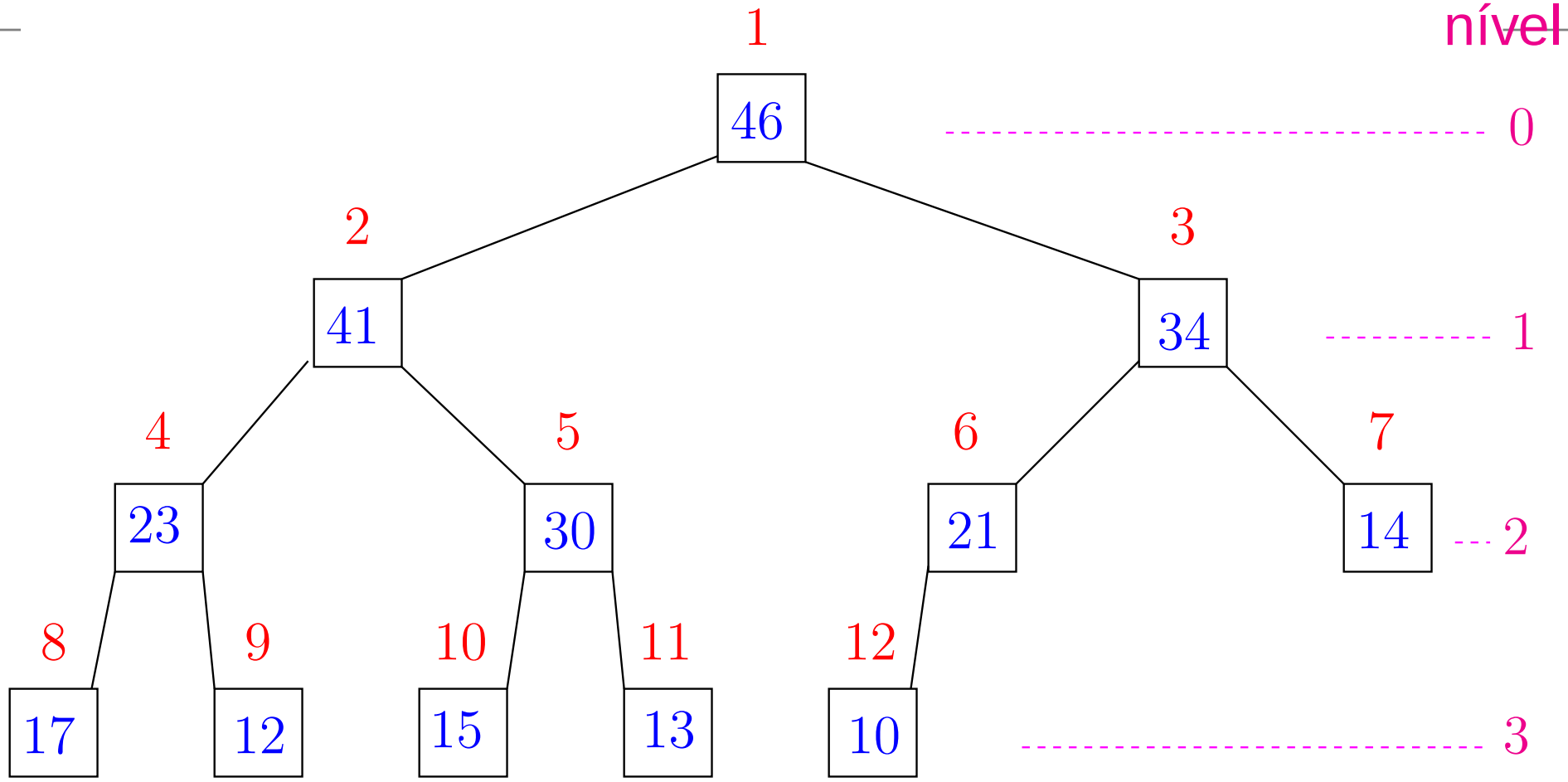
1	2	3	4	5	6	7	8	9	10	11	12
46	41	14	23	30	21	34	17	12	15	13	10

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 $A[1\,..\,n]$ 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,

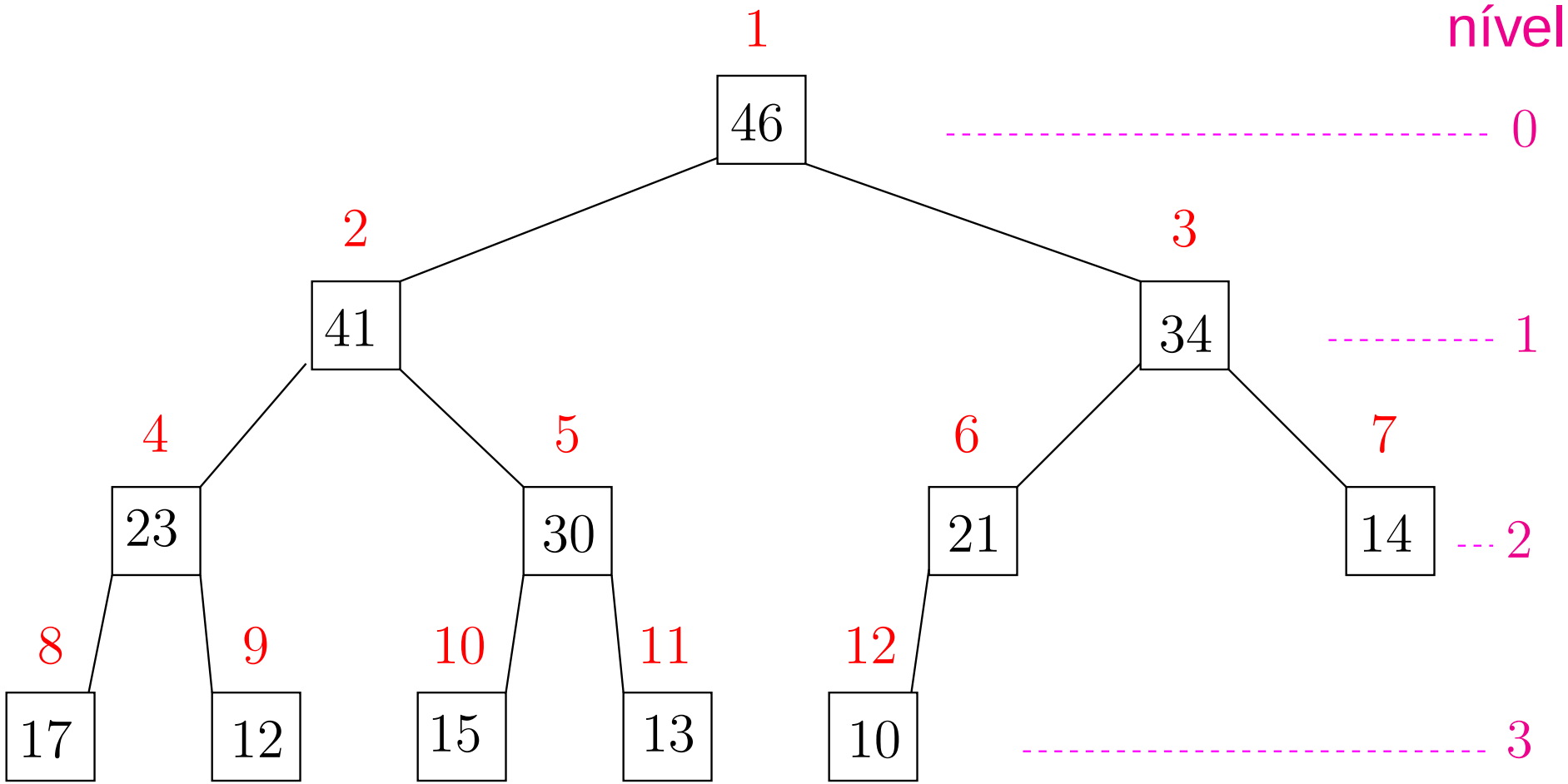
$$
\begin{aligned}
T(n) &= \sum_{h=1}^{\lfloor \lg n \rfloor} 2^{\lfloor \lg n \rfloor - h} h \\
&\leq \sum_{h=1}^{\lfloor \lg n \rfloor} \frac{n}{2^h} h \\
&\leq n \left(\frac{1}{2^1} + \frac{2}{2^2} + \frac{3}{2^3} + \cdots + \frac{\lfloor \lg n \rfloor}{2^{\lfloor \lg n \rfloor}} \right) \\
&< n \frac{1/2}{(1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

