Árvores binárias

Prof: Bruna Moreira
brunamoreira@unb.br

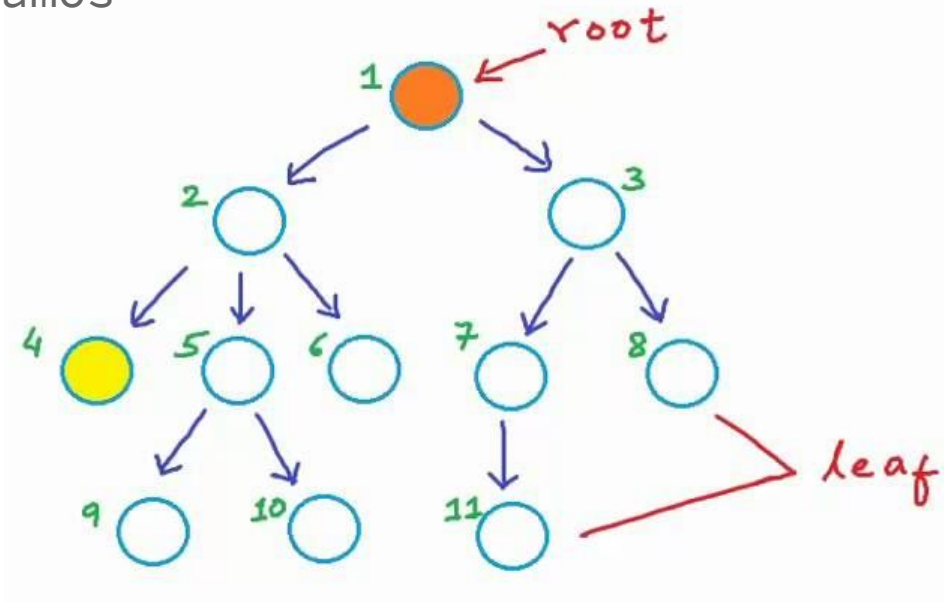
Árvores

Estruturas compostas de nós e ramos

Estruturas não lineares

- Único predecessor
- N sucessores

Caminho único da raiz até o nó



Listas

Linear

Estruturas compostas por nós

Valor

Ponteiro para *next* e *prev*

HEAD - primeiro nó

TAIL - último nó

Árvores

Não-linear

Estrutura composta por nós

Valor

Ponteiro para filhos (n)

ROOT - primeiro nó

LEAVES - últimos nós

Árvores: Características

Nível do nó

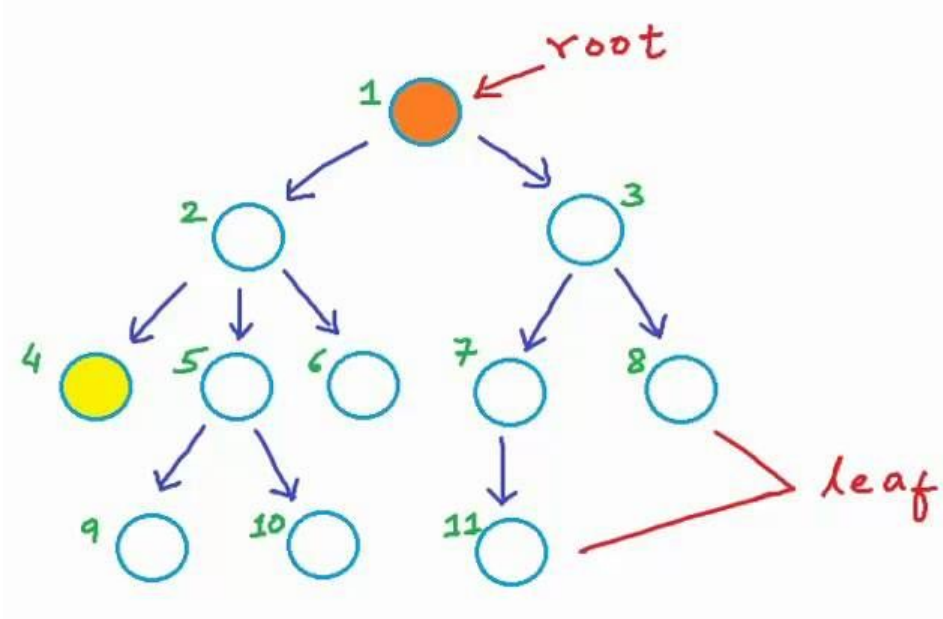
- Caminho da raiz até o nó +1
- Quantidade de nós no caminho

