

Análise do Heap Binário e HeapSort

André Vignatti

Análise de Pior Caso

Teorema. *`build-max-heap` faz no máximo $4n$ comparações entre elem. do vetor.*

Demonstração. A quantidade de nós no nível i é 2^i e, para cada nó, a execução de `max-heapify` custa $2(h - i)$ no pior caso.

Somando todos os níveis:

$$\begin{aligned} C(n) &= 1 \cdot 2h + 2 \cdot 2(h - 1) + 4 \cdot 2(h - 2) + \dots + 2^{h-1} \cdot 2 \cdot 1 + 2^h \cdot 2 \cdot 0 \\ &= 2(1 \cdot h + 2 \cdot (h - 1) + 4 \cdot (h - 2) + \dots + 2^{h-1} \cdot 2 \cdot 1 + 2^h \cdot 2 \cdot 0) \\ &= 2 \sum_{i=0}^h 2^i \cdot (h - i) \\ &= 2 \left(h \sum_{i=0}^h 2^i - \sum_{i=0}^h 2^i \cdot i \right). \end{aligned}$$

De aulas passadas, sabemos que

$$\sum_{i=0}^h 2^i = 2^{h+1} - 1 \quad \text{e} \quad \sum_{i=0}^h 2^i \cdot i = h2^{h+2} - (h+1)2^{h+1} + 2.$$

Assim, substituindo,

$$\begin{aligned} C(n) &= 2(h(2^{h+1} - 1) - (h2^{h+2} - (h+1)2^{h+1} + 2)) \\ &= 2(h2^{h+1} - h - h2^{h+2} + h2^{h+1} + 2^{h+1} - 2) \\ &= 2(h2^{h+2} - h2^{h+2} - h + 2^{h+1} - 2) \\ &\leq 2^{h+2} \\ &= 4 \cdot 2^h \end{aligned}$$

Lembrando (aulas passadas), $h = \lfloor \log_2 n \rfloor$. Assim,

$$C(n) = 4 \cdot 2^{\lfloor \log_2 n \rfloor} \leq 4 \cdot 2^{\log_2 n} = 4n.$$

□

Heapsort

O algoritmo Heapsort serve para ordenar um vetor.

```
heapsort( $v, n$ )
```

```
  build-max-heap( $v, n$ )
  para  $i \leftarrow n$  até 2 faça
    troca( $v[1], v[i]$ )
    max-heapify( $v, 1, i$ )
```

Exemplo

	1	2	3	4	5	6	7	8	9	10
v	15	35	40	10	20	5	45	70	25	90

Após build-max-heap, obtemos um *heap* de máximo:

	1	2	3	4	5	6	7	8	9	10
v	90	70	45	25	35	5	40	10	15	20

(executar no quadro a partir daqui)

Análise de Pior Caso

Em vetor de tamanho i (árvore binária quase completa), a maior altura é $\lceil \log_2 i \rceil$.

Portanto, **max-heapify** em vetor de tamanho i custa $2\lceil \log_2 i \rceil$ no pior caso.

Somando todas iterações,

$$\sum_{i=2}^n 2\lceil \log_2 i \rceil \leq \sum_{i=1}^n 2\log_2 n = 2n \log_2 n.$$

Finalmente, somando com o custo do **build-max-heap**

$$C(n) \leq 2n \log_2 n + 4n.$$