

Ordenação por seleção

1			<i>i</i>							<i>n</i>
38	10	20	44	50	50	55	60	75	85	99

1			<i>i</i>							<i>n</i>
20	10	38	44	50	50	55	60	75	85	99

1	<i>i</i>									<i>n</i>
20	10	38	44	50	50	55	60	75	85	99

Navigation icons

Ordenação por seleção

1			<i>i</i>							<i>n</i>
38	10	20	44	50	50	55	60	75	85	99

1			<i>i</i>							<i>n</i>
20	10	38	44	50	50	55	60	75	85	99

1	<i>i</i>									<i>n</i>
10	20	38	44	50	50	55	60	75	85	99

1										<i>n</i>
10	20	38	44	50	50	55	60	75	85	99

Navigation icons

Função selecao

Algoritmo rearranja $v[0..n-1]$ em ordem crescente

```
void selecao (int n, int v[])
{
    int i, j, max, x;
    1 for (i = n-1; /*B*/ i > 0; i--) {
    2     max = i;
    3     for (j = i-1; j >= 0; j--)
    4         if (v[j] > v[max]) max = j;
    5     x=v[i]; v[i]=v[max]; v[max]=x;
    }
}
```

Navigation icons

Função selecao

Algoritmo rearranja $v[1..n]$ em ordem crescente

```
void selecao (int n, int v[])
{
    int i, j, max, x;
    1 for (i = n; /*B*/ i >= 2; i--) {
    2     max = i;
    3     for (j = i-1; j >= 1; j--)
    4         if (v[j] > v[max]) max = j;
    5     x=v[i]; v[i]=v[max]; v[max]=x;
    }
}
```

Navigation icons

Função selecao

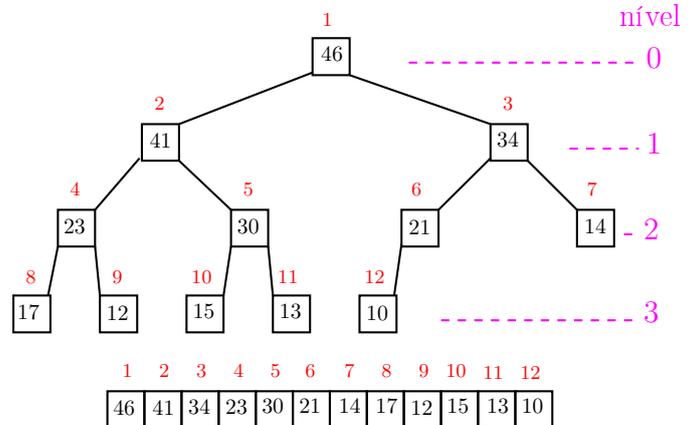
Relações invariantes: Em /*B*/ vale que:

- (i0) $v[i+1..n]$ é crescente;
- (i1) $v[1..i] \leq v[i+1]$;

