"Estou fazendo um sistema operacional gratuito (apenas um hobby, não será grande e profissional como GNU) para 386/486 AT." (Linus Torvalds referindo-se ao Linux; 1991).

Construindo uma Heap

Paulo Ricardo Lisboa de Almeida

DSBD

UFPR
UNIVERSIDADE FEDERAL DO PARANÁ

Da aula passada

O que o max-heapify faz?

```
função max-heapify(h,i,n)
l ← esquerda(i)
r ← direita(i)
se l ≤ n e h[l] > h[i]
      maior ← l
senão
      maior ← i
se r ≤ n e h[r] > h[maior]
      maior ← r
se maior ≠ i
      trocar(h,i,maior)
      max-heapify(h,maior,n)
```

Da aula passada

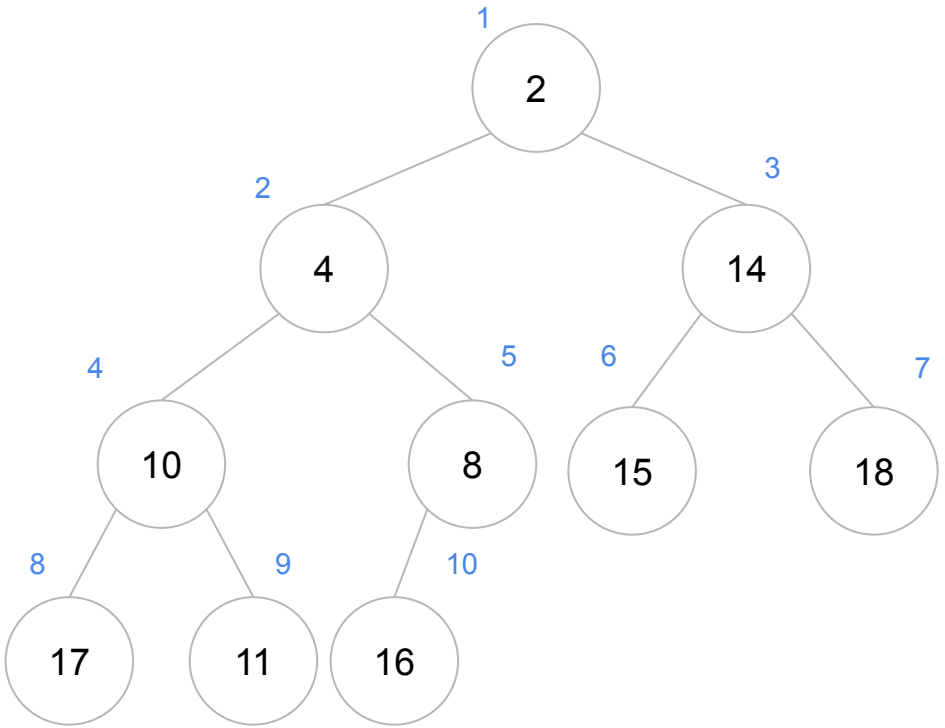
O que o max-heapify faz?

função max-heapify(h,i,n)
entrada: um vetor h indexado por h[1..n] e um valor i ∈ [1..n]. A árvore esquerda(i) e direita(i) são max-heaps. h[i] pode ser menor que seus filhos.
saída: a subárvore com raiz em i é modificada de forma que a propriedade da max-heap seja satisfeita

```
l ← esquerda(i)
r ← direita(i)
se l ≤ n e h[l] > h[i]
      maior ← l
senão
      maior ← i
se r ≤ n e h[r] > h[maior]
      maior ← r
se maior ≠ i
      trocar(h,i,maior)
      max-heapify(h,maior,n)
```

Pergunta

Como usar o max-heapify para criar uma max-heap?