$T(n)$ 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,

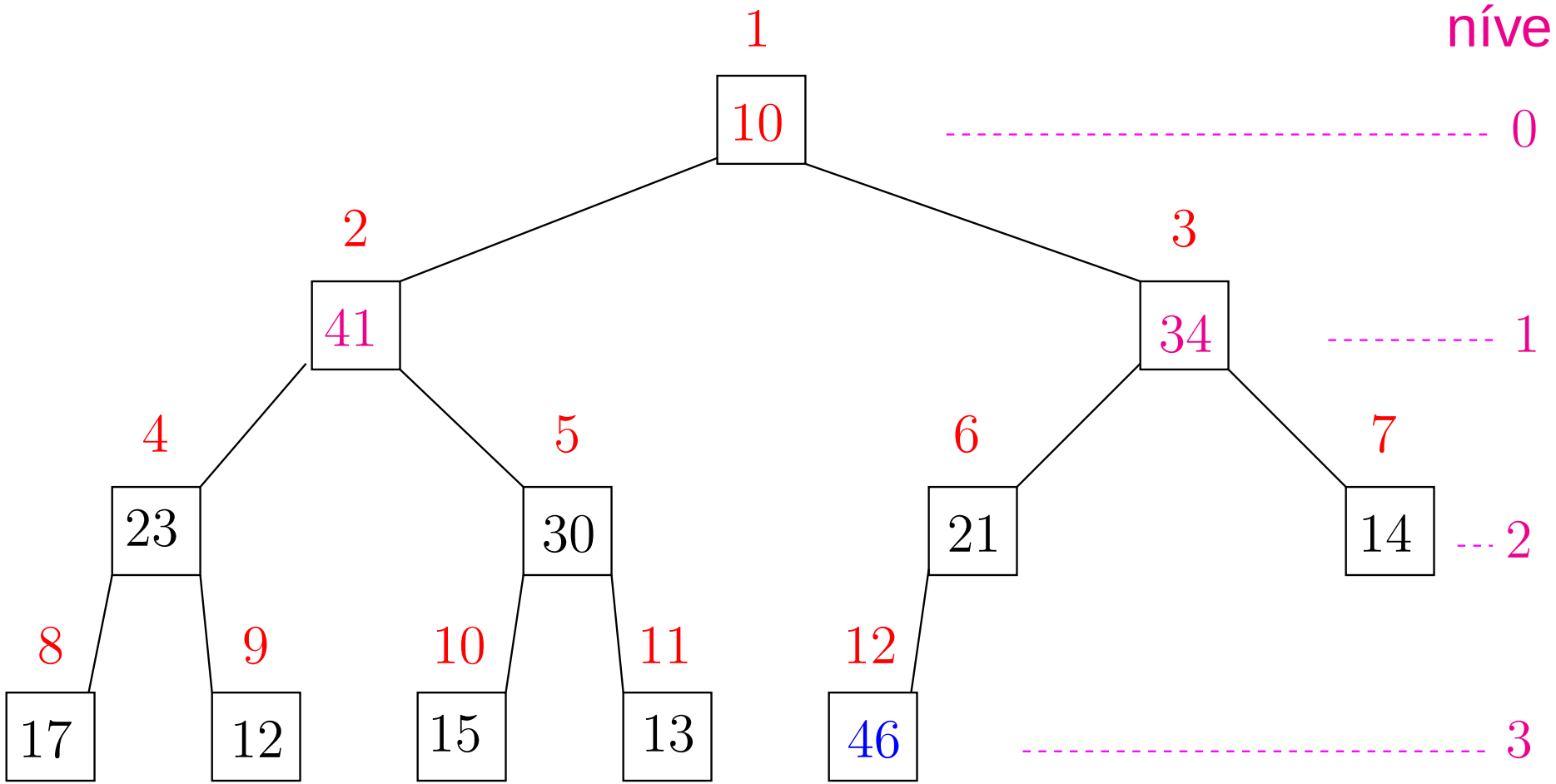
$$
\begin{aligned}
T(n) &= \sum_{h=0}^{\lfloor \lg n \rfloor} 2^{\lfloor \lg n \rfloor - h} \mathrm{O}(h) \\
&= \mathrm{O}\left(n \sum_{h=0}^{\lfloor \lg n \rfloor} \frac{h}{2^h} \right) \\
&= \mathrm{O}\left(n \sum_{h=0}^{\infty} \frac{h}{2^h} \right) \\
&= \mathrm{O}\left(n \frac{1/2}{(1-1/2)^2} \right)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

Heap sort



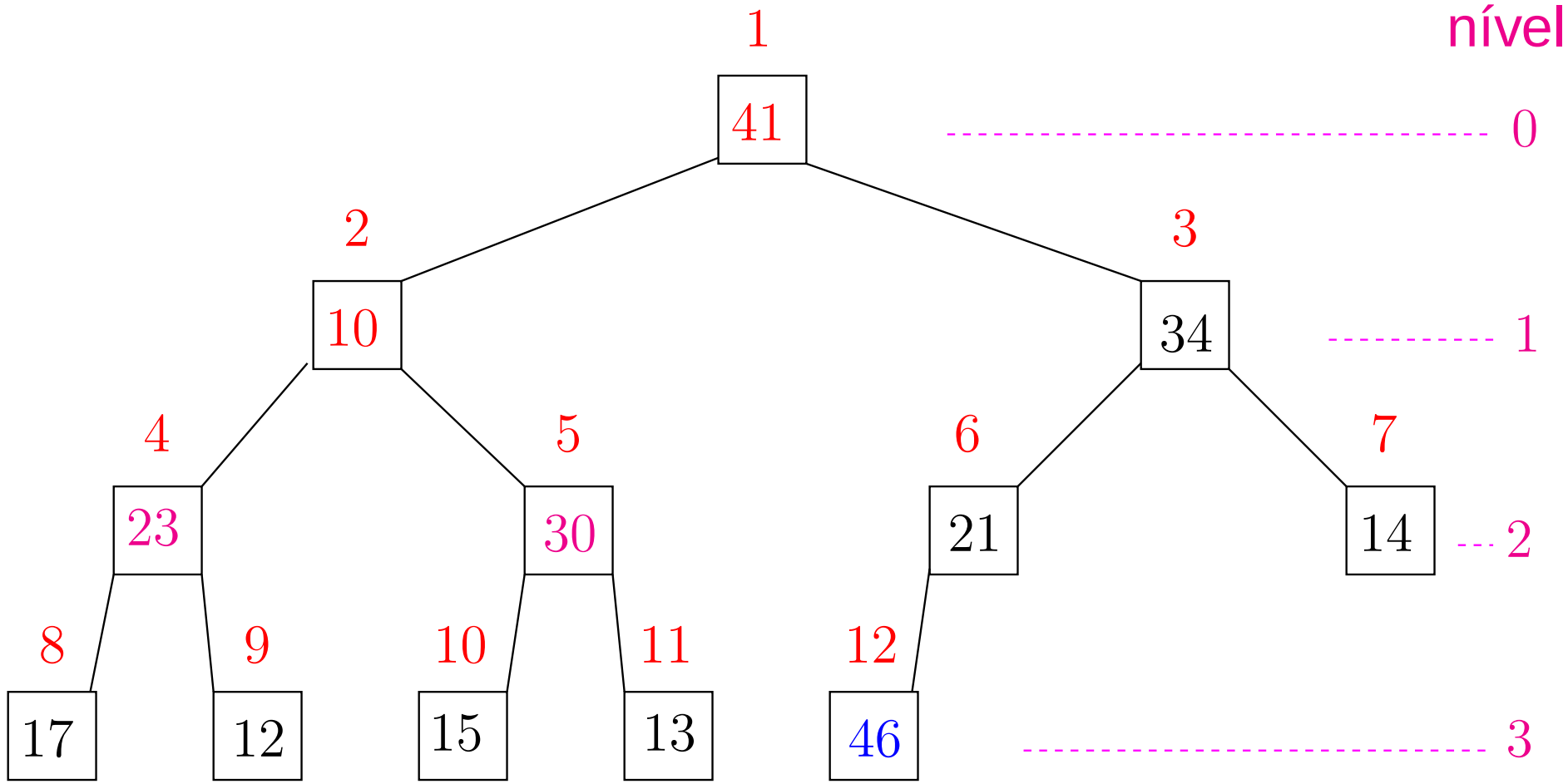
1	2	3	4	5	6	7	8	9	10	11	12
46	41	34	23	30	21	14	17	12	15	13	10

Heap sort



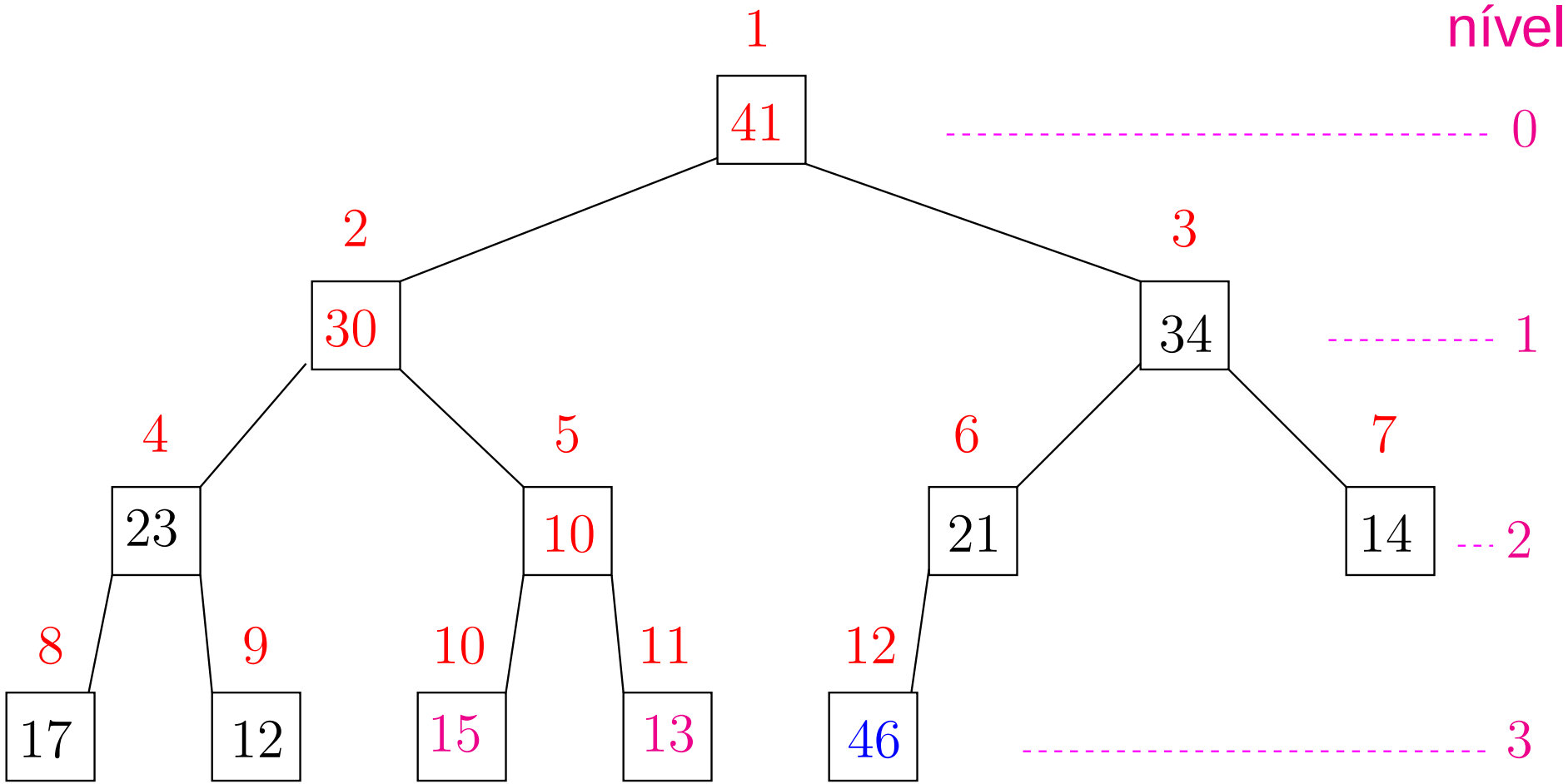
1	2	3	4	5	6	7	8	9	10	11	12
10	41	34	23	30	21	14	17	12	15	13	46

