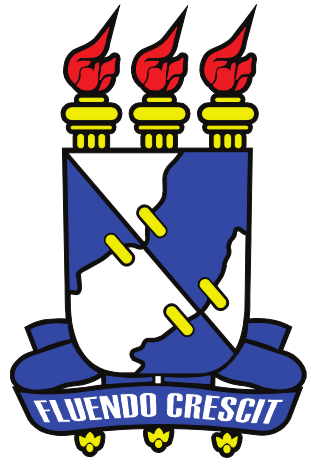


Estrutura de Dados I

Heap

Bruno Prado

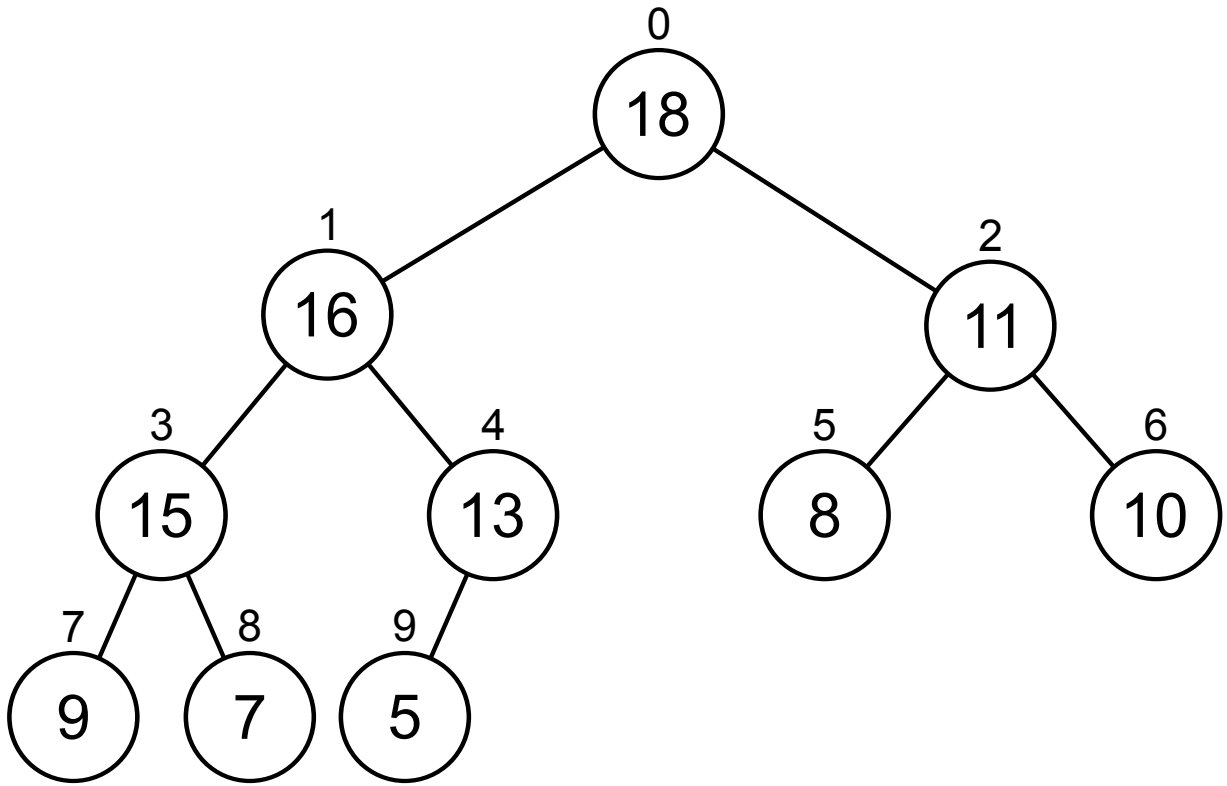
Departamento de Computação / UFS

Introdução

- O que é uma árvore heap?
  - Árvore binária de prioridade
  - Representação implícita em vetor
  - Indexação dos nós