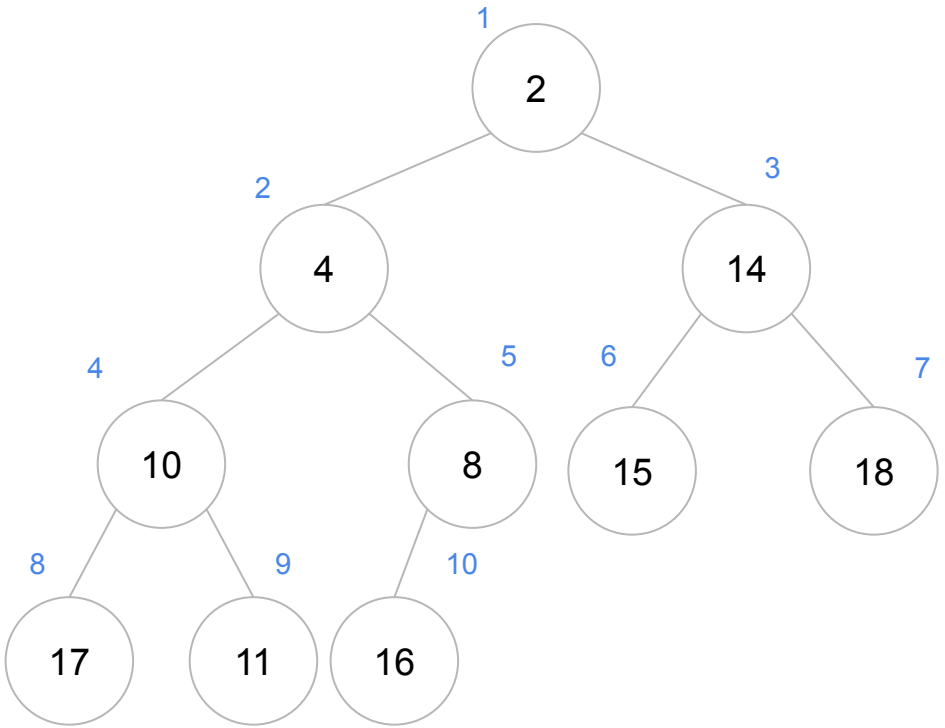


1	2	3	4	5	6	7	8	9	10
2	4	14	10	8	15	18	17	11	16

Pergunta

Como usar o max-heapify para criar uma max-heap?