1			<i>i</i>							<i>n</i>
38	10	20	44	50	50	55	60	75	85	99

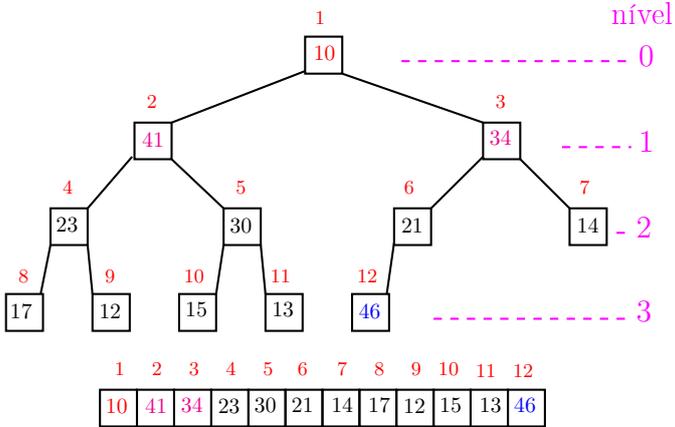
Navigation icons

Heapsort



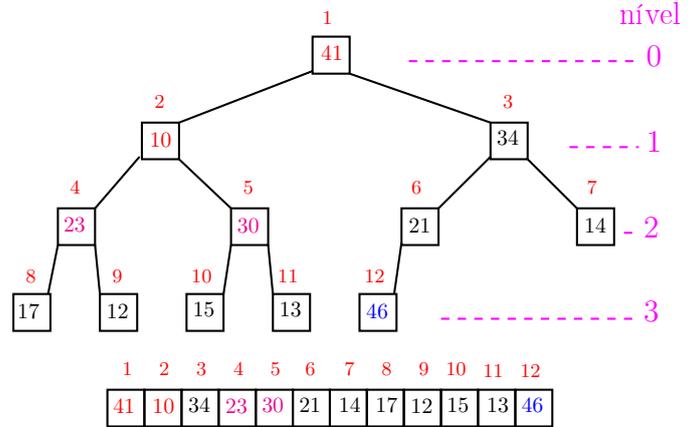
Navigation icons

Heapsort



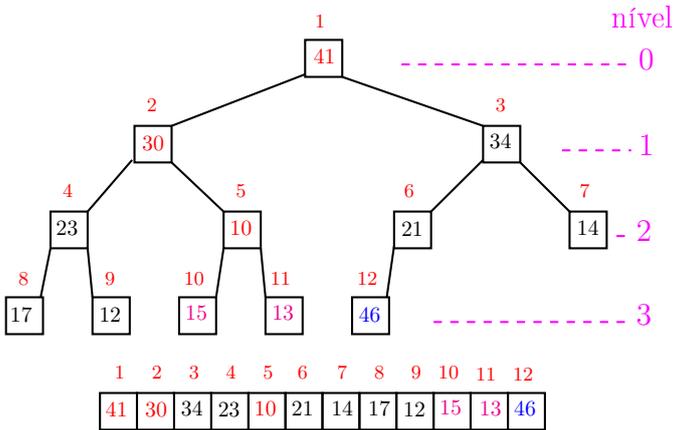
Navigation icons

Heapsort



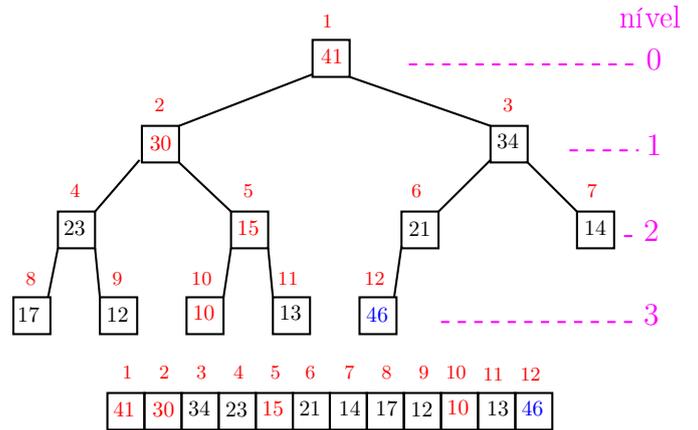
Navigation icons

Heapsort



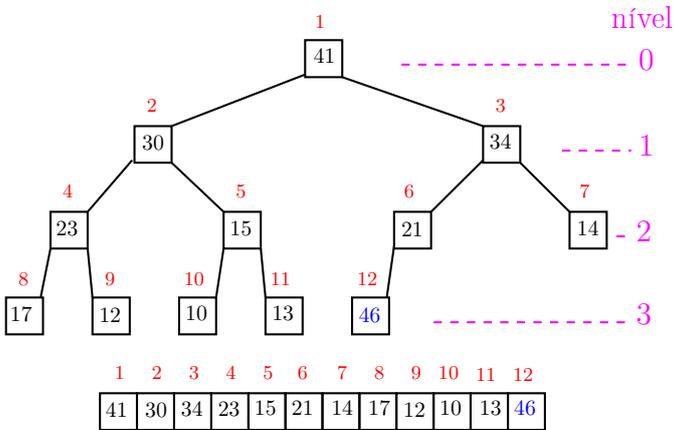
Navigation icons

Heapsort



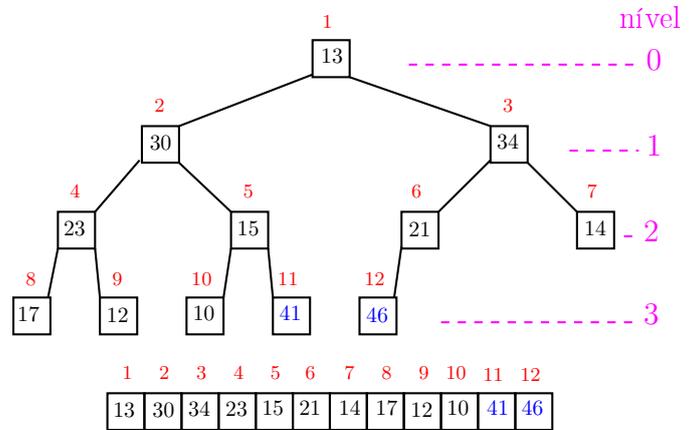
Navigation icons

Heapsort



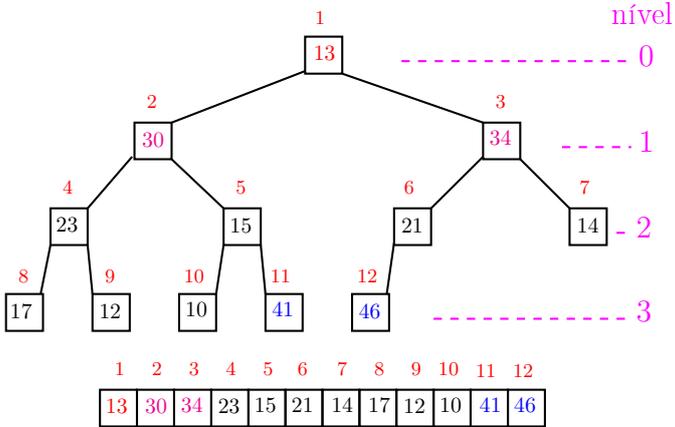
Navigation icons

Heapsort



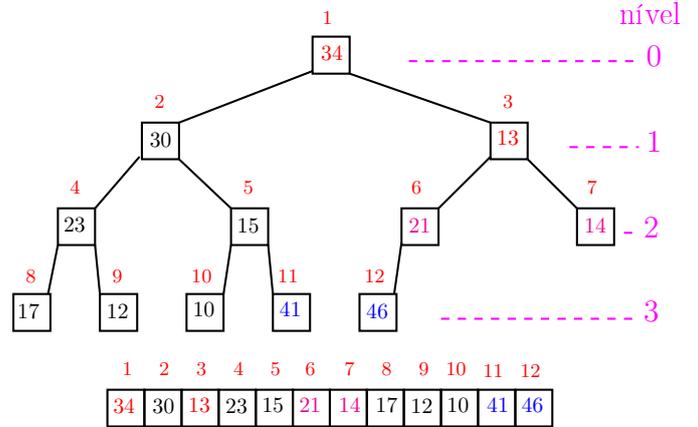
Navigation icons

Heapsort



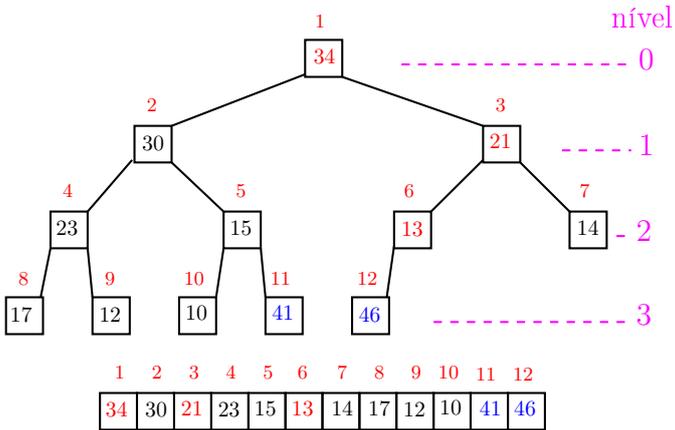
Navigation icons

Heapsort



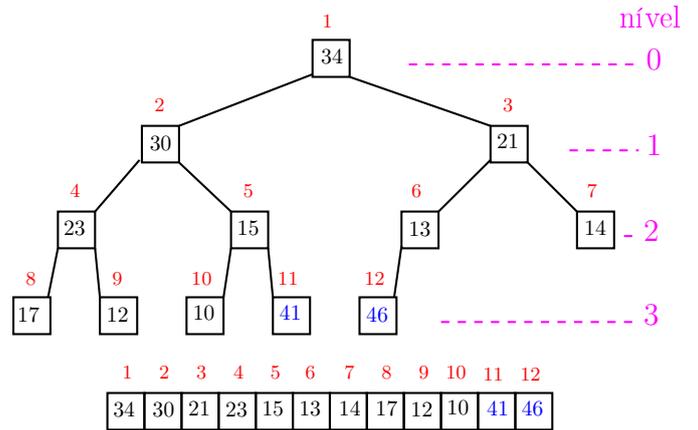
Navigation icons