Introdução

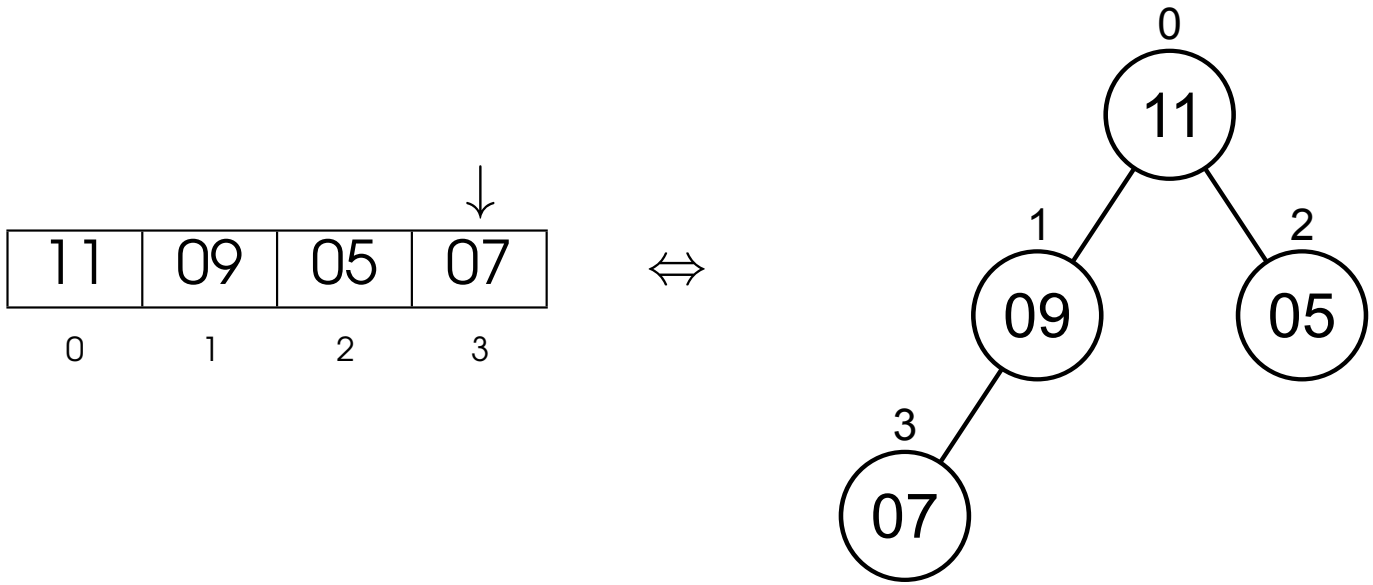
- Árvore binária heap



18	16	11	15	13	8	10	9	7	5
0	1	2	3	4	5	6	7	8	9

Heap

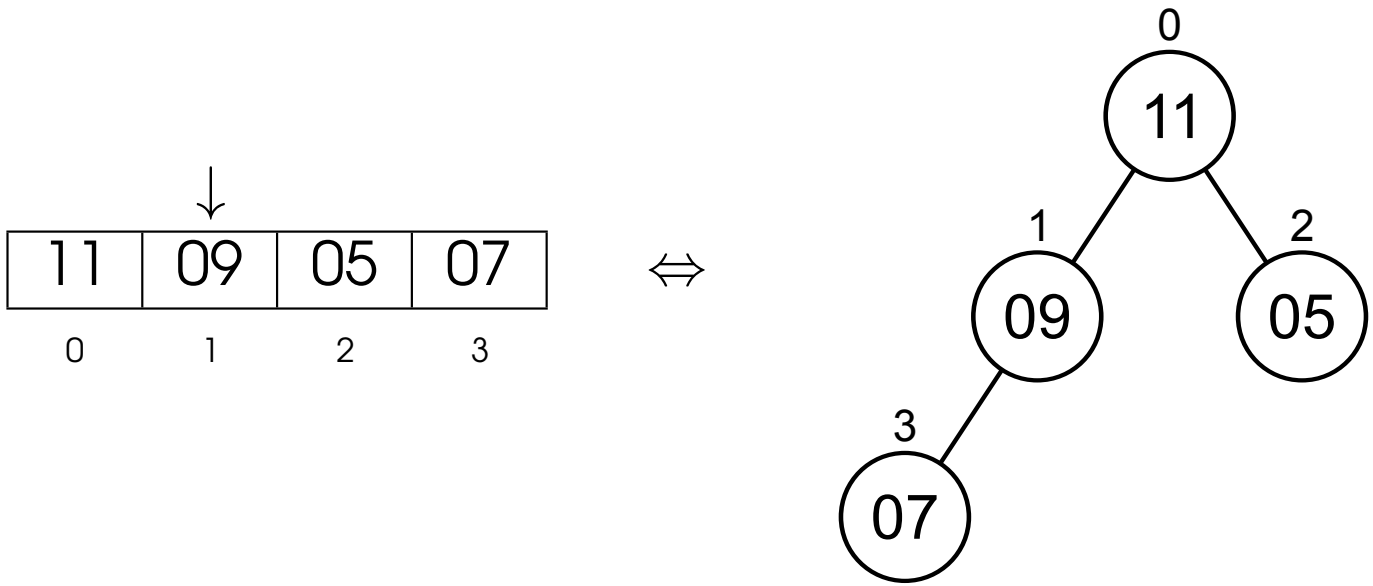
- Representação e indexação
  - Nó pai



$$Pai(i) = \frac{i-1}{2}$$

Heap

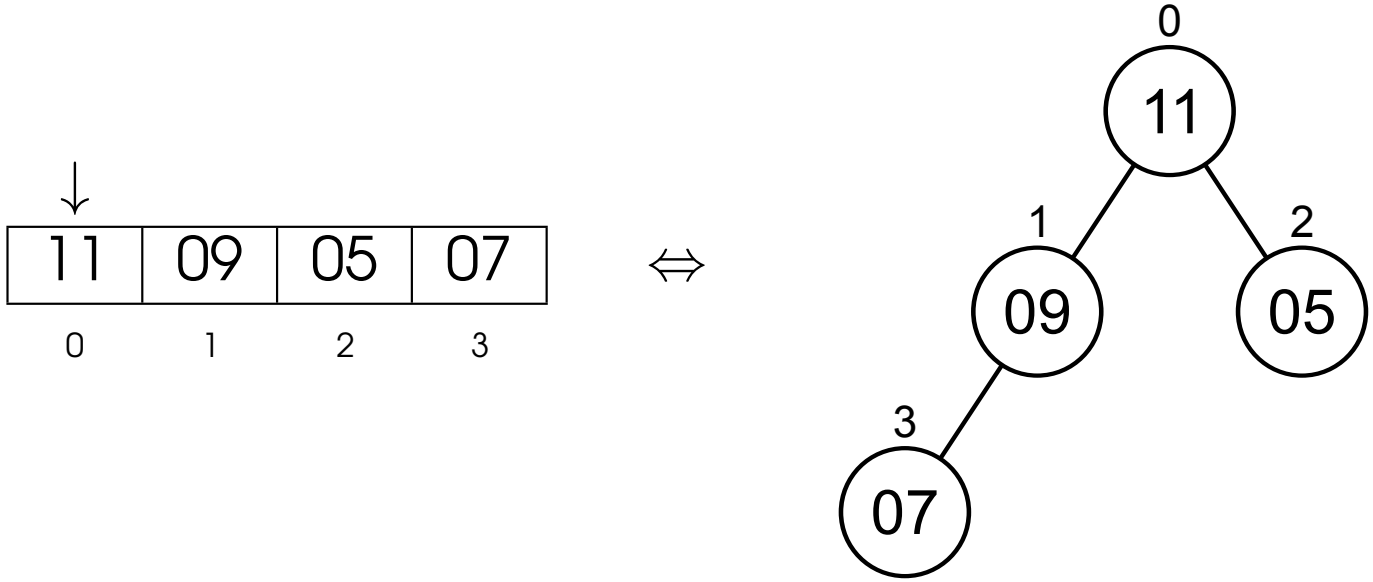
- Representação e indexação
  - Nó filho esquerdo