As folhas de uma heap ficam em $h[\lfloor n/2 \rfloor + 1..n]$



1	2	3	4	5	6	7	8	9	10
2	4	14	10	8	**15**	**18**	**17**	**11**	**16**

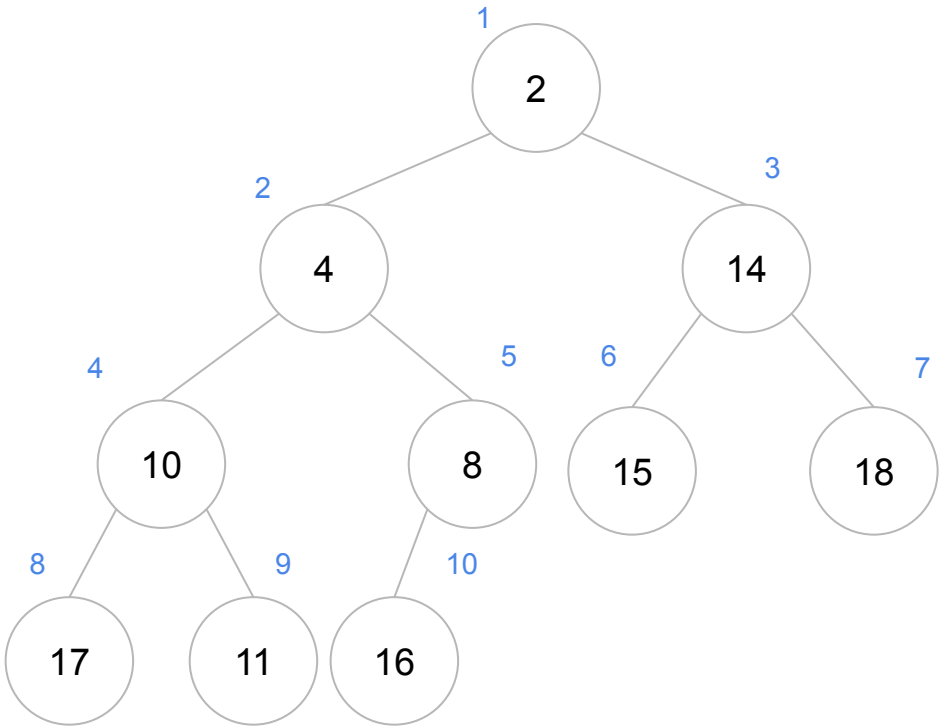
Pergunta

Como usar o max-heapify para criar uma max-heap?

As folhas de uma heap ficam em $h[\lfloor n/2 \rfloor + 1..n]$

Toda subárvore que começa em uma folha satisfaz a propriedade da max-heap

Basta adicionar novos nós na heap “de trás para frente” utilizando o max-heapify.

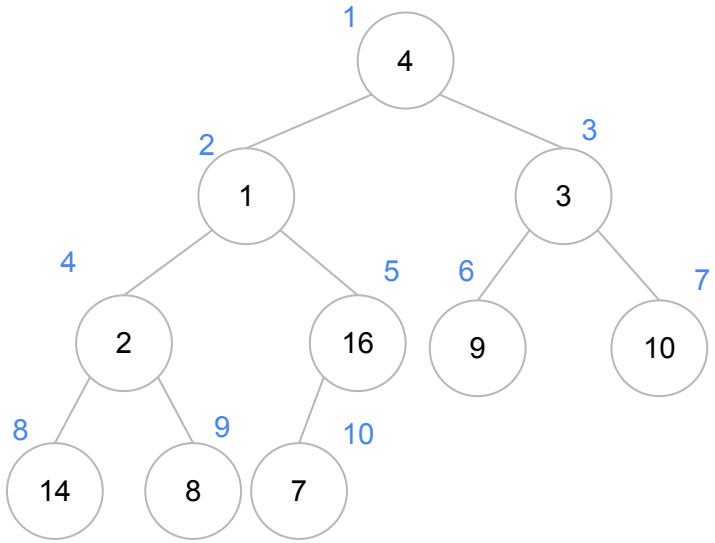


1	2	3	4	5	6	7	8	9	10
2	4	14	10	8	**15**	**18**	**17**	**11**	**16**

Max-Heap

função construir-max-heap(v,n)
entrada: um vetor v indexado por v[1..n].
saída: o vetor modificado de forma que ele seja uma max-heap
para i = ⌊n/2⌋ até 1 passo -1
    max-heapify(v,i,n)

Teste de Mesa

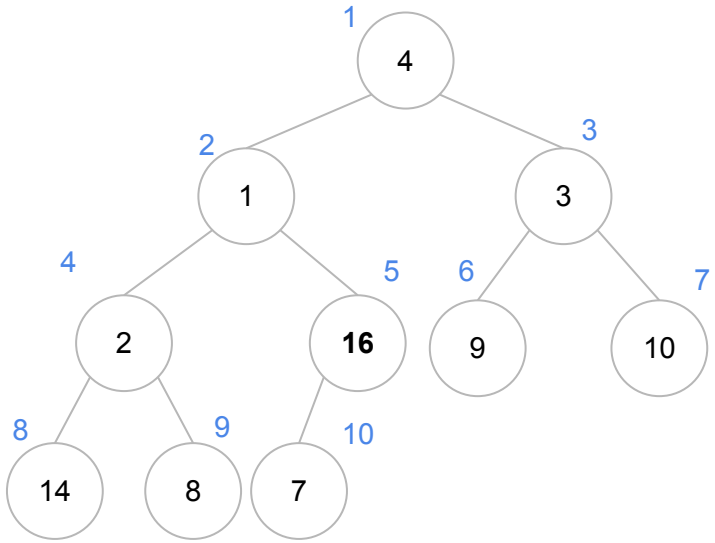


1	2	3	4	5	6	7	8	9	10
4	1	3	2	16	9	10	14	8	7

função construir-max-heap(v,n)
para i =⌊n/2⌋até 1 passo -1
    max-heapify(v,i,n)