Heap sort



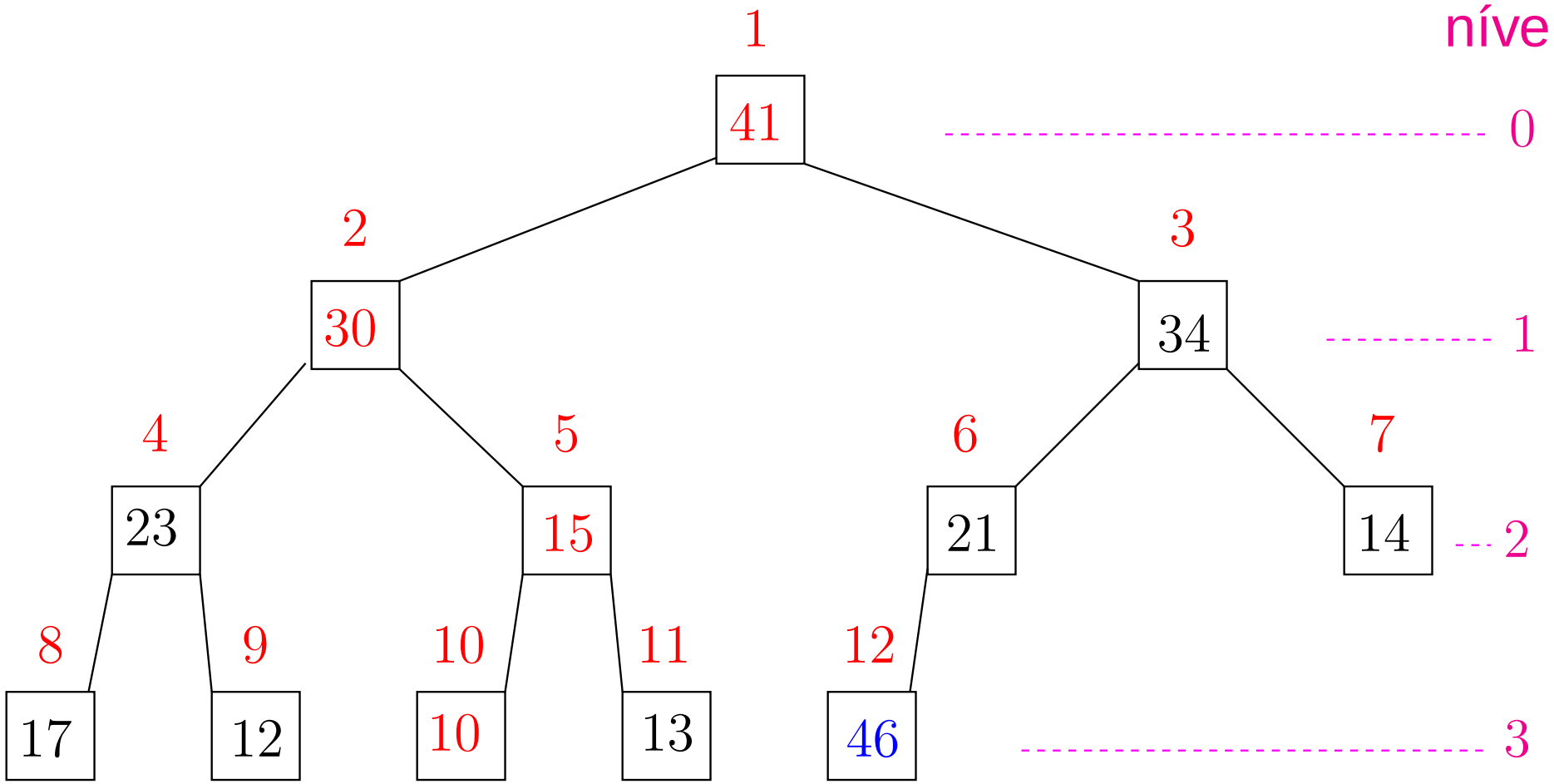
1	2	3	4	5	6	7	8	9	10	11	12
41	10	34	23	30	21	14	17	12	15	13	46

Heap sort



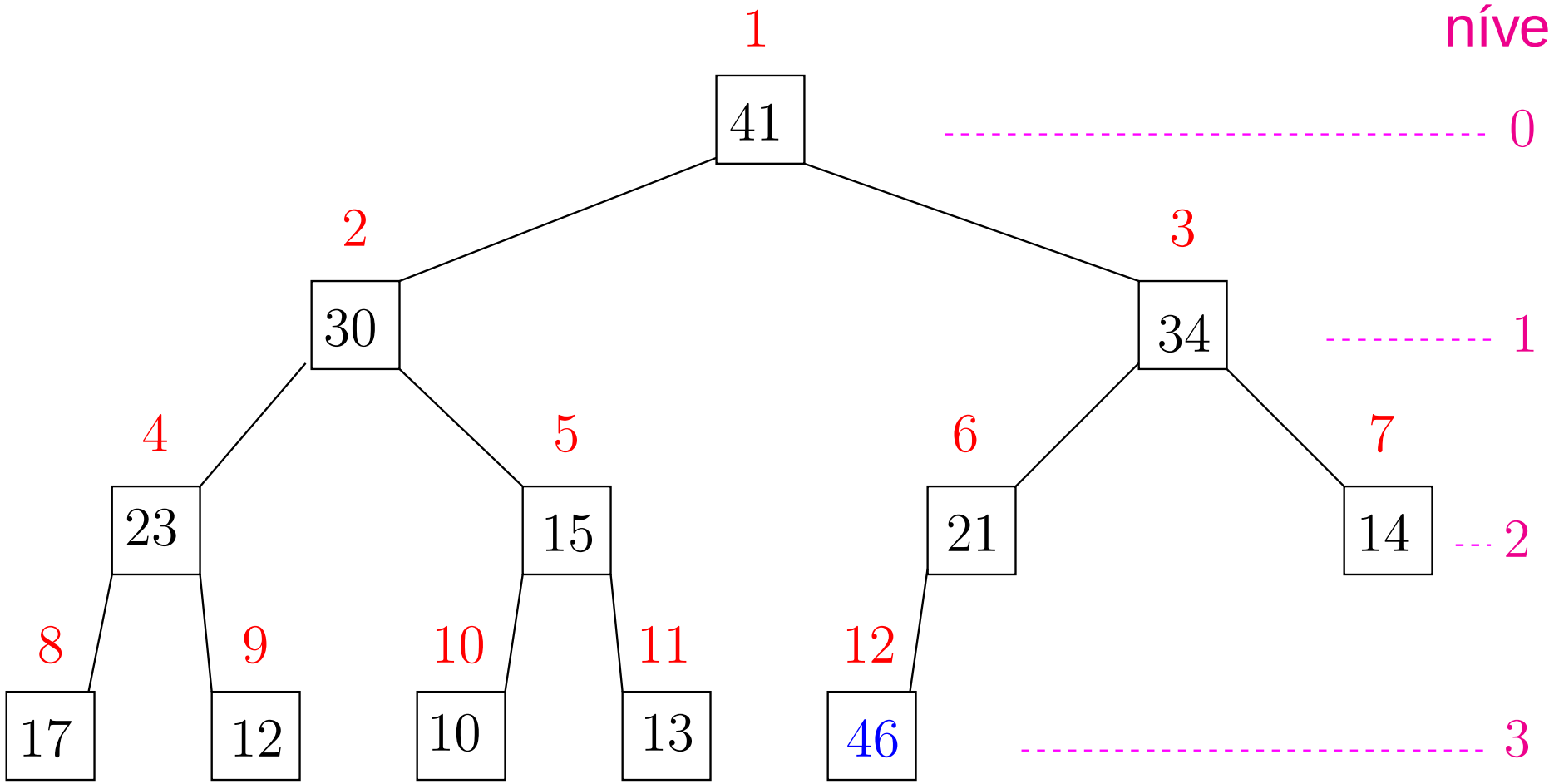
1	2	3	4	5	6	7	8	9	10	11	12
41	30	34	23	10	21	14	17	12	15	13	46