$$Esquerdo(i) = 2i + 1$$

Heap

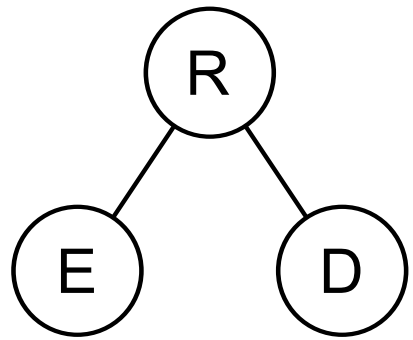
- Representação e indexação
  - Nó filho direito



$$Direito(i) = 2i + 2$$

Heap

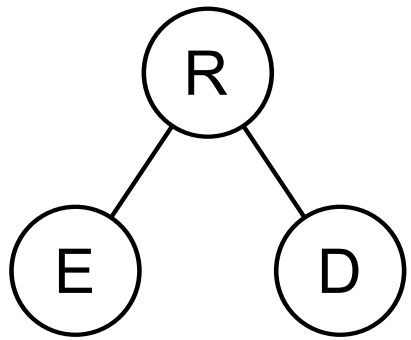
- Tipos de árvores heap
  - Heap mínimo



Propriedade $R \leq E$ e $R \leq D$

Heap

- Tipos de árvores heap
  - Heap máximo



Propriedade $R \geq E$ e $R \geq D$

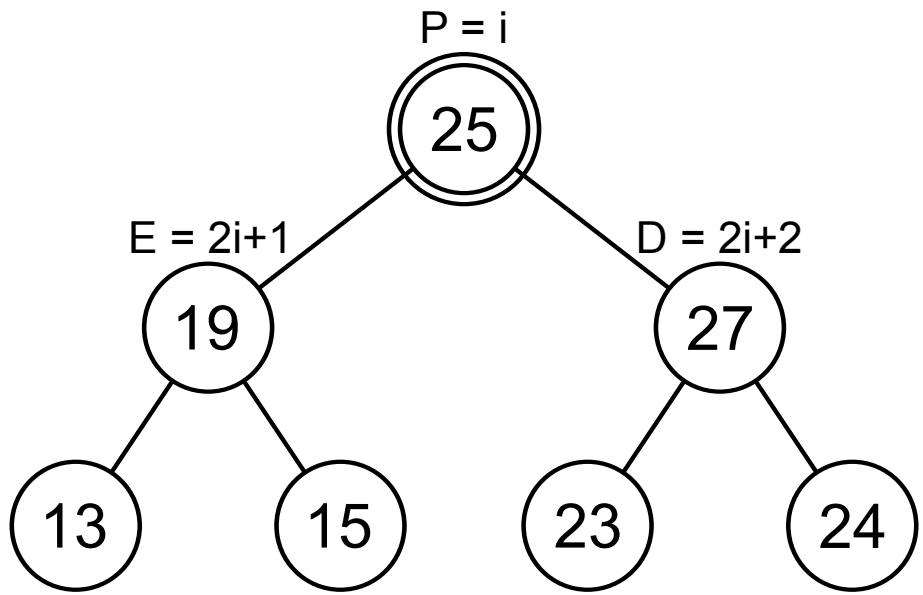
Heap

- Aplicação da propriedade de heap

```
// Procedimento heapify
void heapify(int* V, unsigned int T, unsigned int i) {
        unsigned int P = i;
        unsigned int E = esquerdo(i);
        unsigned int D = direito(i);
        if(E < T && V(E) >V(P)) P = E;
        if(D < T && V(D) >V(P)) P = D;
        if(P != i) {
                troca(V, P, i);
                heapify(V, T, P);
        }
}
```