Heapsort



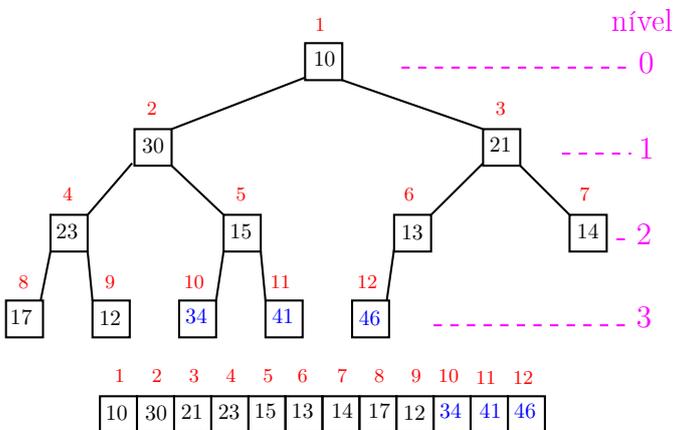
Navigation icons

Heapsort



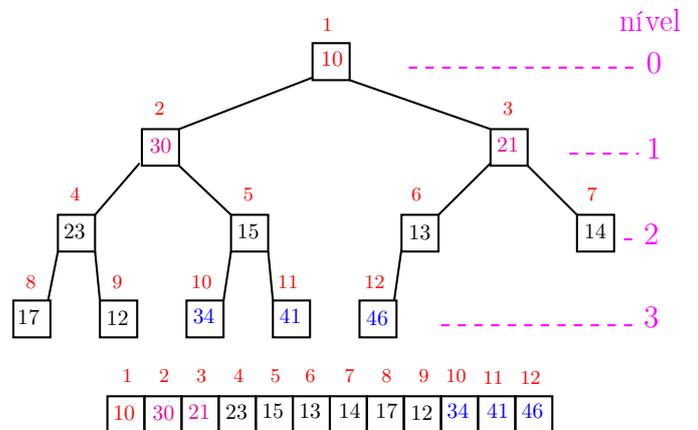
Navigation icons

Heapsort



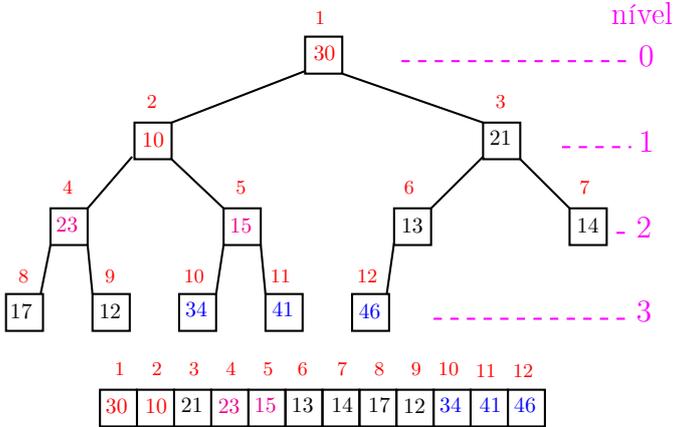
Navigation icons

Heapsort



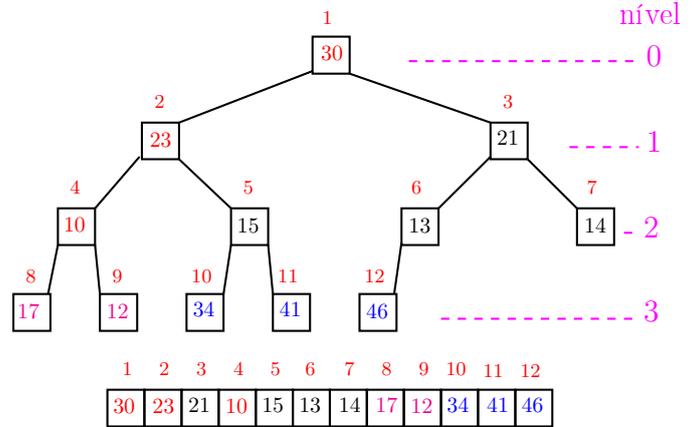
Navigation icons

Heapsort



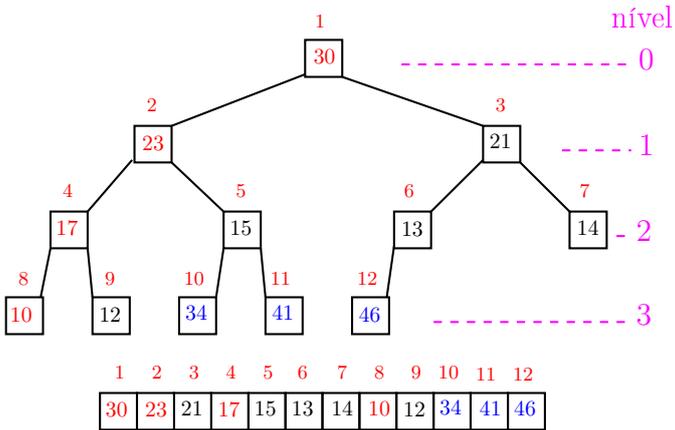
Navigation icons

Heapsort



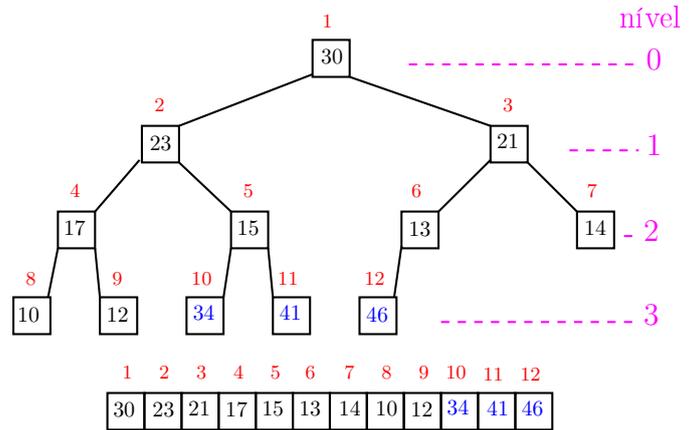
Navigation icons

Heapsort



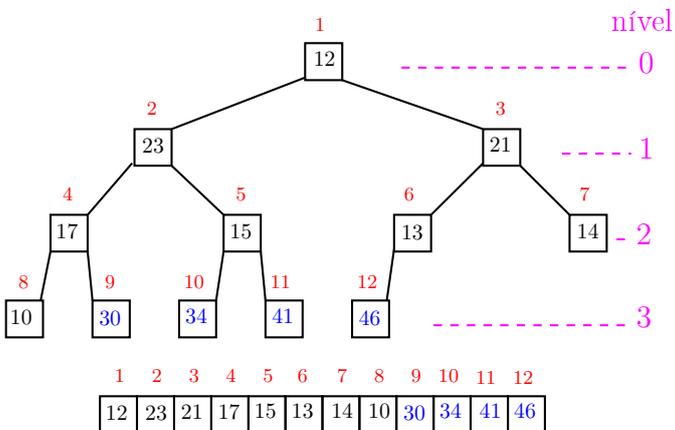
Navigation icons

Heapsort



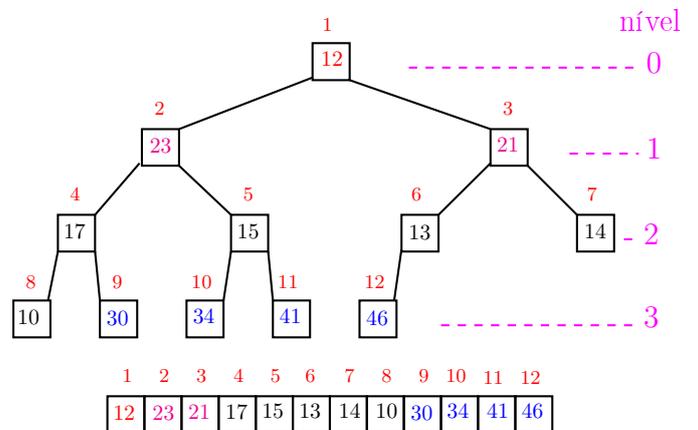
Navigation icons

Heapsort



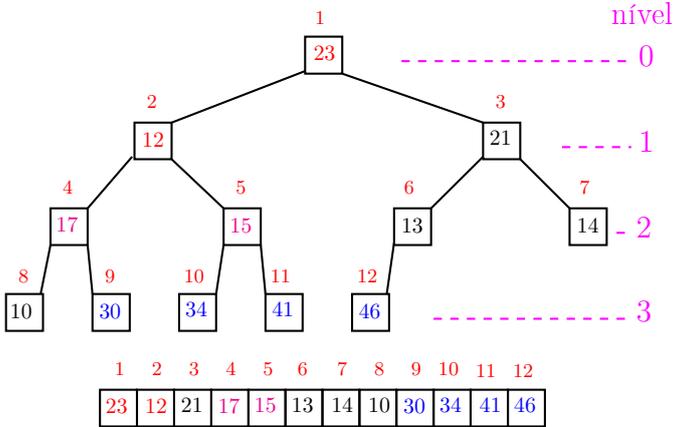
Navigation icons

Heapsort



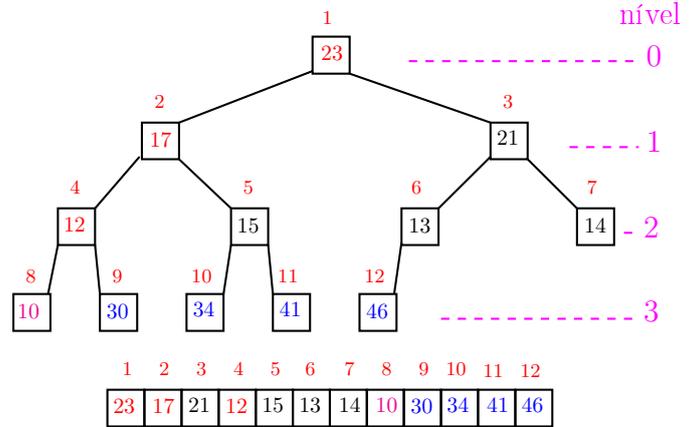
Navigation icons

Heapsort



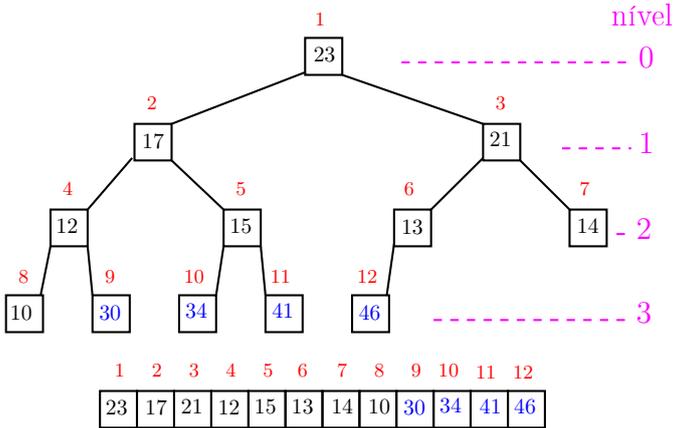