Teste de Mesa

n	i
10	5



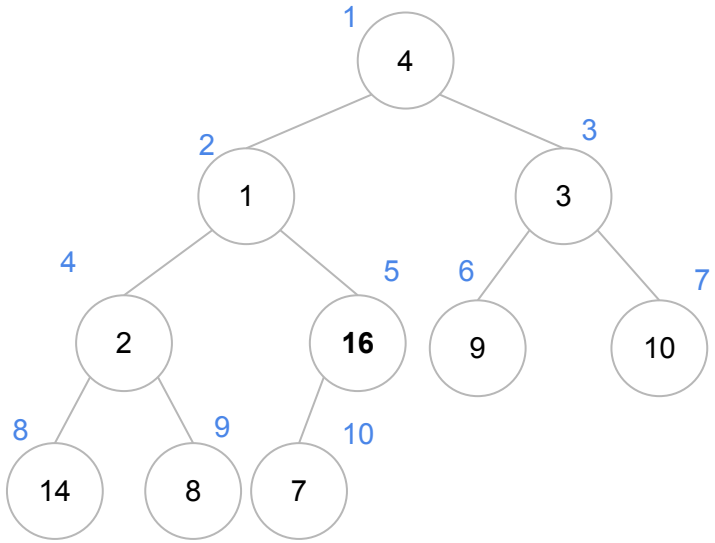
1	2	3	4	5	6	7	8	9	10
4	1	3	2	**16**	9	10	14	8	7

função construir-max-heap(v,n)

```
para i =⌊n/2⌋até 1 passo -1
    max-heapify(v,i,n)
```

Teste de Mesa

n	i
10	5

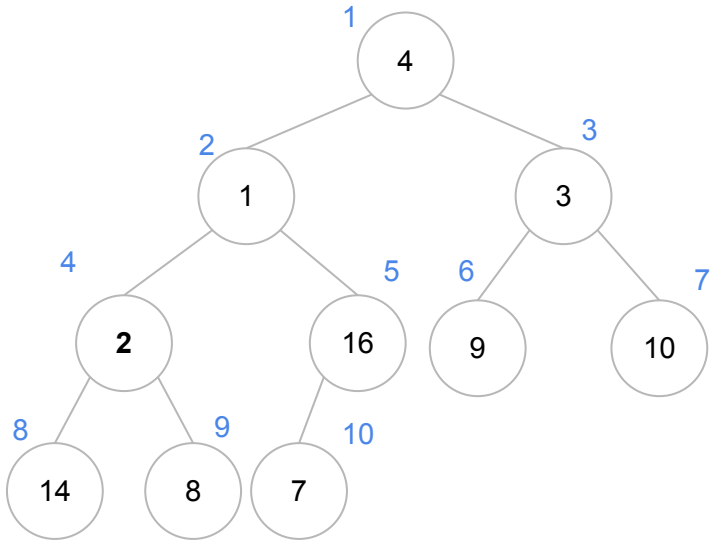


1	2	3	4	5	6	7	8	9	10
4	1	3	2	**16**	9	10	14	8	7

```
função construir-max-heap(v,n)
para i =⌊n/2⌋até 1 passo -1
    max-heapify(v,i,n)
```