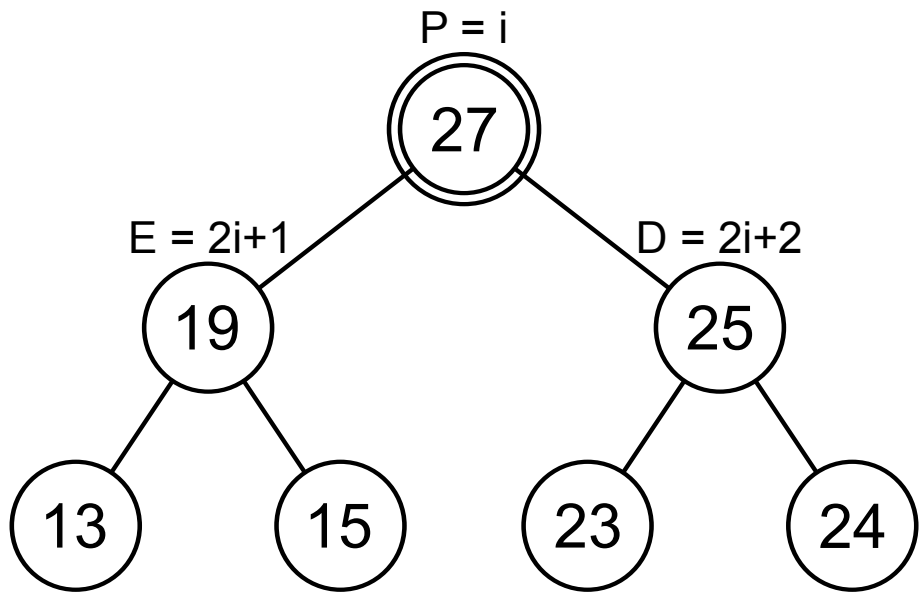
Heap

- Aplicação da propriedade de heap
  - Procedimento heapify na raiz



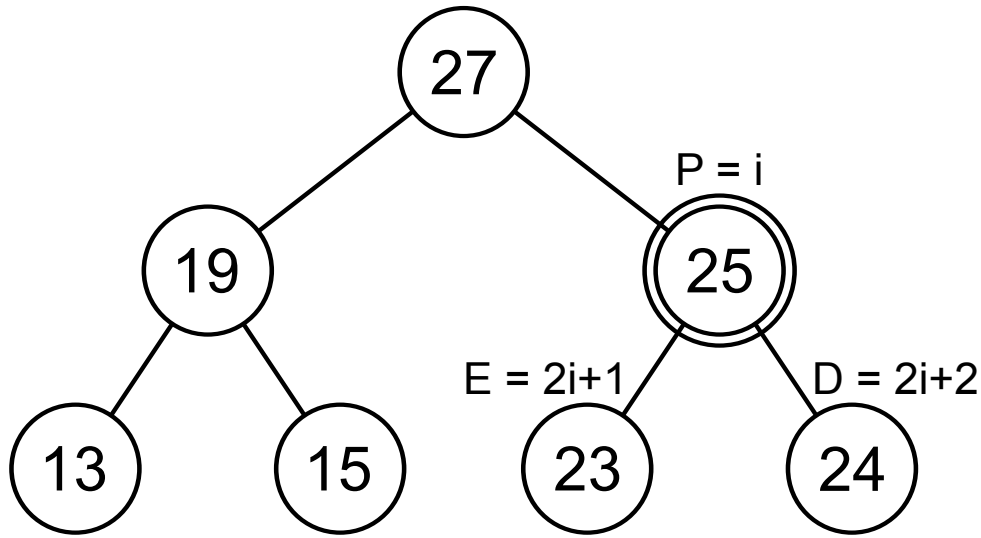
Heap

- Aplicação da propriedade de heap
  - Procedimento heapify na raiz



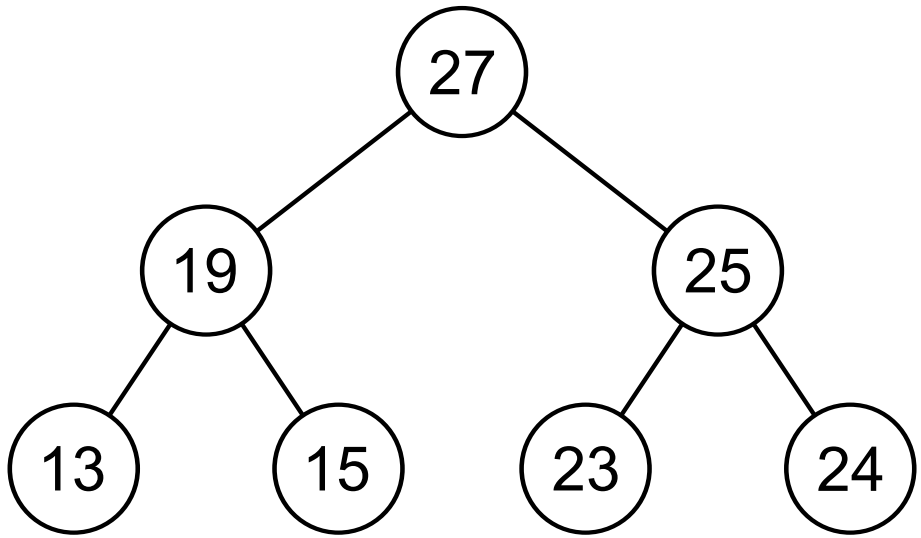
Heap

- Aplicação da propriedade de heap
  - Procedimento heapify no filho da direita



Heap

- Aplicação da propriedade de heap
  - Procedimento heapify finalizada

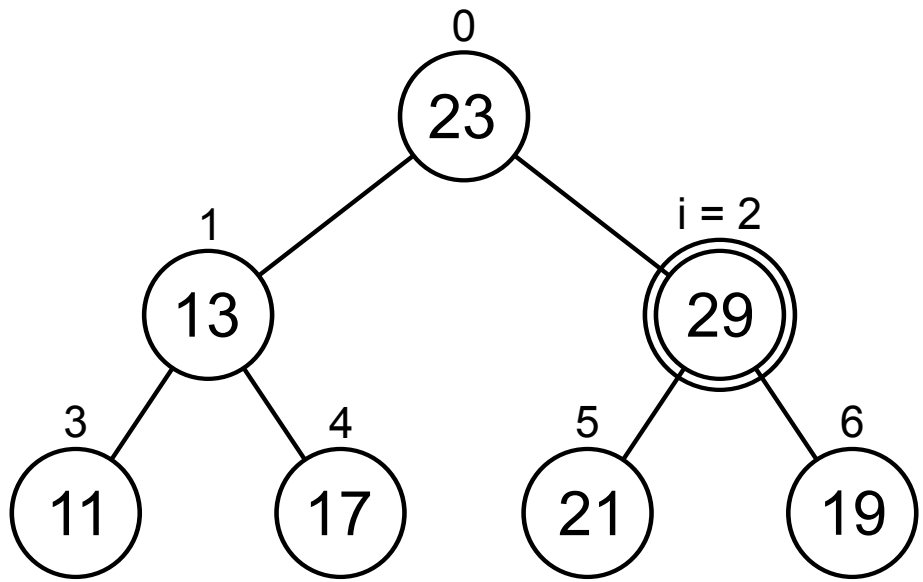


Heap

- Complexidade do procedimento heapify
  - A altura h da árvore é $\log_2 n$
  - Custo das operações $O(h)$