Heap sort



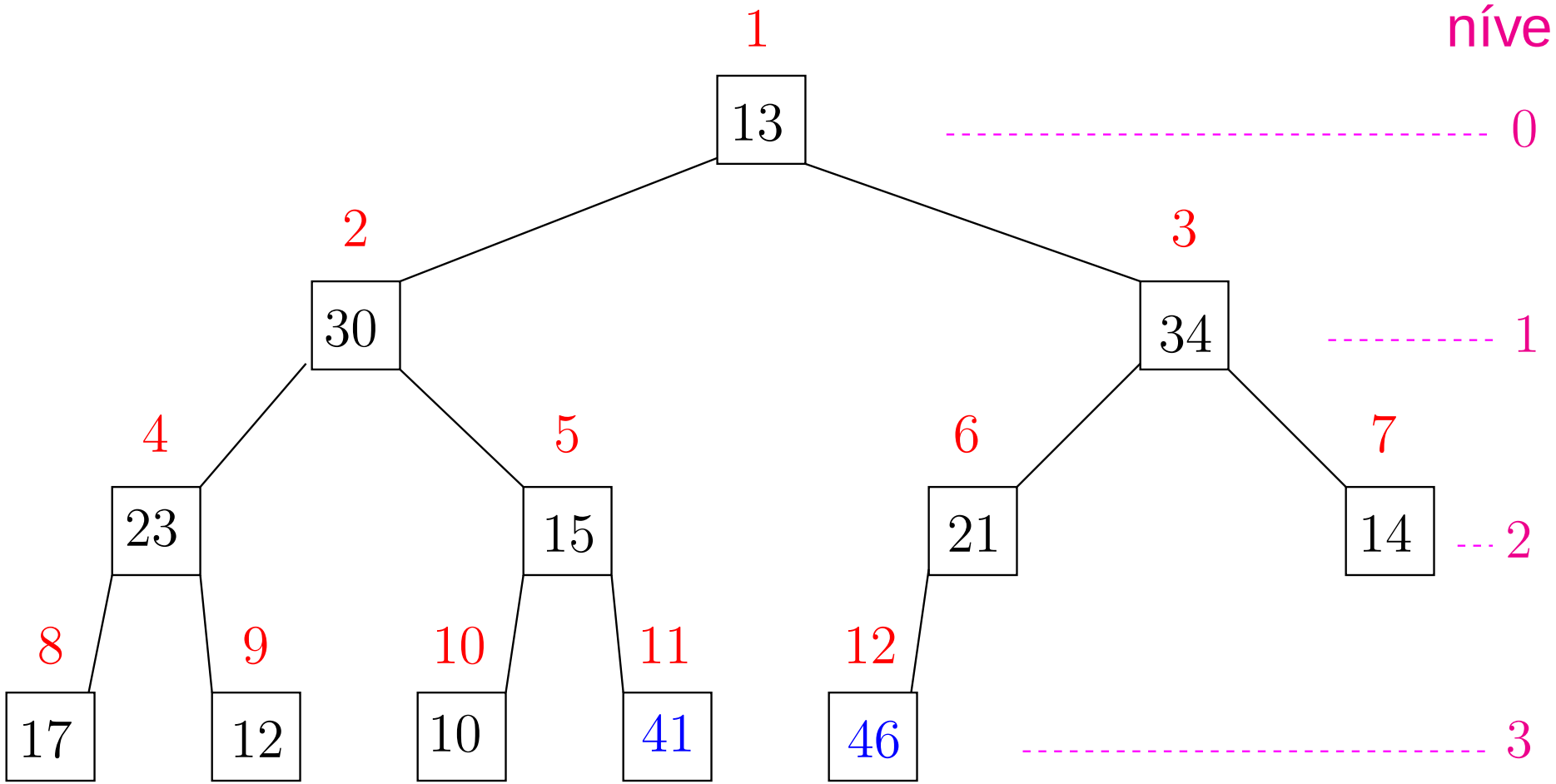
1	2	3	4	5	6	7	8	9	10	11	12
41	30	34	23	15	21	14	17	12	10	13	46

Heap sort



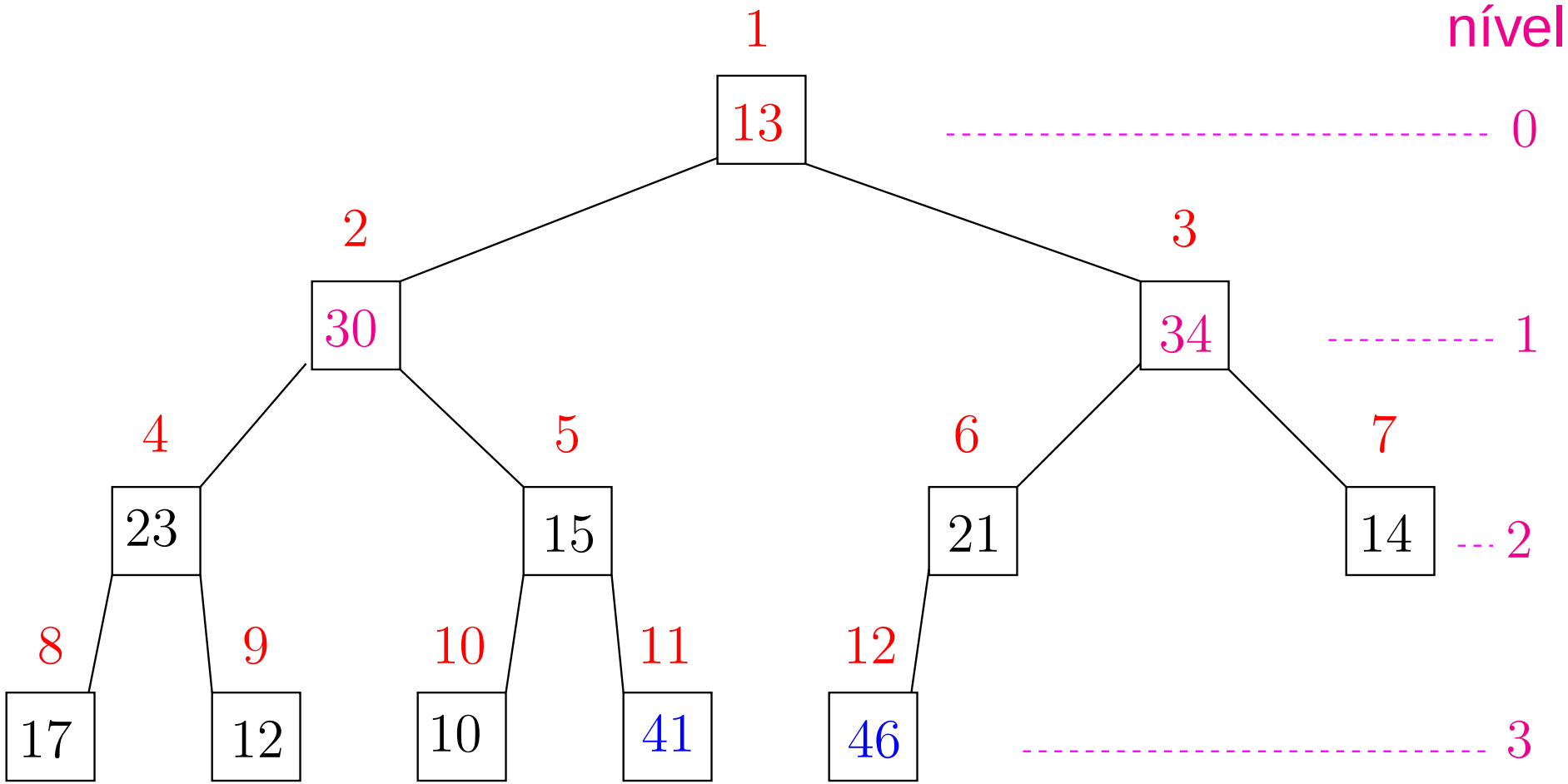
1	2	3	4	5	6	7	8	9	10	11	12
41	30	34	23	15	21	14	17	12	10	13	46