Navigation icons

Heapsort



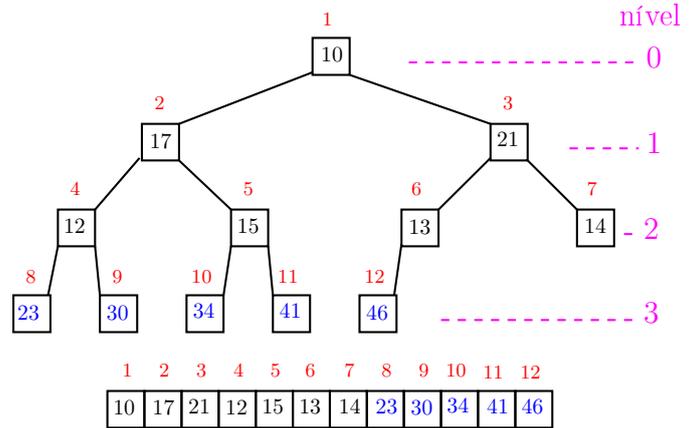
Navigation icons

Heapsort



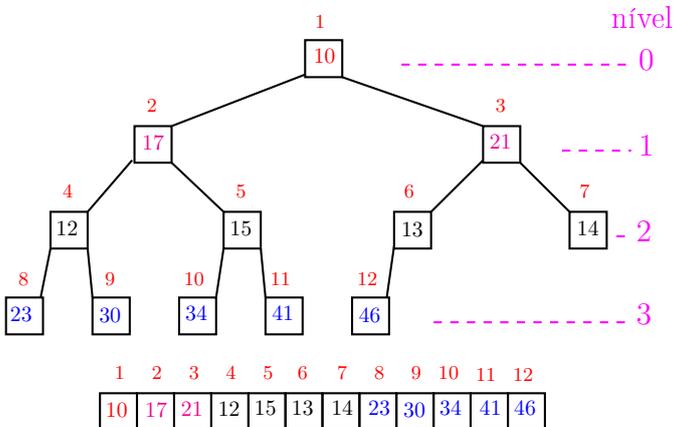
Navigation icons

Heapsort



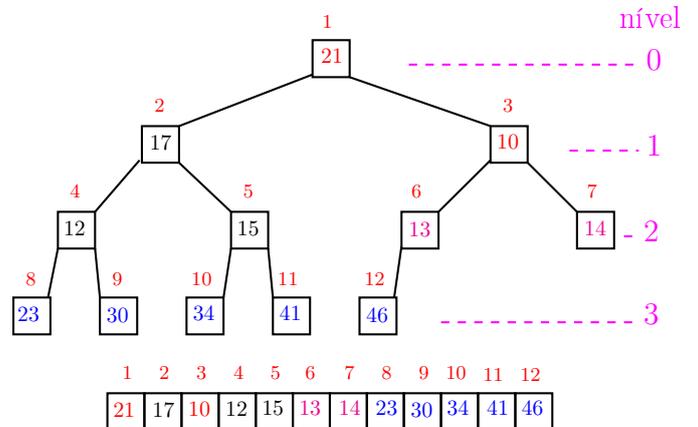
Navigation icons

Heapsort



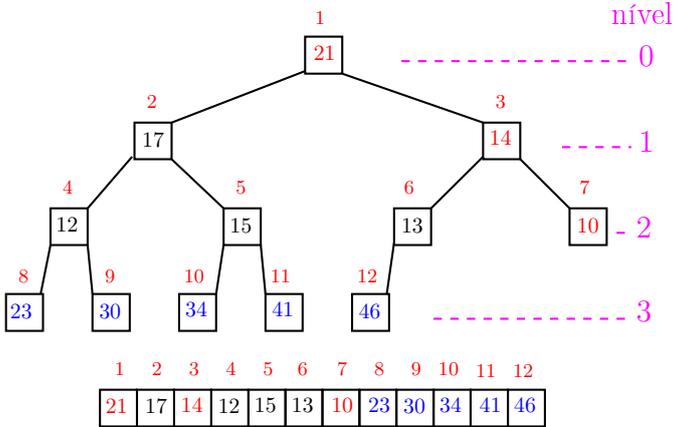
Navigation icons

Heapsort



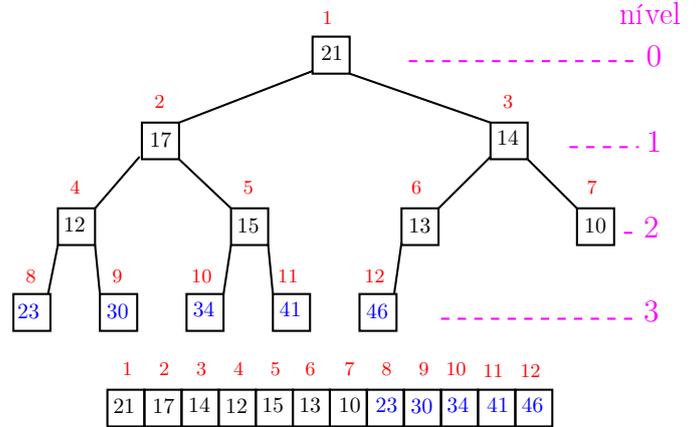
Navigation icons

Heapsort



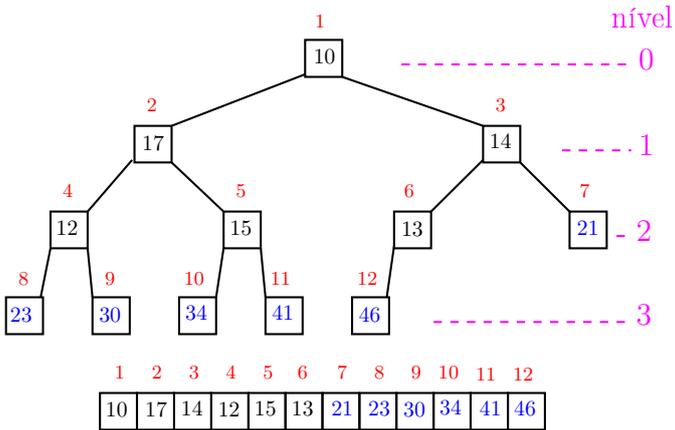
Navigation icons

Heapsort



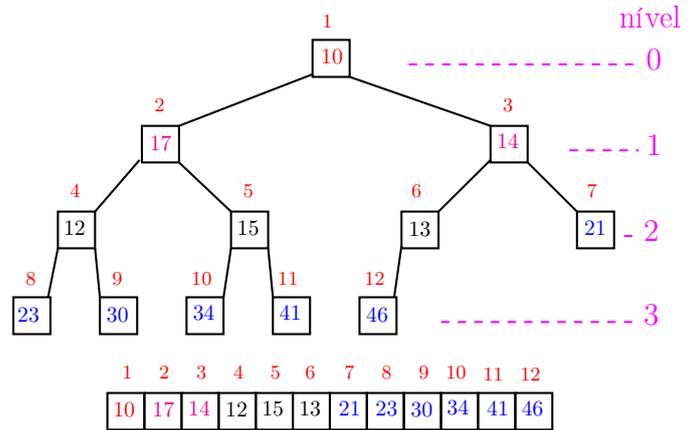
Navigation icons

Heapsort



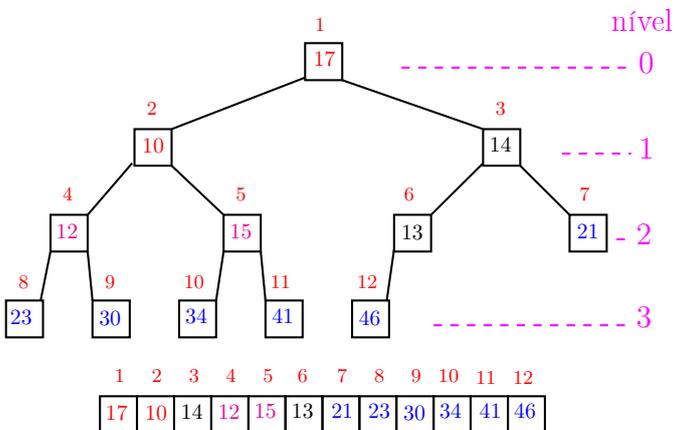
Navigation icons

Heapsort



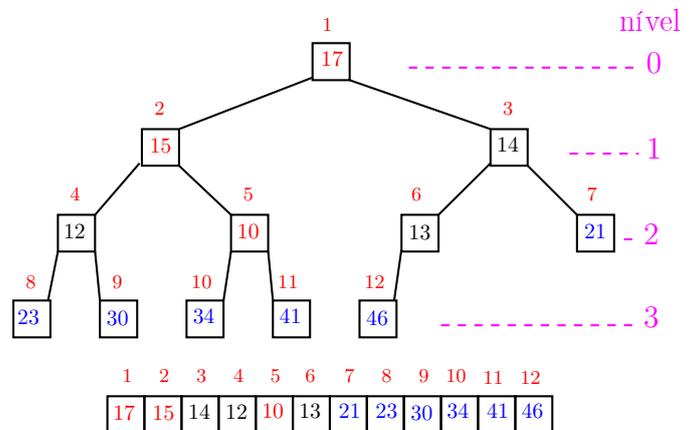
Navigation icons

Heapsort



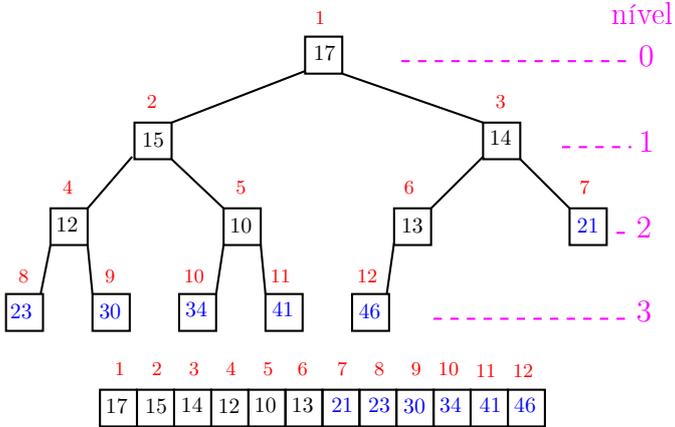
Navigation icons