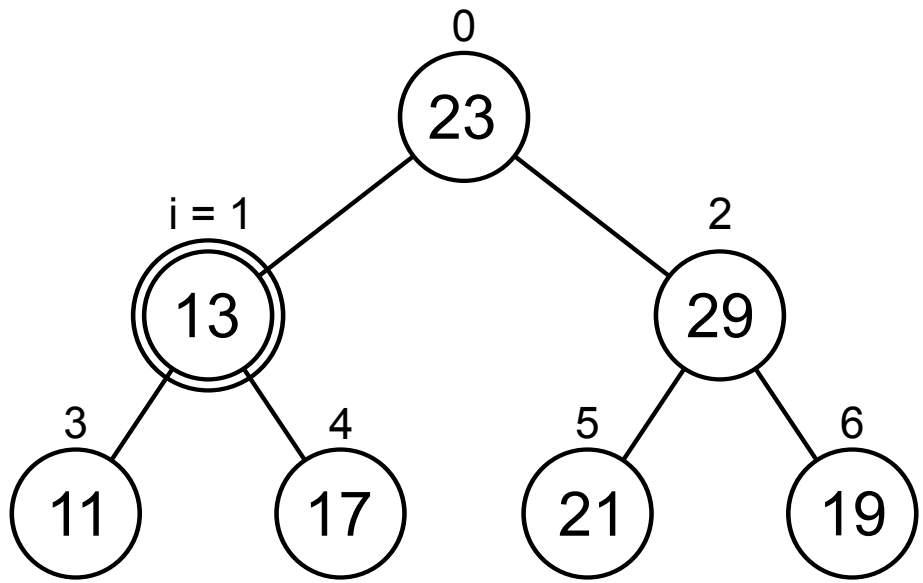
Heap

- Construção da árvore heap
  - Começa pelo último nó com filhos
  - Heapify no índice $i = \frac{Tamanho-1}{2}$



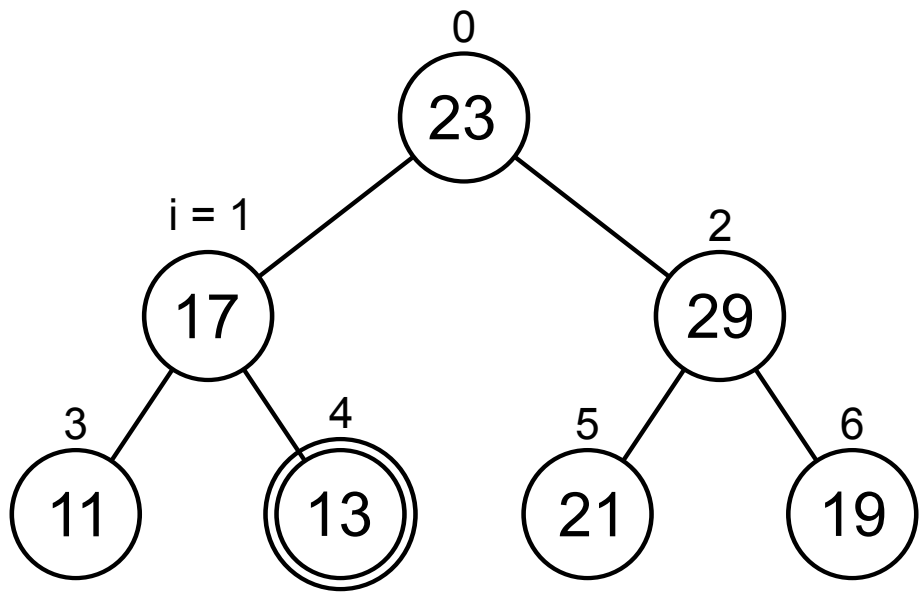
Heap

- Construção da árvore heap
  - O índice é decrementado até atingir a raiz
  - Heapify no índice $i = \left(\frac{Tamanho-1}{2}\right) - 1$



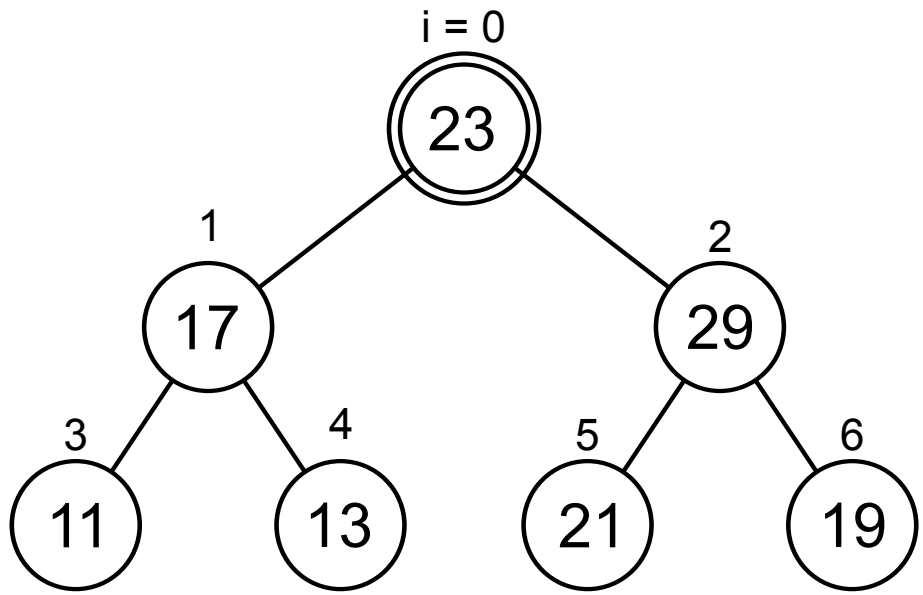
Heap

- Construção da árvore heap
  - O índice é decrementado até atingir a raiz
  - Heapify no índice $i = \left(\frac{Tamanho-1}{2}\right) - 1$