Altura da árvore

- Maior nível entre os nós

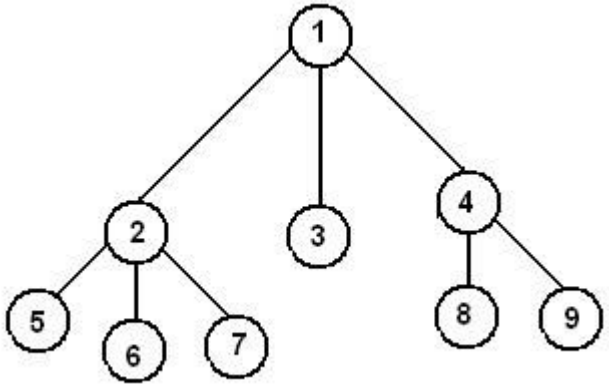
Altura

- Árvore vazia: 0
- Árvore com um elemento: 1



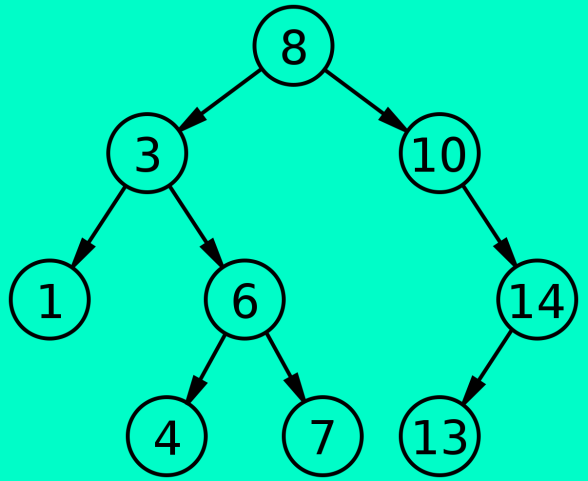
Árvore N-Ária

N filhos (nulos ou não)



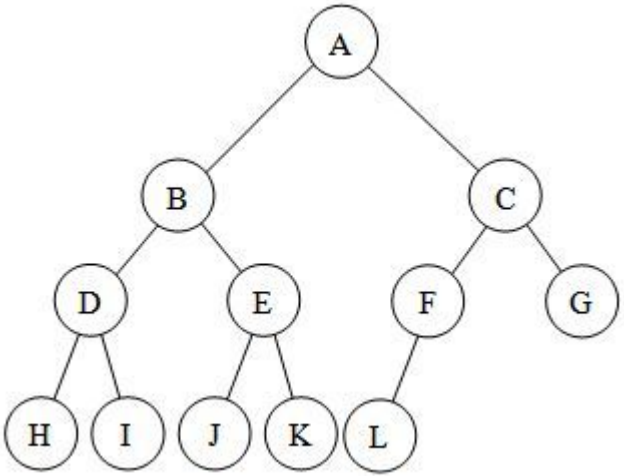
Árvore Binária

2 filhos (nulos ou não)



Árvores: Características

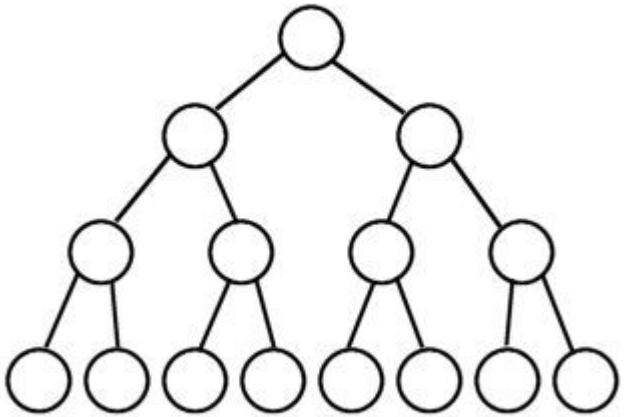
Árvore completa



Sub-árvore vazia no penúltimo ou último nível

Pode ter um filho (L)