Heap sort



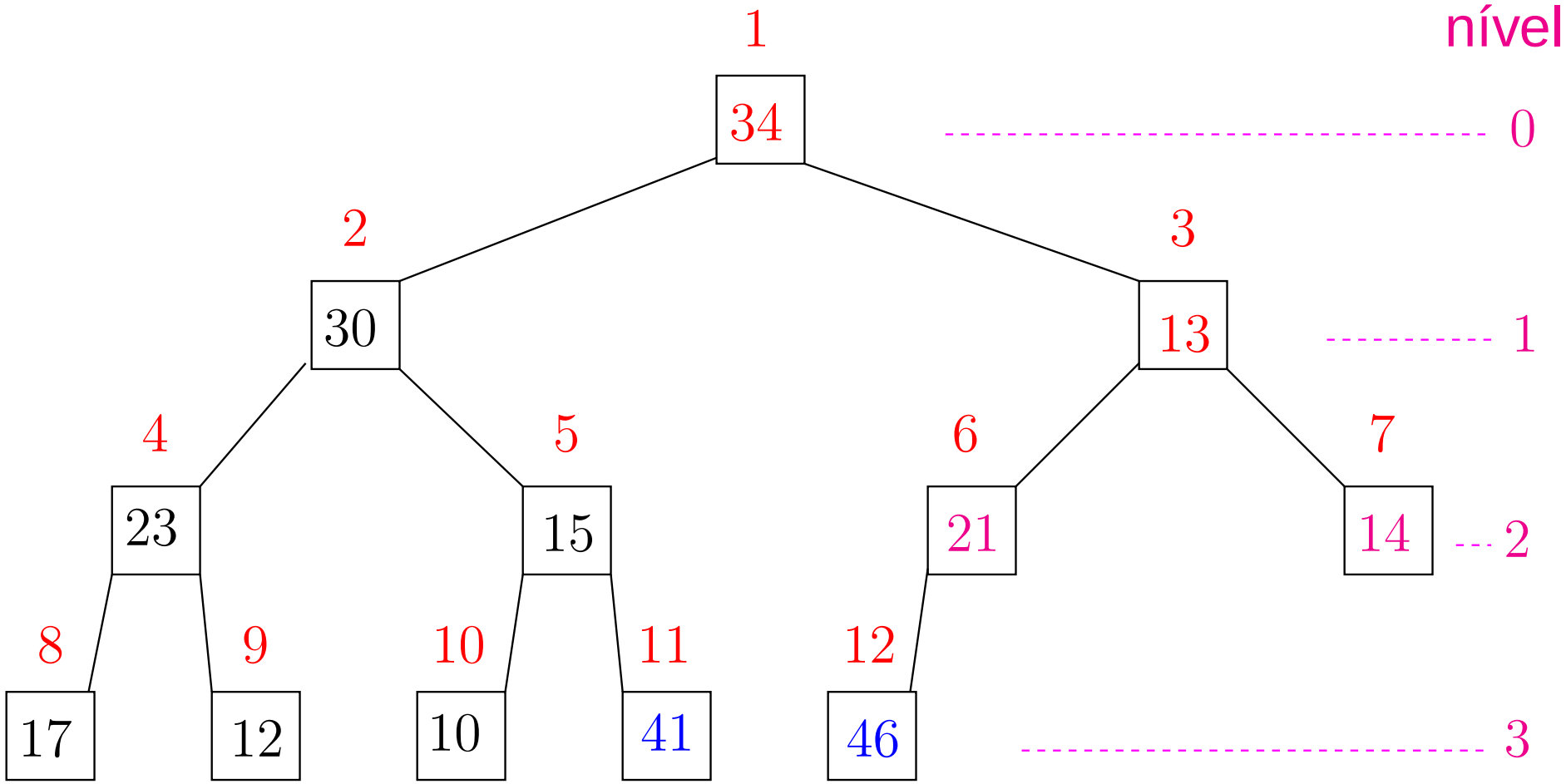
1	2	3	4	5	6	7	8	9	10	11	12
13	30	34	23	15	21	14	17	12	10	41	46

Heap sort



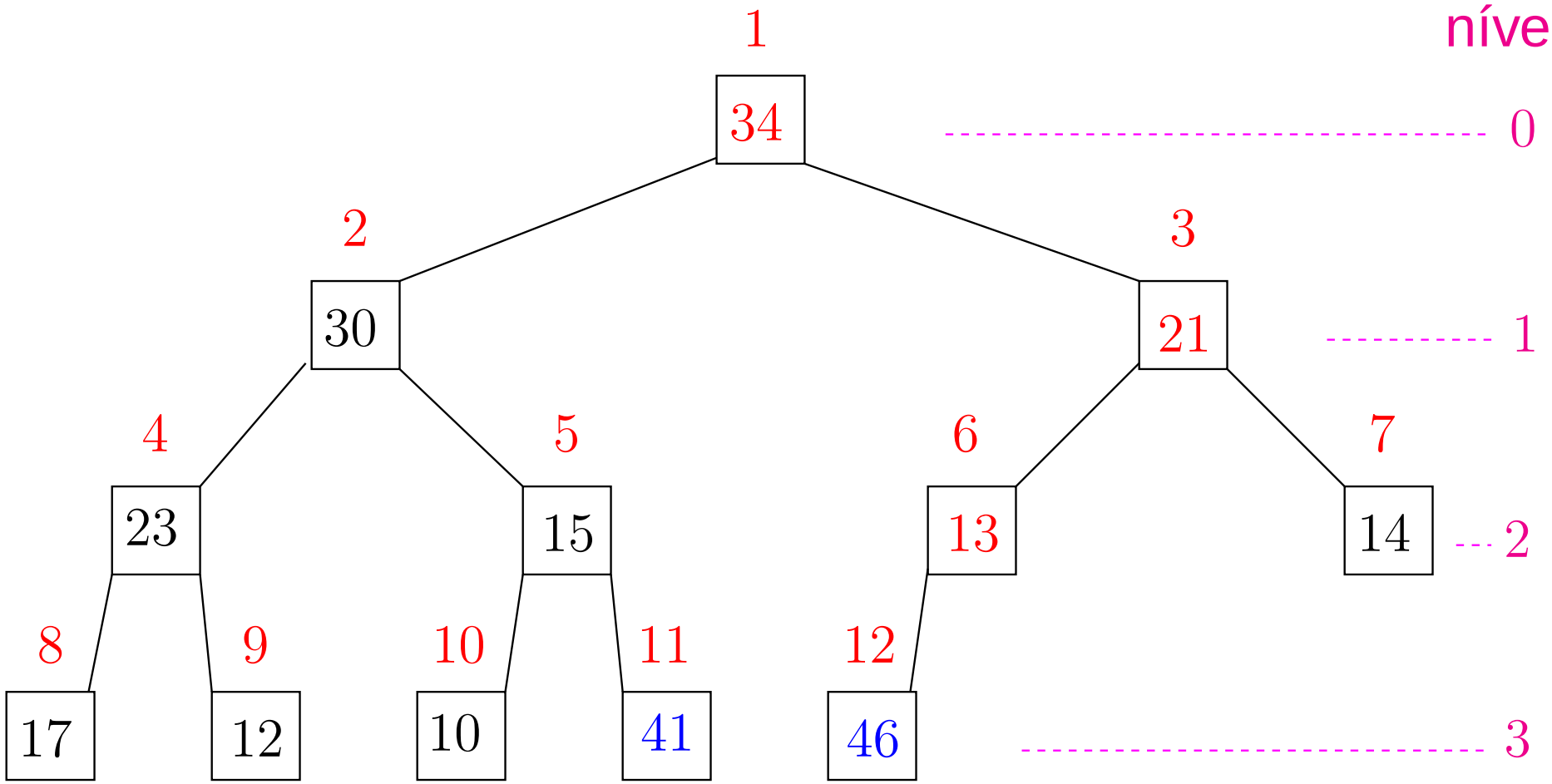
1	2	3	4	5	6	7	8	9	10	11	12
13	30	34	23	15	21	14	17	12	10	41	46

Heap sort



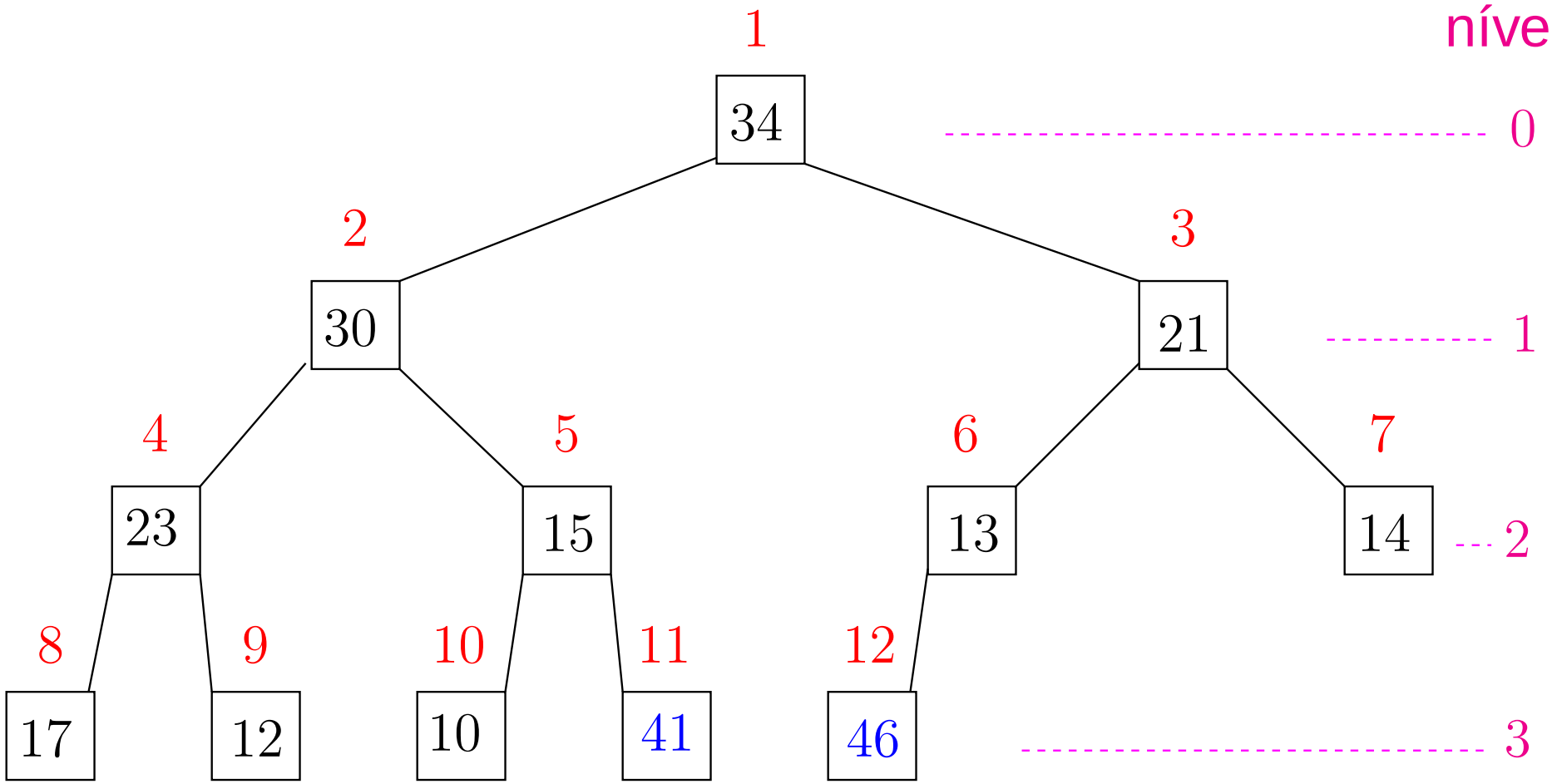
1	2	3	4	5	6	7	8	9	10	11	12
34	30	13	23	15	21	14	17	12	10	41	46