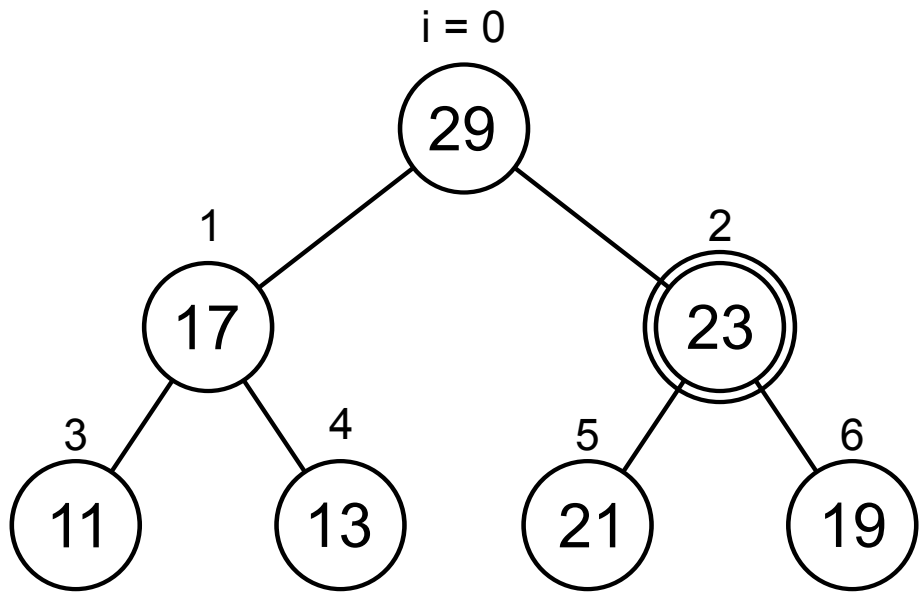
Heap

- Construção da árvore heap
  - O índice é decrementado até atingir a raiz
  - Heapify no índice $i = \left(\frac{Tamanho-1}{2}\right) - 2$



Heap

- Construção da árvore heap
  - O índice é decrementado até atingir a raiz
  - Heapify no índice $i = \left(\frac{Tamanho-1}{2}\right) - 2$

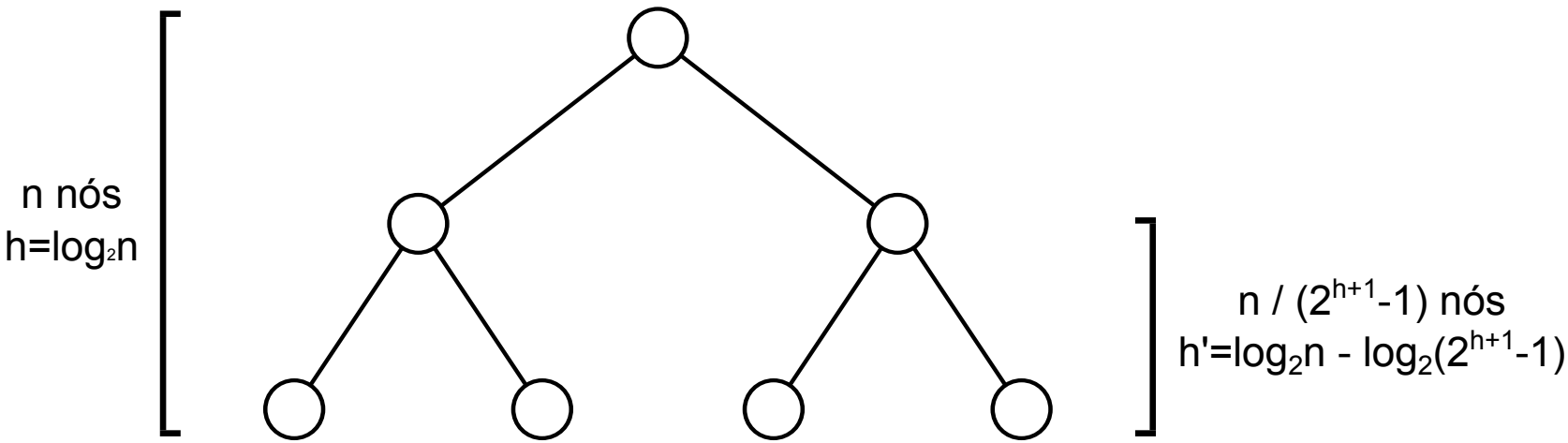


Heap

- Análise de complexidade
  - São feitas $\frac{n}{2}$ iterações do heapify que custa $O(h)$ para construção da árvore heap
  - Custo total é $\left(\frac{n}{2}\right) \times O(h) = O(\frac{n}{2} \times h) = O(n \log_2 n)$

Heap

- Análise de complexidade
  - No nível i existem até 2^i nós
  - Temos máximo de $\sum_{i=0}^{h} 2^i = 2^{h+1} - 1$ nós
  - Acima no nível h temos altura $h' = \log_2 n - \log_2(2^{h+1} - 1)$ e até $\frac{n}{2^{h+1}-1}$ nós



Heap

- Análise de complexidade
  - O tempo de execução do heapify está limitada a altura h' que possui até $\frac{n}{2^{h+1}-1}$ nós

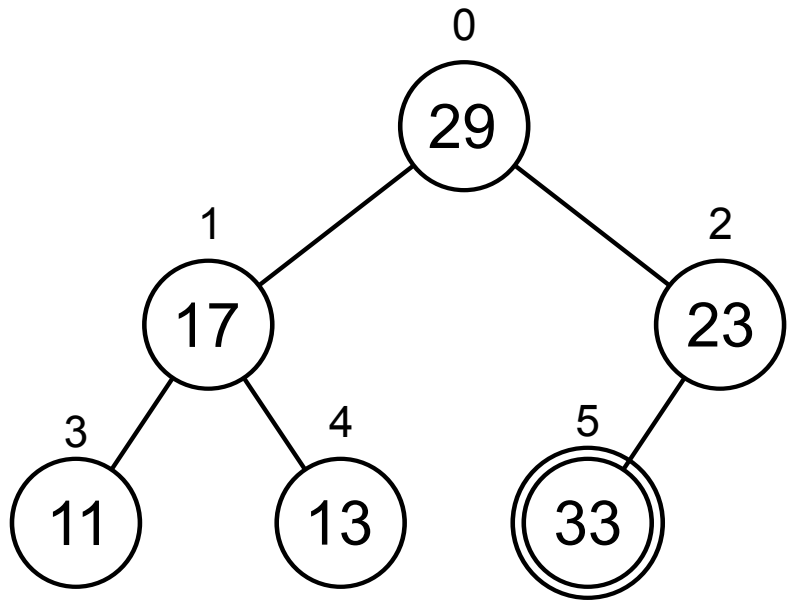
$$
\begin{aligned}
\textit{construir\_heap}(n) &= O\left(\sum_{h=0}^{\log_2 n} \frac{n}{2^{h+1}-1} \times h\right) \\
&= O\left(n \times \sum_{h=0}^{\log_2 n} \frac{h}{2^h}\right) \\
&= O\left(n \times \sum_{h=0}^{\infty} \frac{h}{2^h}\right) \\
&= O(n)
\end{aligned}
$$