Heapsort



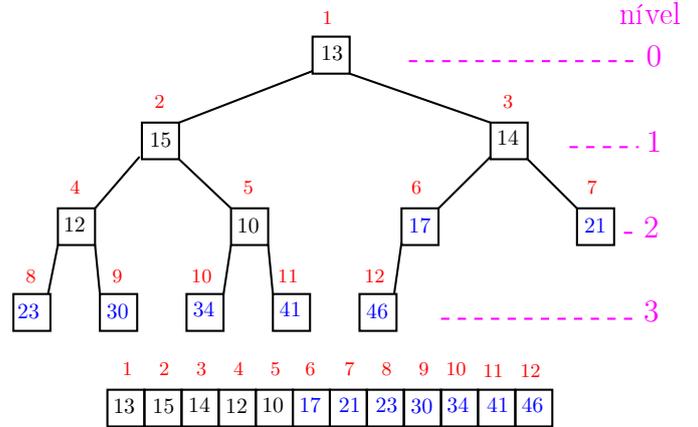
Navigation icons

Heapsort



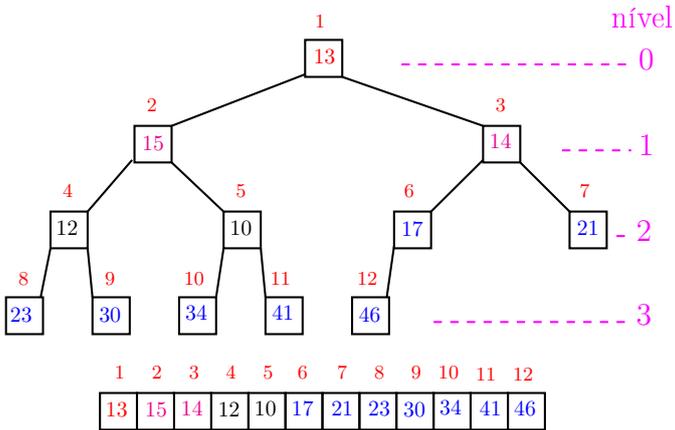
Navigation icons

Heapsort



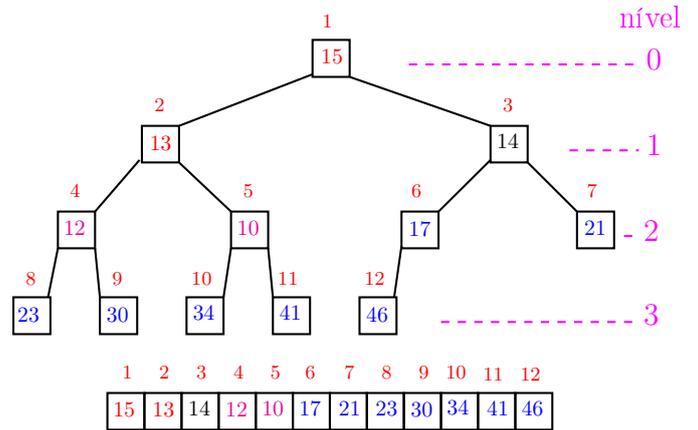
Navigation icons

Heapsort



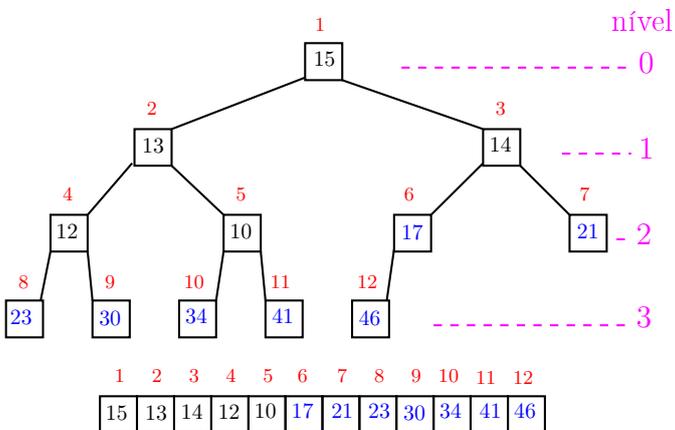
Navigation icons

Heapsort



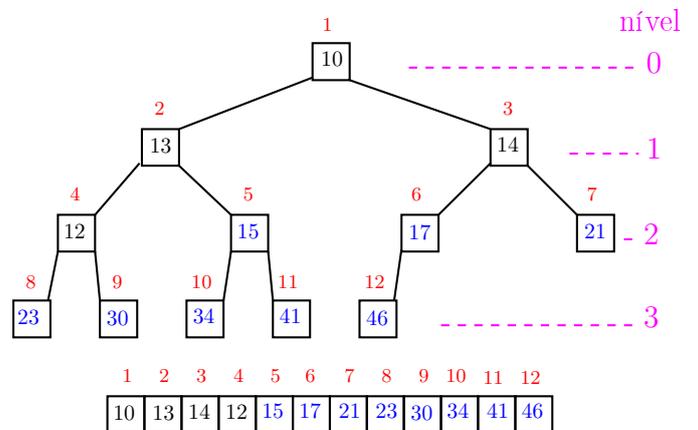
Navigation icons

Heapsort



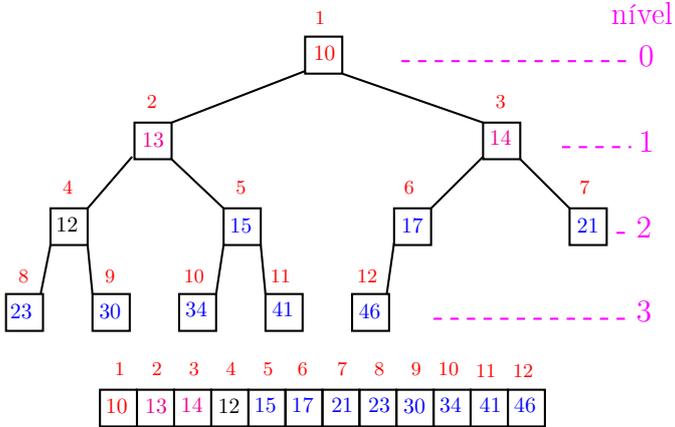
Navigation icons

Heapsort



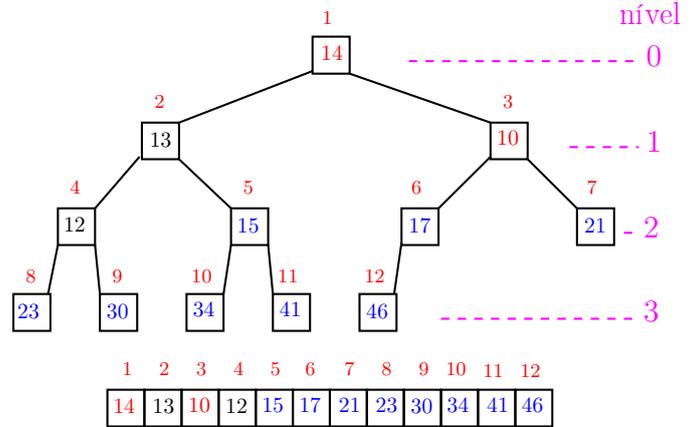
Navigation icons

Heapsort



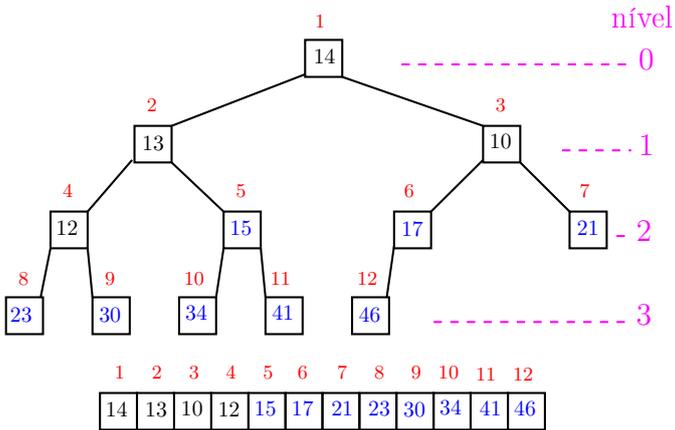
Navigation icons

Heapsort



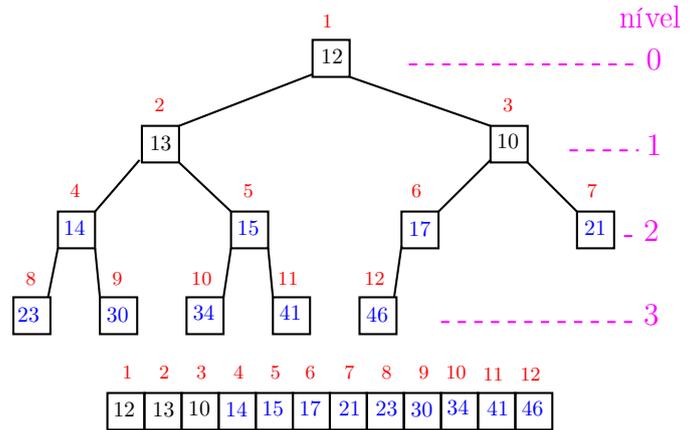
Navigation icons

Heapsort



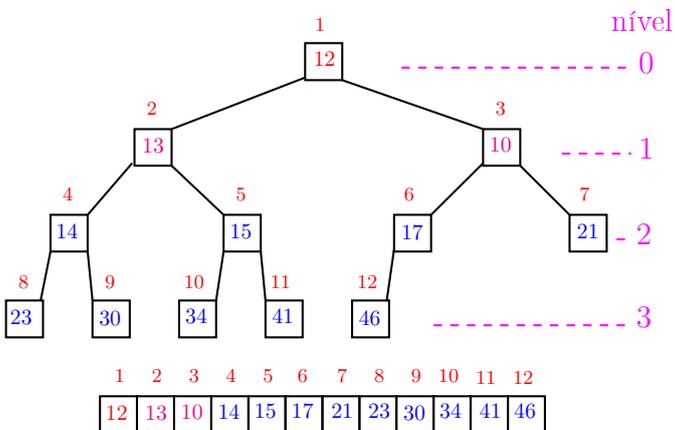
Navigation icons

Heapsort



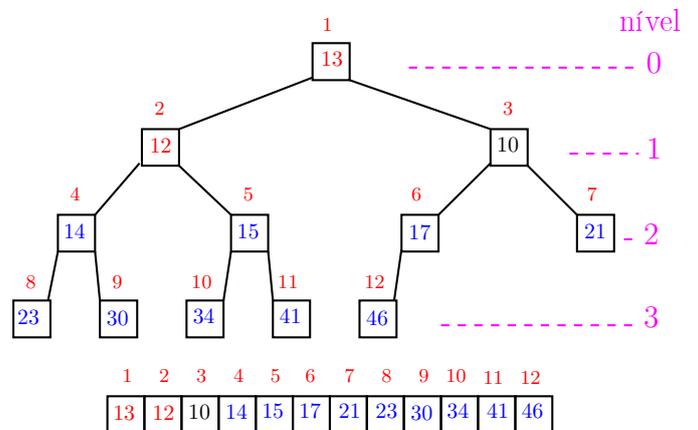