Heap sort



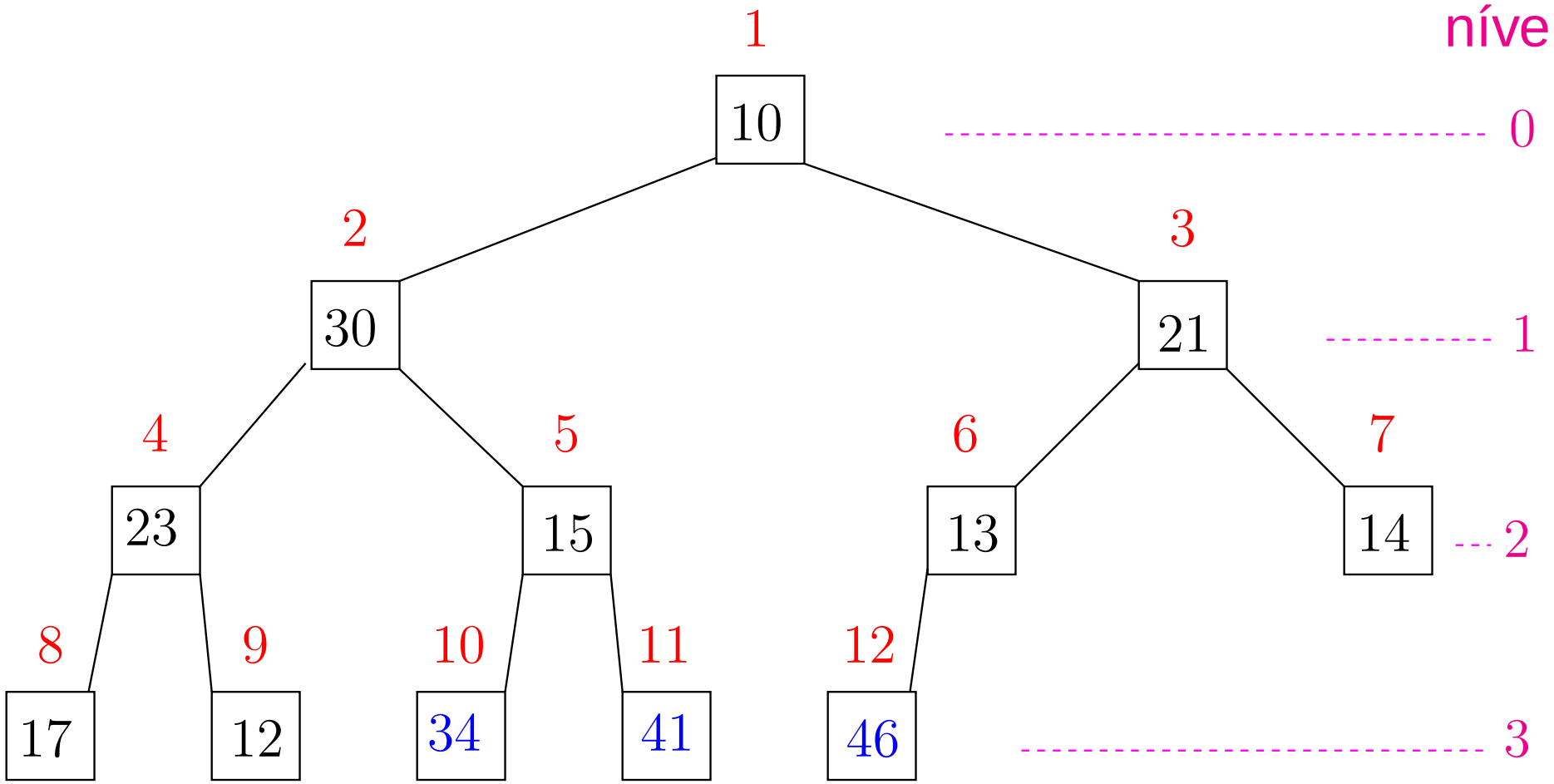
1	2	3	4	5	6	7	8	9	10	11	12
34	30	21	23	15	13	14	17	12	10	41	46

Heap sort



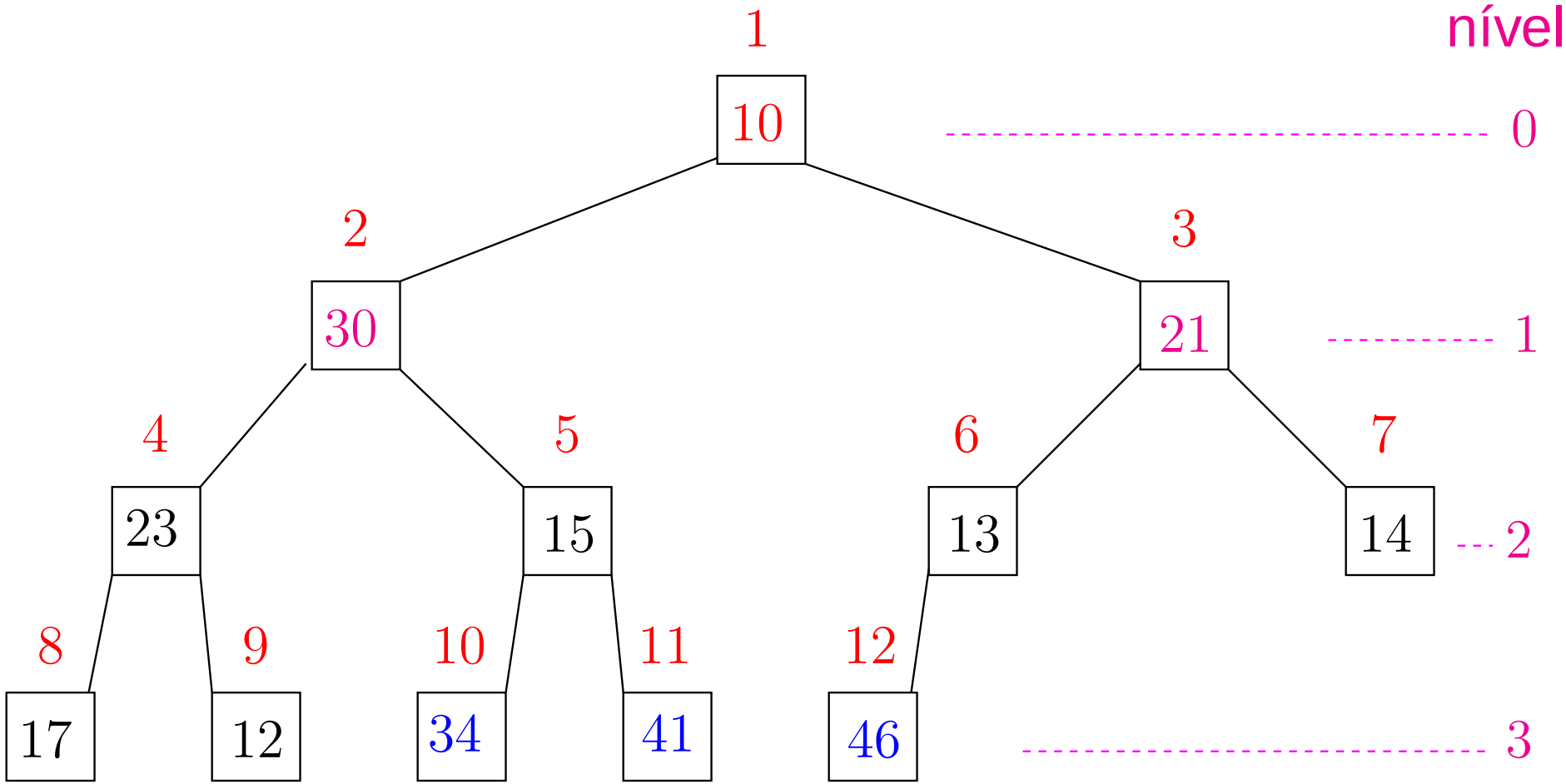
1	2	3	4	5	6	7	8	9	10	11	12
34	30	21	23	15	13	14	17	12	10	41	46