Teste de Mesa

n	i
10	5
	4



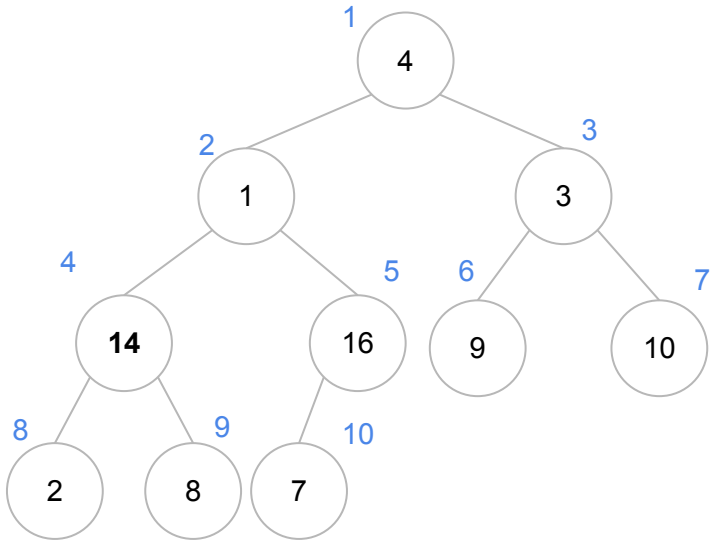
1	2	3	4	5	6	7	8	9	10
4	1	3	**2**	16	9	10	14	8	7

função construir-max-heap(v,n)

```
para i =⌊n/2⌋até 1 passo -1
    max-heapify(v,i,n)
```

Teste de Mesa

n	i
10	5
	4

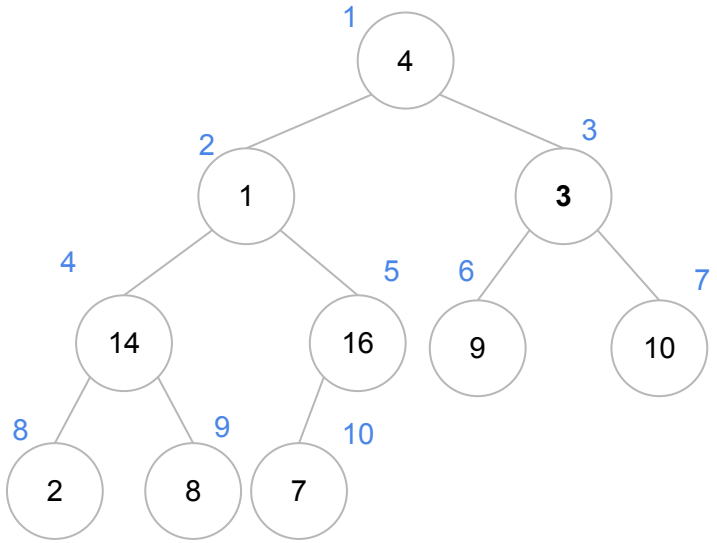


1	2	3	4	5	6	7	8	9	10
4	1	3	**14**	16	9	10	2	8	7

```
função construir-max-heap(v,n)
para i =⌊n/2⌋até 1 passo -1
    max-heapify(v,i,n)
```

Teste de Mesa

n	i
10	5
	4
	3



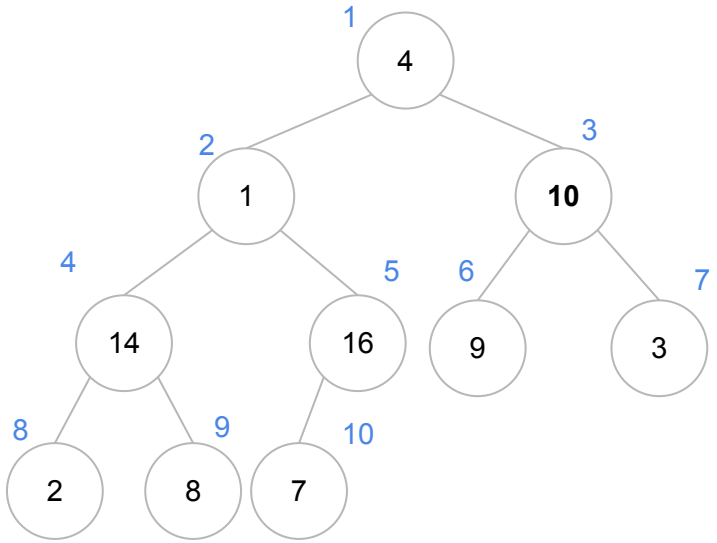
1	2	3	4	5	6	7	8	9	10
4	1	**3**	14	16	9	10	2	8	7

função construir-max-heap(v,n)

```
para i =⌊n/2⌋até 1 passo -1
    max-heapify(v,i,n)
```