Árvore cheia

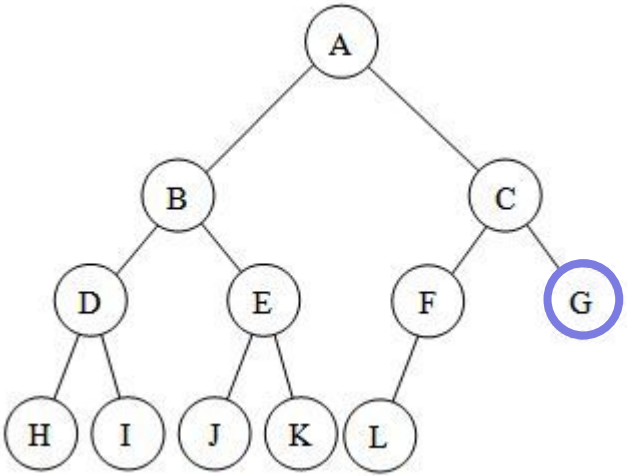


Sub-árvore vazia no último nível

Nós têm 0 ou 2 filhos

Árvores: Características

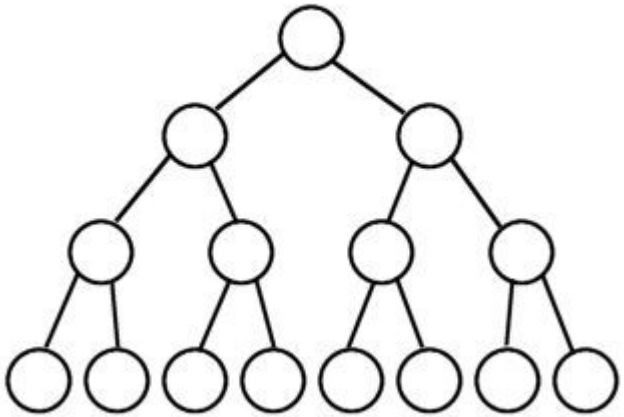
Árvore completa



Sub-árvore vazia no **penúltimo** ou último nível

Pode ter um filho (L)