Heap sort



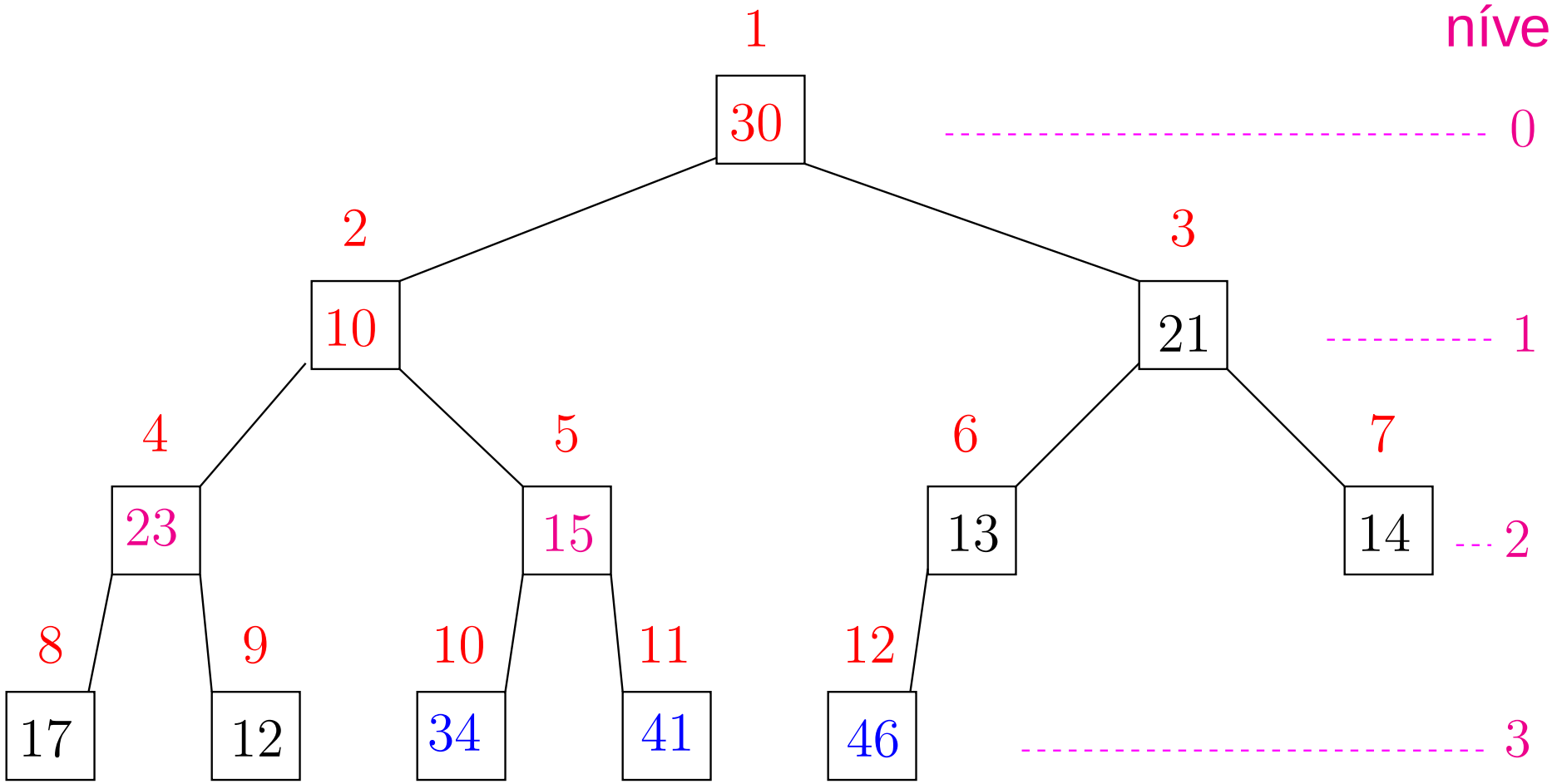
1	2	3	4	5	6	7	8	9	10	11	12
10	30	21	23	15	13	14	17	12	34	41	46

Heap sort



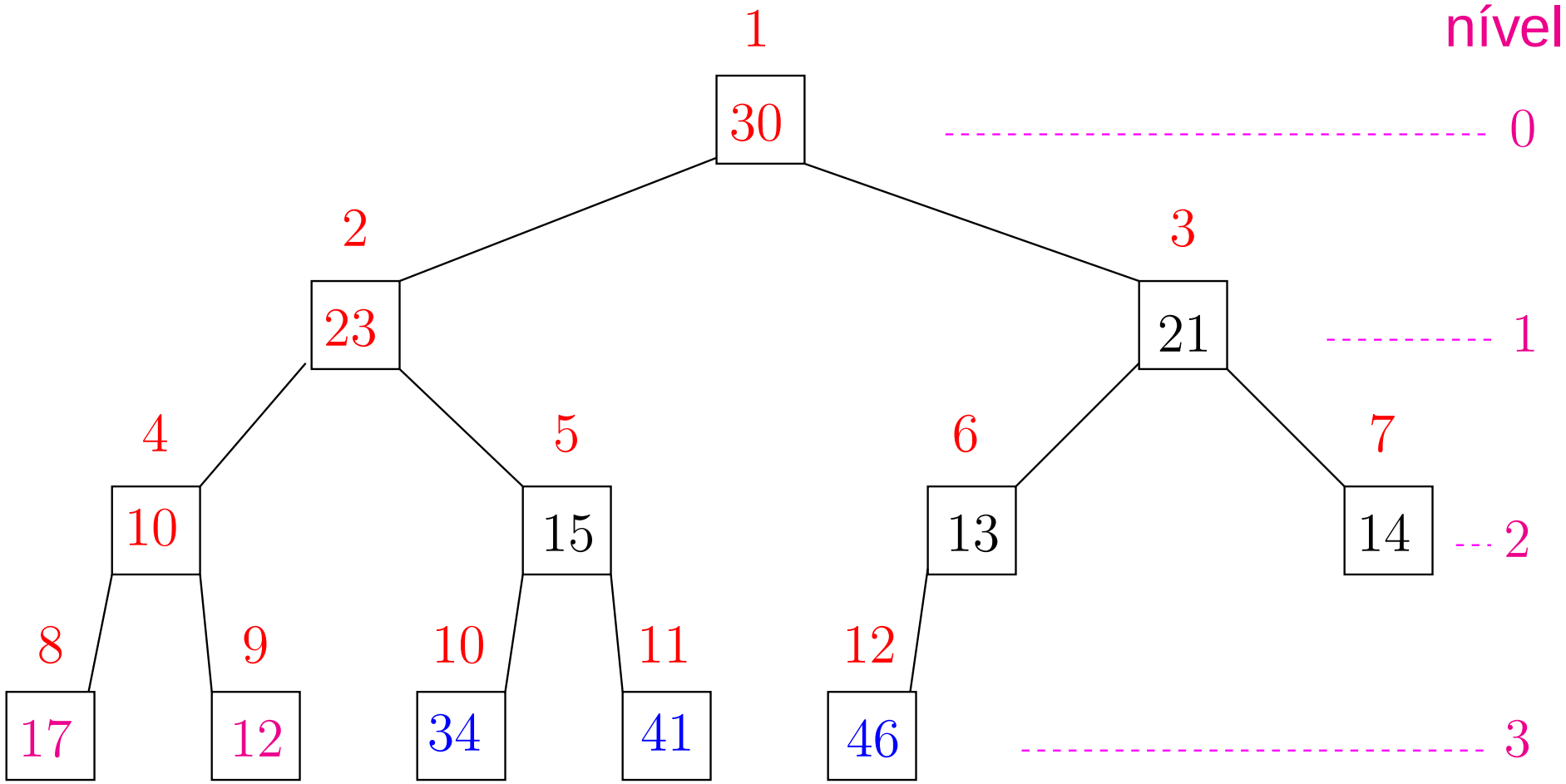
1	2	3	4	5	6	7	8	9	10	11	12
10	30	21	23	15	13	14	17	12	34	41	46

Heap sort



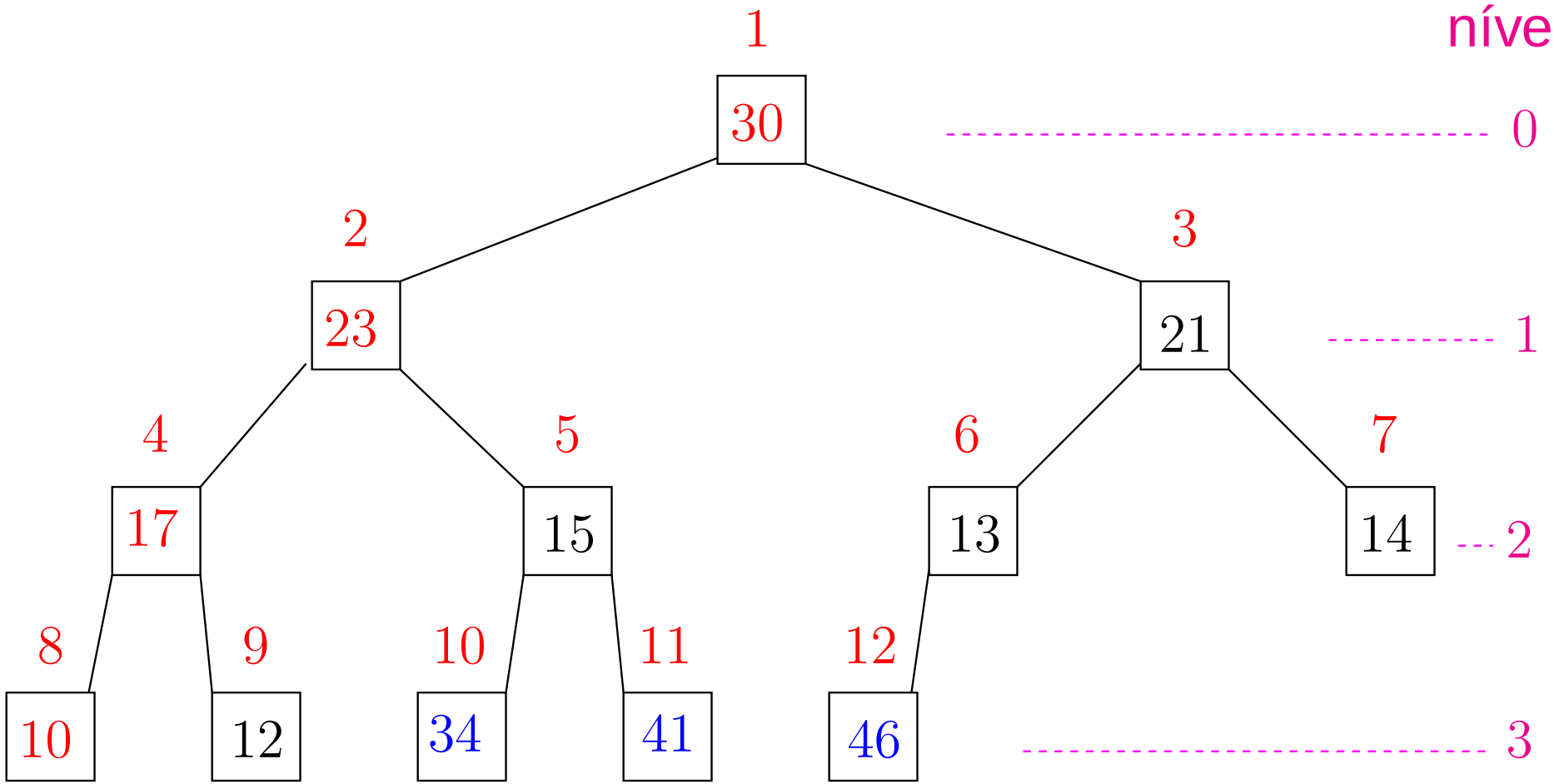
1	2	3	4	5	6	7	8	9	10	11	12
30	10	21	23	15	13	14	17	12	34	41	46