Teste de Mesa

n	i
10	5
	4
	3

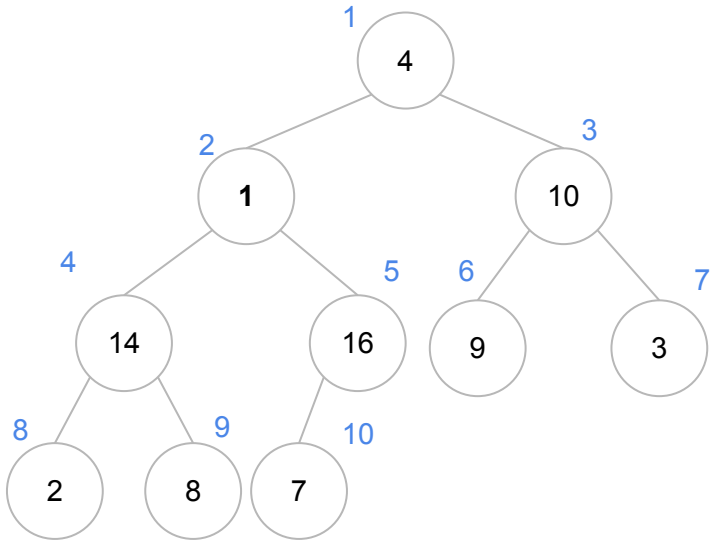


1	2	3	4	5	6	7	8	9	10
4	1	**10**	14	16	9	3	2	8	7

```
função construir-max-heap(v,n)
para i =⌊n/2⌋até 1 passo -1
    max-heapify(v,i,n)
```

Teste de Mesa

n	i
10	5
	4
	3
	2

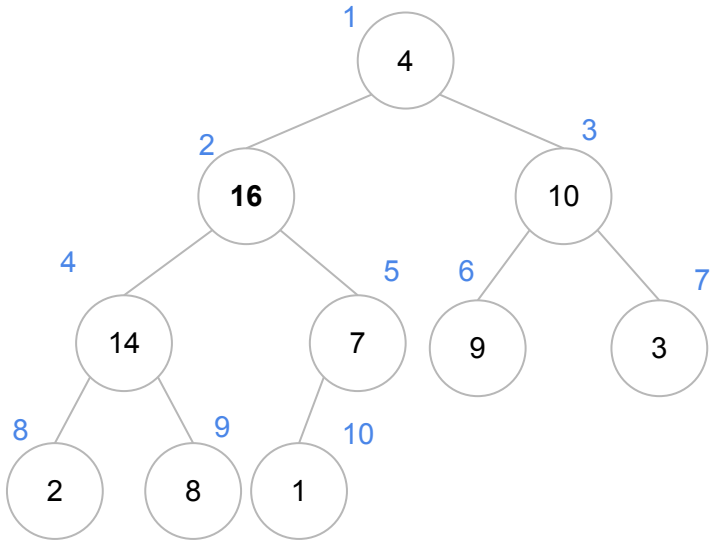


1	2	3	4	5	6	7	8	9	10
4	**1**	10	14	16	9	3	2	8	7

função construir-max-heap(v,n)
para i =⌊n/2⌋até 1 passo -1
max-heapify(v,i,n)