Heap

- Análise de complexidade
  - Espaço $O(1)$
  - Tempo $O(n)$

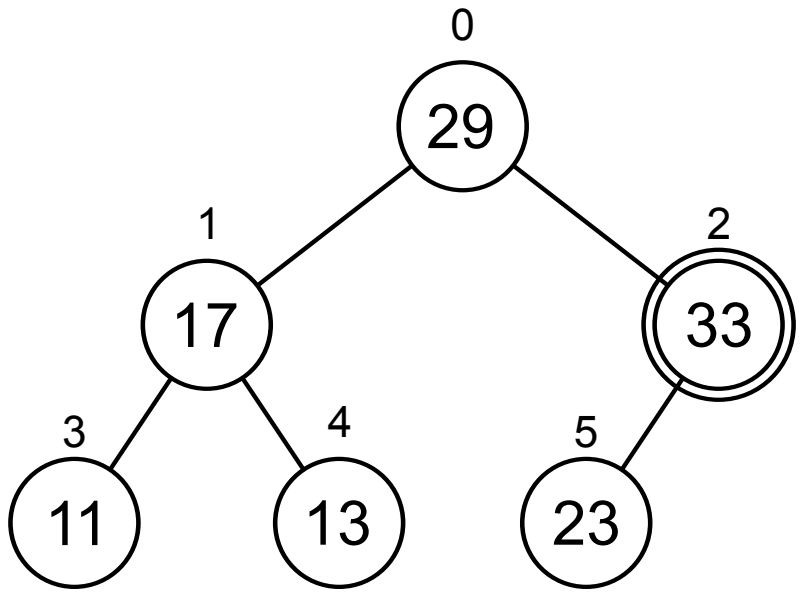
Heap

- Inserção na árvore heap
  - É feita a inserção do número 33 no final do vetor
  - Para garantir a propriedade de heap, é aplicado o procedimento heapify no pai



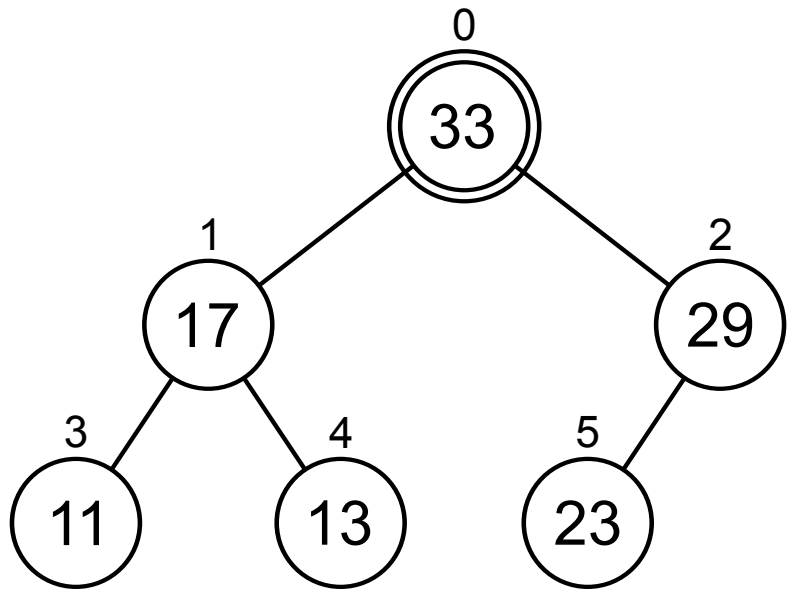
Heap

- Inserção na árvore heap
  - Como o número inserido é maior do que o pai, é feita a troca de posições
  - O procedimento heapify é aplicado ao pai do nível



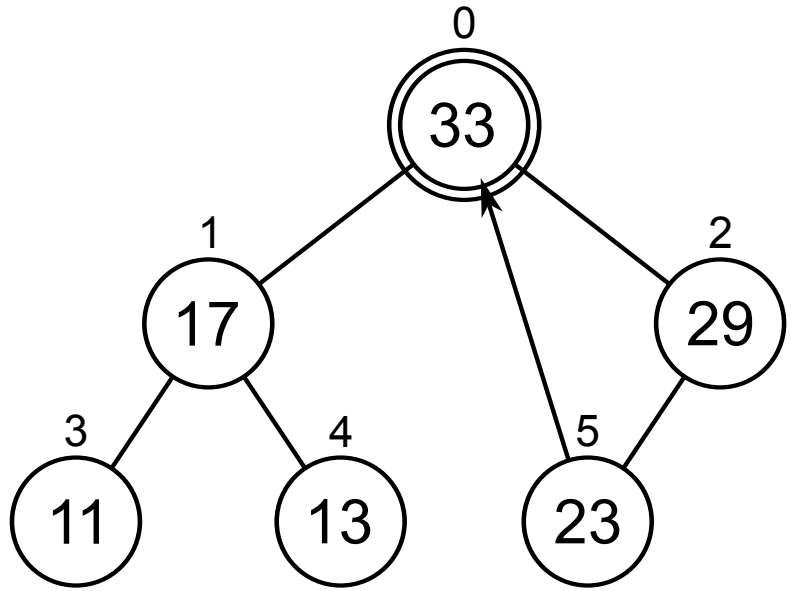
Heap

- Inserção na árvore heap
  - Como o número inserido é maior do que o pai, é feita a troca de posições
  - O procedimento heapify é aplicado ao pai do nível