Árvore cheia

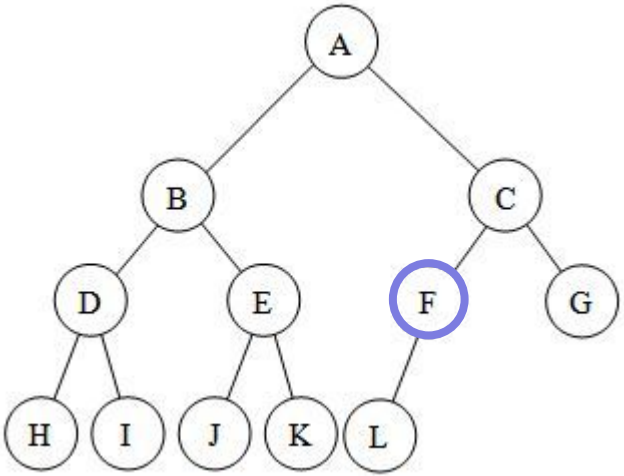


Sub-árvore vazia no último nível

Nós têm 0 ou 2 filhos

Árvores: Características

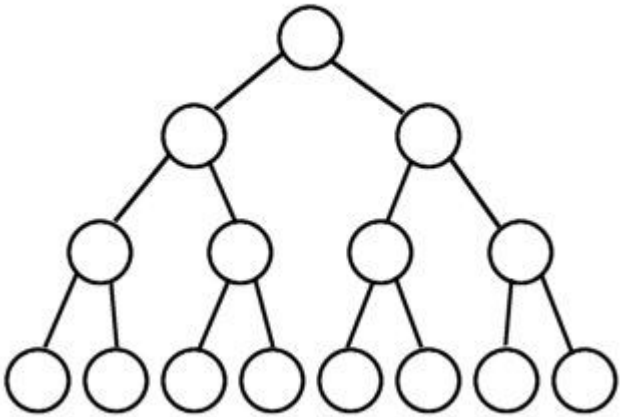
Árvore completa



Sub-árvore vazia no penúltimo ou último nível

Pode ter **um filho**

Árvore cheia

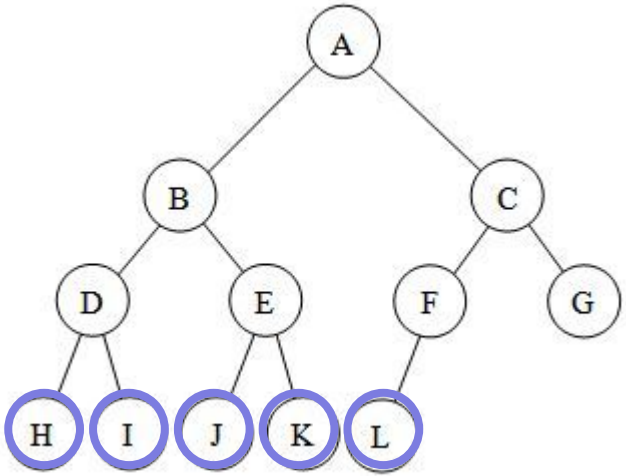


Sub-árvore vazia no último nível

Nós têm 0 ou 2 filhos

Árvores: Características

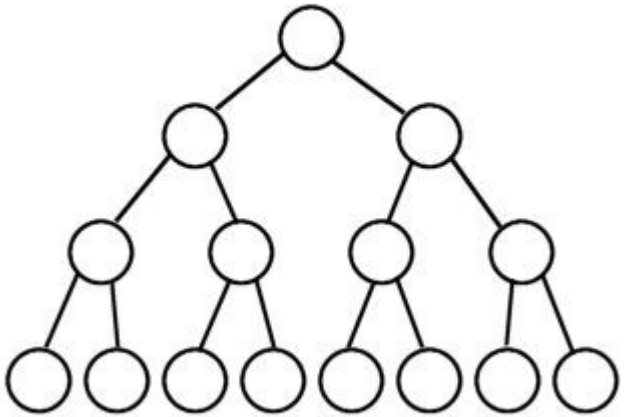
Árvore completa



Sub-árvore vazia no penúltimo ou **último** nível

Pode ter um filho (L)

Árvore cheia

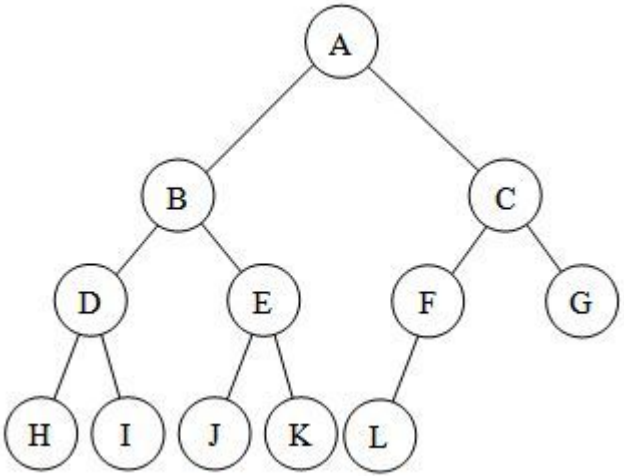


Sub-árvore vazia no último nível

Nós têm 0 ou 2 filhos

Árvores: Características

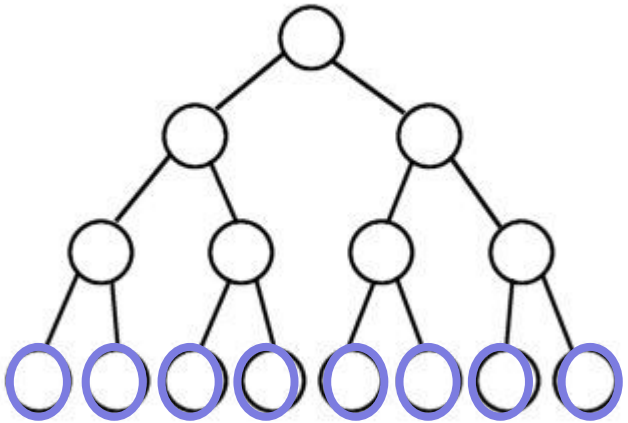
Árvore completa



Sub-árvore vazia no penúltimo ou último nível

Pode ter um filho (L)