Navigation icons

Heapsort



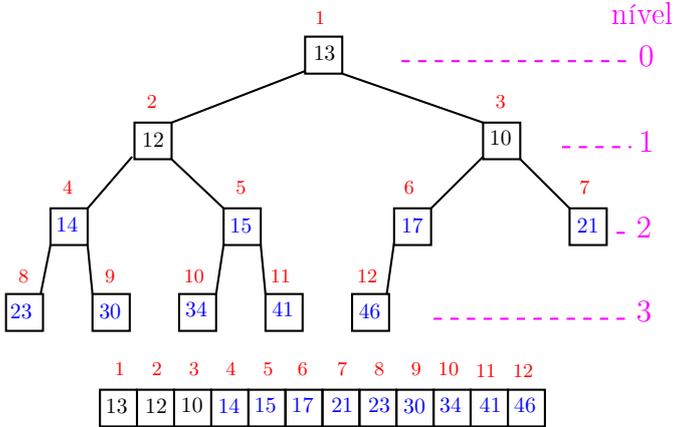
Navigation icons

Heapsort



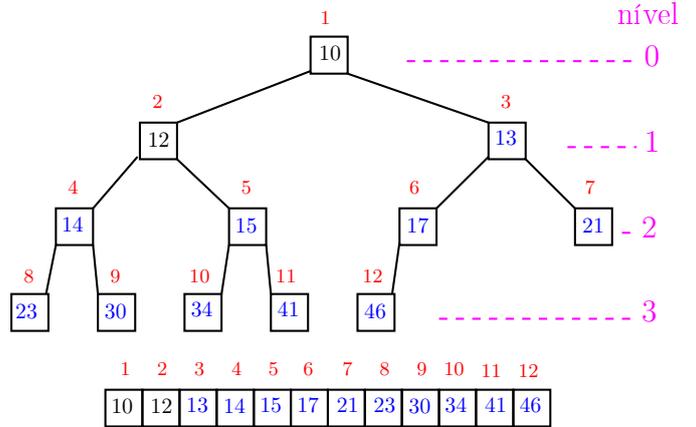
Navigation icons

Heapsort



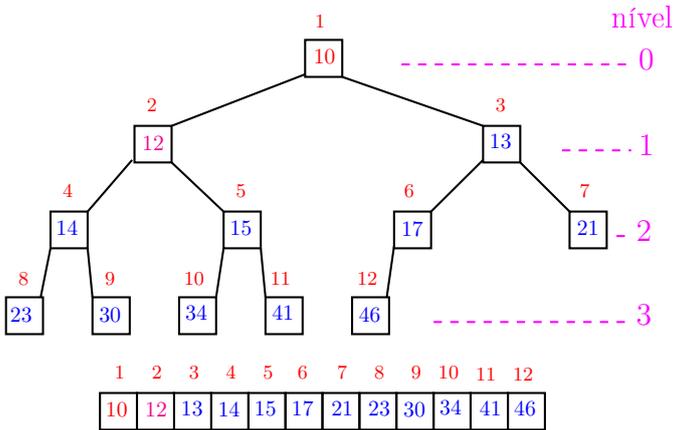
Navigation icons

Heapsort



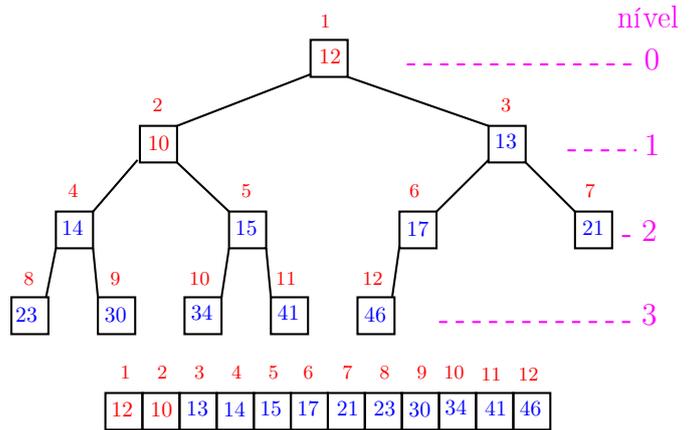
Navigation icons

Heapsort



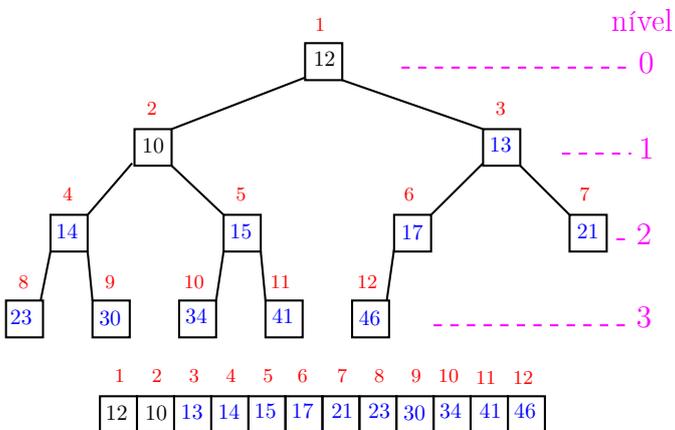
Navigation icons

Heapsort



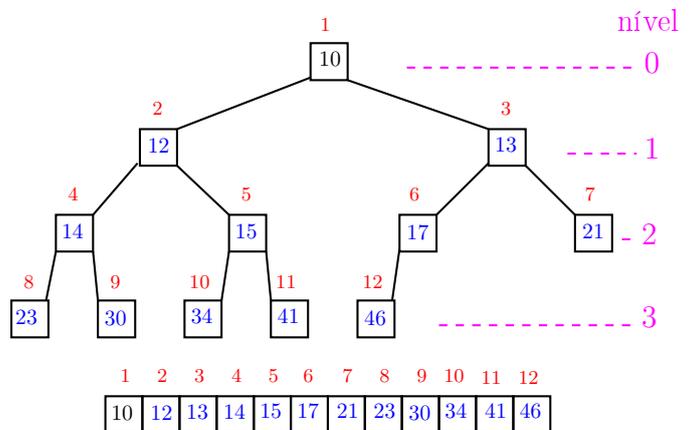
Navigation icons

Heapsort



Navigation icons

Heapsort



Navigation icons

Conclusão

O consumo de tempo da função `heapSort` é proporcional a $n \lg n$.

O consumo de tempo da função `heapSort` é $O(n \lg n)$.

Mais análise experimental

Algoritmos implementados:

`mergeR` `mergeSort` recursivo.