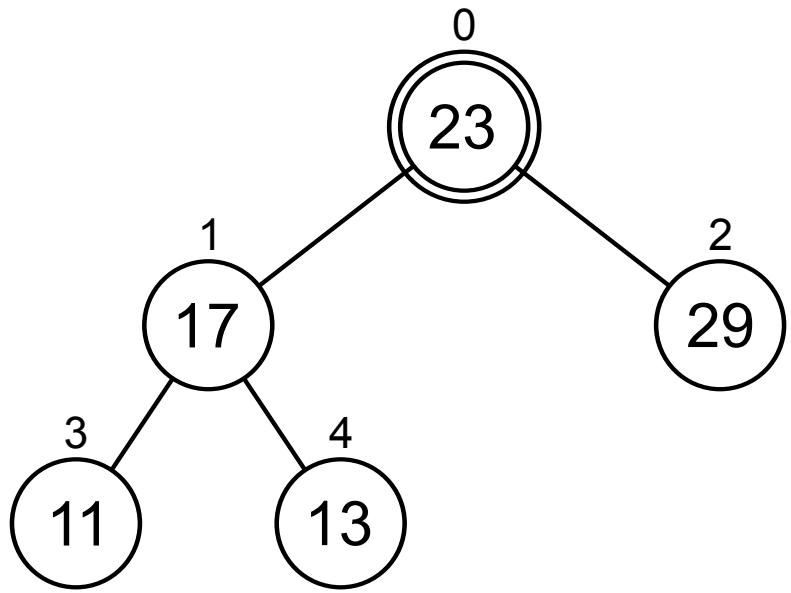
Heap

- Remoção da árvore heap
  - O número sempre é removido da raiz da árvore
  - É feita sua substituição pelo último número



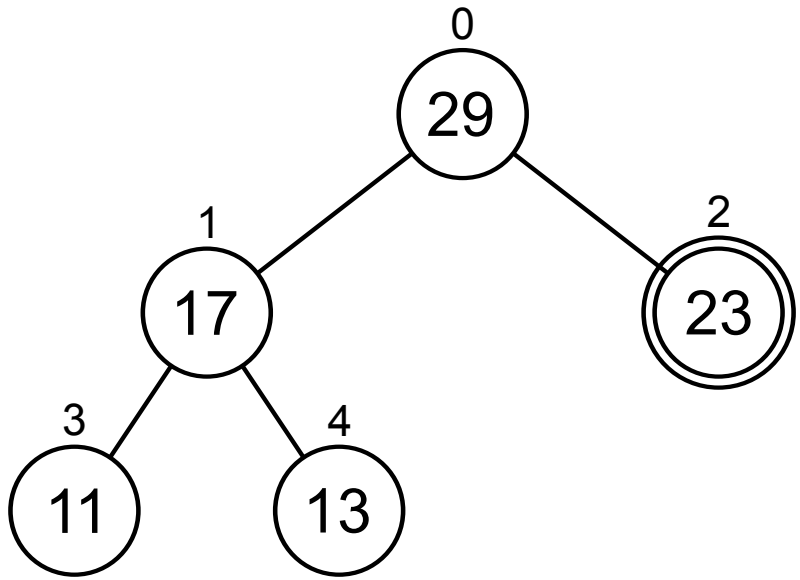
Heap

- Remoção da árvore heap
  - Para manter as propriedades do heap, é aplicado o procedimento heapify na raiz



Heap

- Remoção da árvore heap
  - Para manter as propriedades do heap, é aplicado o procedimento heapify na raiz



Heap

- Análise de complexidade
  - Espaço $O(1)$
  - Tempo $O(h)$

Exemplo

- Construa uma árvore heap mínimo e máximo
  - Considere os números do vetor abaixo
  - Ilustre a construção passo a passo

13	2	34	11	7	43	9
0	1	2	3	4	5	6

Exercício

- A empresa de tecnologia Poxim Tech e a empresa de capitalização Banana Cap estão desenvolvendo um sistema para apuração eficiente dos resultados dos concursos de loteria realizados
  - Os apostadores podem escolher 15 números dentre os valores de 1 até 50, sendo igualmente premiados por faixa as apostas com maior e menor número de acertos, impedindo a acumulação do prêmio
  - Em cada concurso são sorteados 10 números distintos que permitem aos apostadores obterem entre 0 e 10 acertos para cada aposta
  - O código da aposta é representado por um número hexadecimal único de 64 bits

Exercício

- Formato de arquivo de entrada
  - [*Prêmio em reais*]
  - [#*Quantidade de apostas*]
  - [$Sorteado_1$] ... [$Sorteado_{10}$]
  - [$Código_1$] [$Número_{1_1}$] ... [$Número_{1_{15}}$]
  - ⋮
  - [$Código_n$] [$Número_{n_1}$] ... [$Número_{n_{15}}$]

```
3000
5
1 2 3 5 8 13 25 33 42 48
1234567890ef9def 1 2 3 9 11 17 19 20 21 30 34 38 39 40 44
45d45def89bc120a 2 3 5 9 13 14 15 17 18 20 33 35 40 41 42
e7967890aef0a86b 1 2 5 8 9 11 15 16 19 21 27 33 35 42 49
2a1289b0abc78def 3 4 5 7 11 16 18 20 24 25 31 34 35 42 50
890a5178bce7efd6 3 4 7 9 15 18 23 24 26 31 32 38 41 43 48
```

Exercício

- Formato de arquivo de saída
  - São exibidos as duas faixas de acerto e a quantidade de apostas premiadas, além da listagem dos códigos das apostas em cada faixa

```
Faixa: 6 acertos
45d45def89bc120a
e7967890aef0a86b
2 ganhadores de R$750

Faixa: 2 acertos
890a5178bce7efd6
1 ganhador de R$1500
```