Árvore cheia

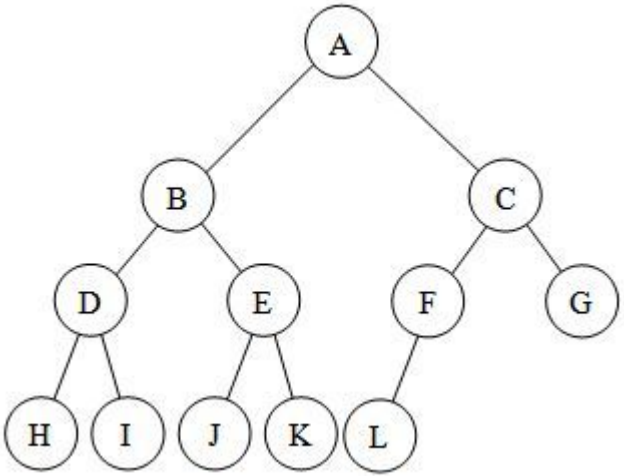


Sub-árvore vazia no **último** nível

Nós têm 0 ou 2 filhos

Árvores: Características

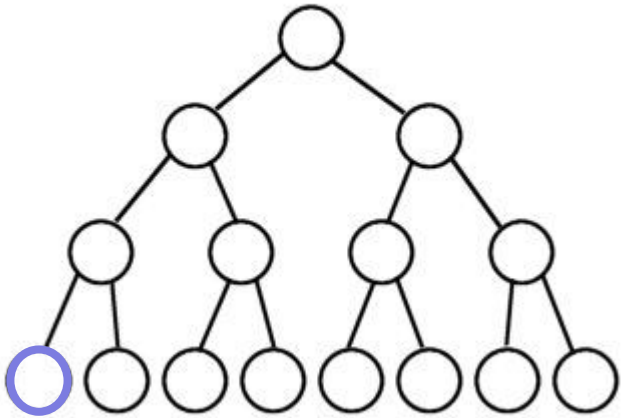
Árvore completa



Sub-árvore vazia no penúltimo ou último nível

Pode ter um filho (L)