Teste de Mesa

n	i
10	5
	4
	3
	2

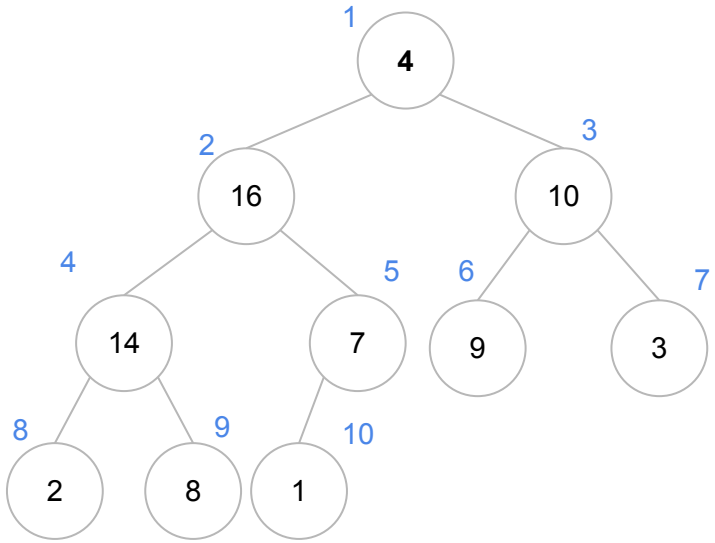


1	2	3	4	5	6	7	8	9	10
4	**16**	10	14	7	9	3	2	8	1

```
função construir-max-heap(v,n)
para i =⌊n/2⌋até 1 passo -1
    max-heapify(v,i,n)
```

Teste de Mesa

n	i
10	5
	4
	3
	2
	1



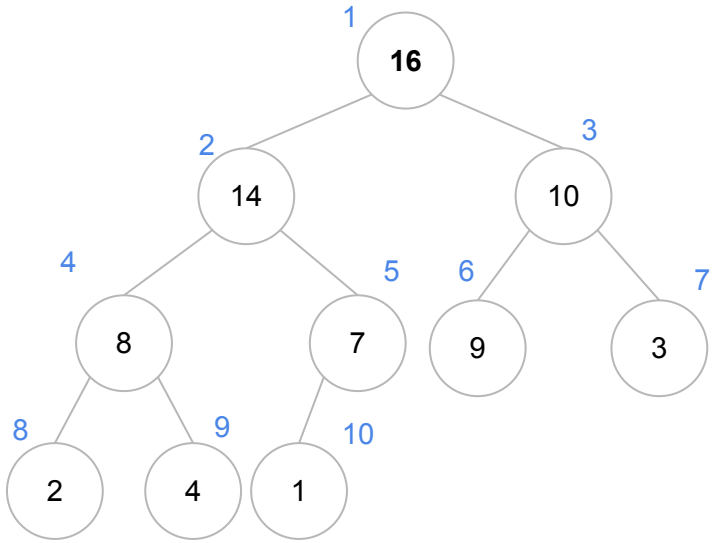
1	2	3	4	5	6	7	8	9	10
4	16	10	14	7	9	3	2	8	1

função construir-max-heap(v,n)

```
para i =⌊n/2⌋até 1 passo -1
    max-heapify(v,i,n)
```

Teste de Mesa

n	i
10	5
	4
	3
	2
	1



1	2	3	4	5	6	7	8	9	10
16	14	10	8	7	9	3	2	4	1

```
função construir-max-heap(v,n)
para i =⌊n/2⌋até 1 passo -1
    max-heapify(v,i,n)
```

Análise

O loop é executado exatamente $\lfloor n/2 \rfloor$ vezes

```
função construir-max-heap(v,n)
para i =⌊n/2⌋até 1 passo -1
     max-heapify(v,i,n)
```

Análise

Considerando o número de comparações entre elementos do vetor

O loop é executado exatamente $\lfloor n/2 \rfloor$ vezes

Cada execução chama max-heapify, que da aula passada, custa no pior caso

$$C^{+}_{mh}(n) = 2\lfloor \log_2 n \rfloor$$

Dessa forma, o pior caso

$$C^{+}(n) = n\lfloor \log_2 n \rfloor$$

```
função construir-max-heap(v,n)
para i =⌊n/2⌋até 1 passo -1
     max-heapify(v,i,n)
```