`mergeI` `mergeSort` iterativo.

`quick` `quickSort` recursivo.

`heap` `heapSort`.

Mais análise experimental

A plataforma utilizada nos experimentos foi um computador rodando Ubuntu GNU/Linux 3.5.0-17

Compilador:

```
gcc -Wall -ansi -O2 -pedantic  
-Wno-unused-result.
```

Computador:

```
model name: Intel(R) Core(TM)2 Quad CPU Q6600 @  
2.40GHz  
cpu MHz : 1596.000  
cache size: 4096 KB  
MemTotal : 3354708 kB
```

Aleatório: média de 10

n	mergeR	mergeI	quick	heap
8192	0.00	0.00	0.00	0.00
16384	0.00	0.00	0.00	0.00
32768	0.01	0.01	0.01	0.00
65536	0.01	0.01	0.01	0.01
131072	0.02	0.02	0.02	0.03
262144	0.05	0.04	0.04	0.06
524288	0.10	0.08	0.08	0.12
1048576	0.21	0.20	0.17	0.28
2097152	0.44	0.43	0.35	0.70
4194304	0.92	0.90	0.73	1.73
8388608	1.90	1.87	1.51	4.13

Tempos em segundos.

Decrescente

n	mergeR	mergeI	quick	heap
1024	0.00	0.00	0.00	0.00
2048	0.00	0.00	0.00	0.00
4096	0.01	0.00	0.01	0.00
8192	0.00	0.00	0.03	0.00
16384	0.00	0.00	0.14	0.00
32768	0.00	0.01	0.57	0.00
65536	0.01	0.01	2.27	0.01
131072	0.02	0.01	9.06	0.02
262144	0.03	0.03	36.31	0.04

Tempos em segundos.

Para $n=524288$ `quickSort` dá `Segmentation fault (core dumped)`

Crescente

n	mergeR	mergeI	quick	heap
1024	0.00	0.00	0.00	0.00
2048	0.00	0.00	0.00	0.00
4096	0.00	0.00	0.00	0.00
8192	0.00	0.00	0.03	0.00
16384	0.00	0.00	0.14	0.01
32768	0.01	0.00	0.57	0.01
65536	0.00	0.01	2.26	0.01
131072	0.02	0.02	9.05	0.02
262144	0.03	0.02	36.21	0.04

Tempos em segundos.

Para $n=524288$ `quickSort` dá `Segmentation fault (core dumped)`