Heap sort



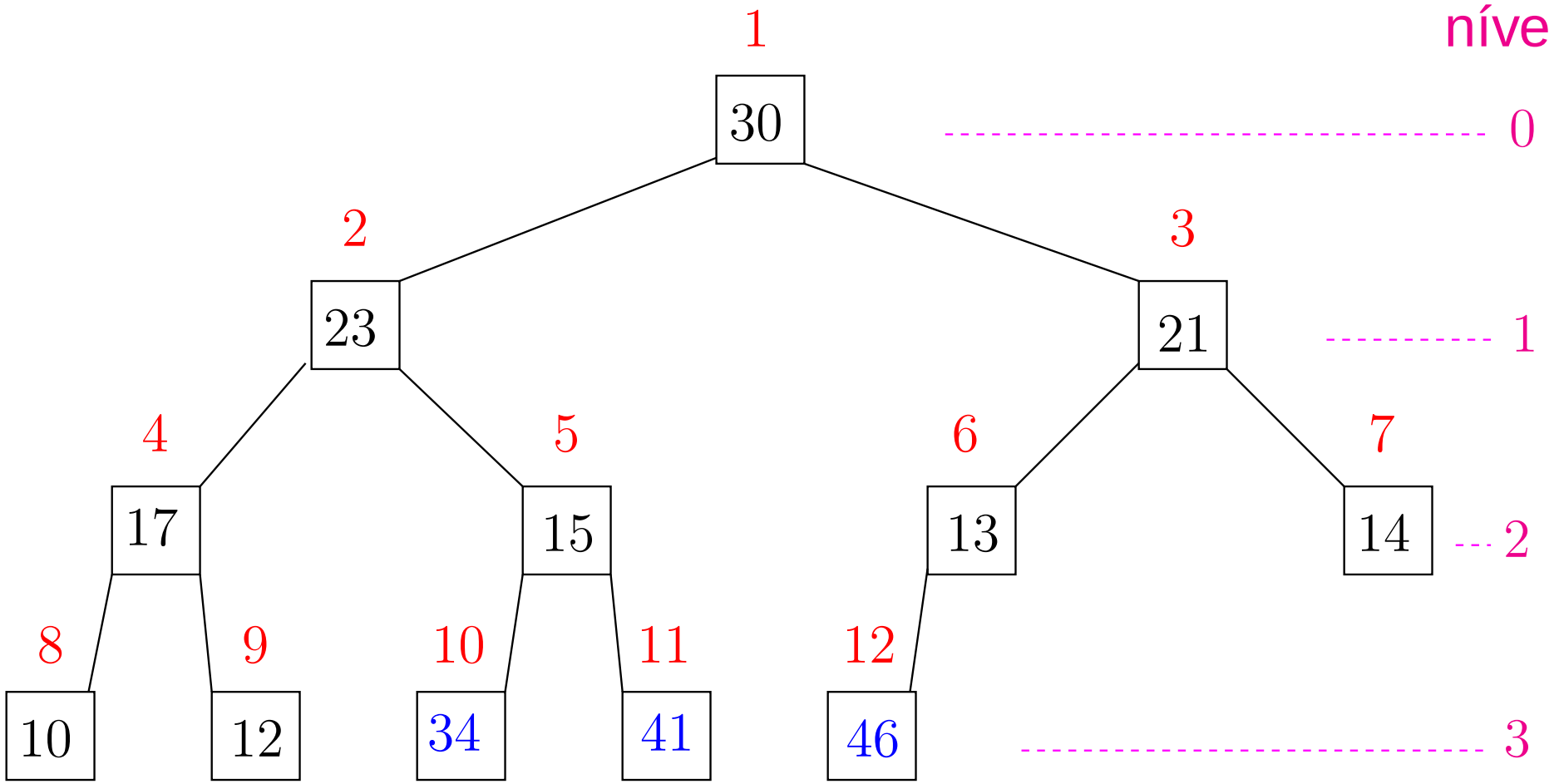
1	2	3	4	5	6	7	8	9	10	11	12
30	23	21	10	15	13	14	17	12	34	41	46

Heap sort



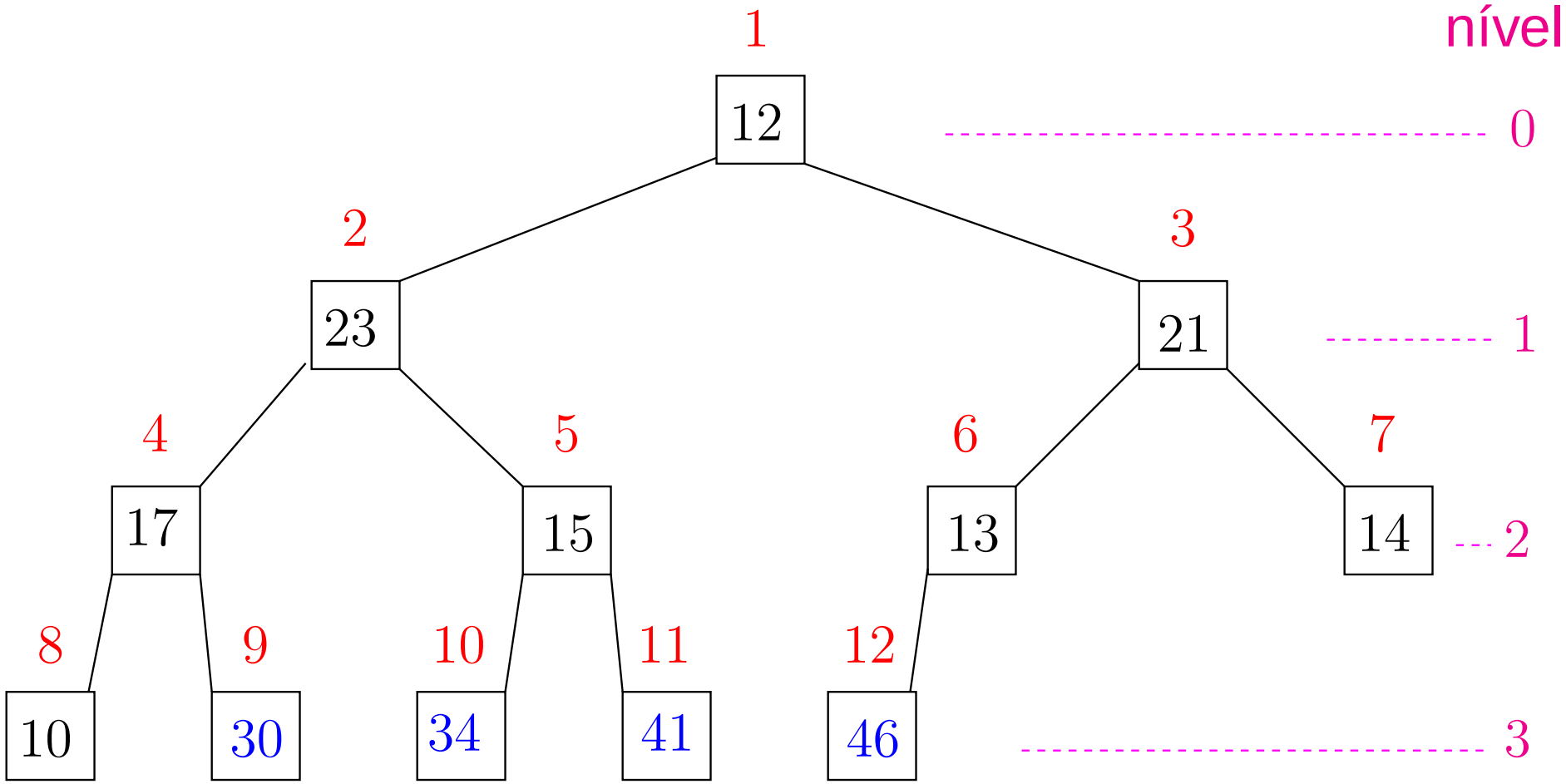
1	2	3	4	5	6	7	8	9	10	11	12
30	23	21	17	15	13	14	10	12	34	41	46

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)        ▷ 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)
```

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 ← 2i
    d ← 2i + 1
    se e ≤ m e A[e] > A[i]
            então A[i] ↔ A[e]
                    D-H (A, m, e)
    se d ≤ m e A[d] > A[i]
            então A[i] ↔ A[d]
                    D-H (A, m, d)
```