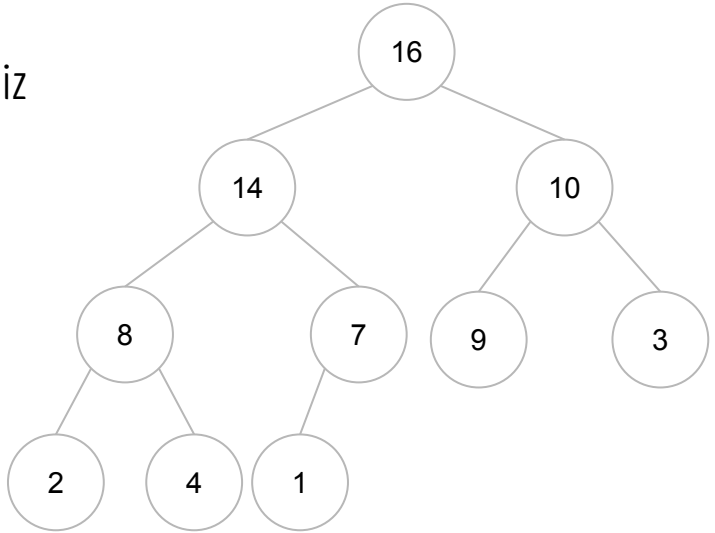
Análise

Mas esse não é o pior caso do algoritmo

Para perceber isso, basta notar que o custo de max-heapify depende da altura da árvore

- O pior caso ocorre apenas quando a chamada envolve a raiz
- Mas a quantidade de nós a partir de onde vamos chamar o algoritmo decresce exponencialmente conforme chegamos mais próximos da raiz

Logo, **apenas uma chamada de max-heapify** vai custar o pior caso




```
função construir-max-heap(v,n)
para i =⌊n/2⌋até 1 passo -1
     max-heapify(v,i,n)
```

Análise

Ao levar isso em consideração, vamos chegar a conclusão que o custo do algoritmo é

$$C^{+}(n) \approx n$$

O custo de se construir uma heap é **linearmente proporcional ao tamanho do vetor**

```
função construir-max-heap(v,n)
para i =⌊n/2⌋até 1 passo -1
     max-heapify(v,i,n)
```

Análise

Ao levar isso em consideração, vamos chegar a conclusão que o custo do algoritmo é

$$C^{+}(n) \approx n$$

A prova não é trivial e não será cobrada na disciplina

Tema para Análise de Algoritmos

Você pode encontrar a prova em Cormen et al, 2012

```
função construir-max-heap(v,n)
para i =⌊n/2⌋até 1 passo -1
     max-heapify(v,i,n)
```

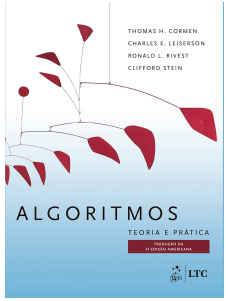
Exercícios

1. Implemente o *construir-max-heap* em C.

Referências

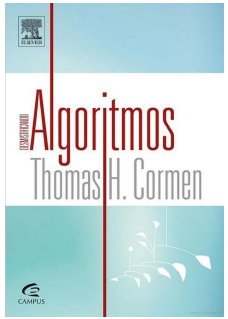
Renato Carmo. Algoritmos e Estruturas de Dados. www.inf.ufpr.br/renato

T. Cormen, C. Leiserson, R. Rivest, C. Stein. Algoritmos: Teoria e Prática. 3a ed. 2012

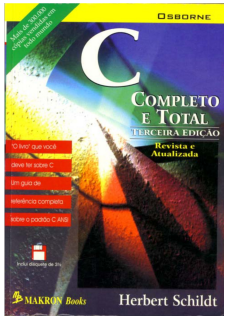


R. Sedgewick, K. Wayne. Algorithms Part I. 4a ed. 2014

T. Cormen. Desmistificando algoritmos. 2017.



H. Schildt. C completo e total. 1996



Licença