Árvore cheia

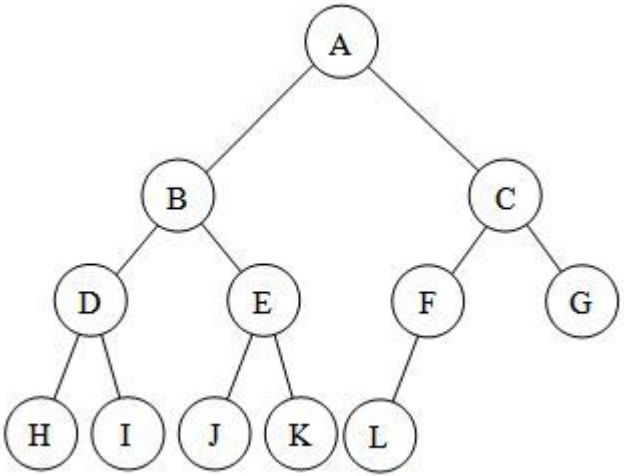


Sub-árvore vazia no último nível

Nós têm **0** ou 2 filhos

Árvores: Características

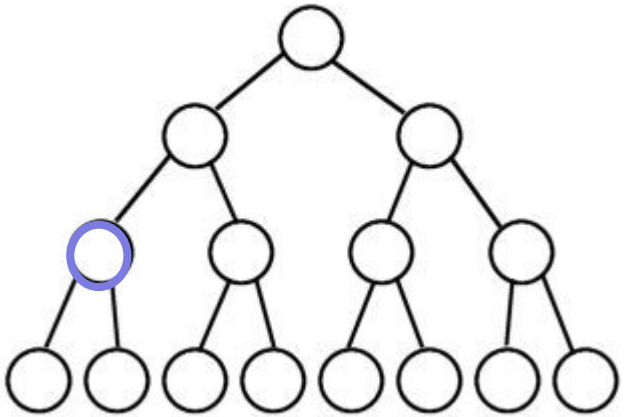
Árvore completa



Sub-árvore vazia no penúltimo ou último nível

Pode ter um filho (L)

Árvore cheia



Sub-árvore vazia no último nível

Nós têm 0 ou **2** filhos

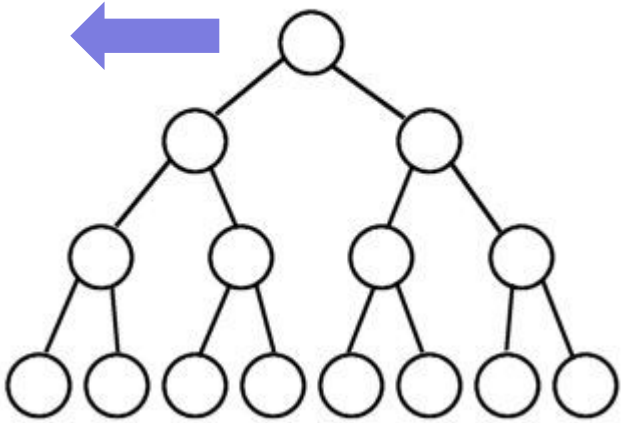
Altura

Nível 1: 1 1

Por nível

Total

Árvore cheia



Sub-árvore vazia no último nível

Nós têm 0 ou 2 filhos

Altura

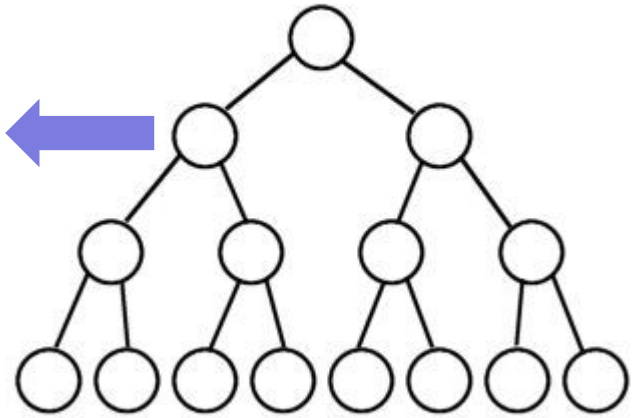
Nível 1: 1 1

Nível 2: 2 3

Por nível

Total

Árvore cheia



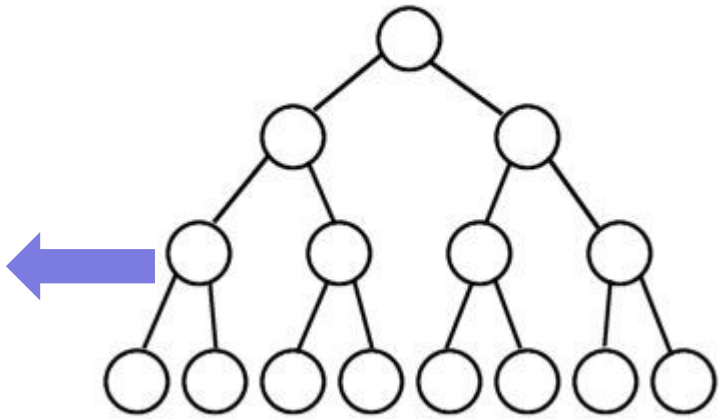
Sub-árvore vazia no último nível

Nós têm 0 ou 2 filhos

Altura

Nível 1:	1	1
Nível 2:	2	3
Nível 3:	4	7

Árvore cheia



Por nível

Total

Sub-árvore vazia no último nível

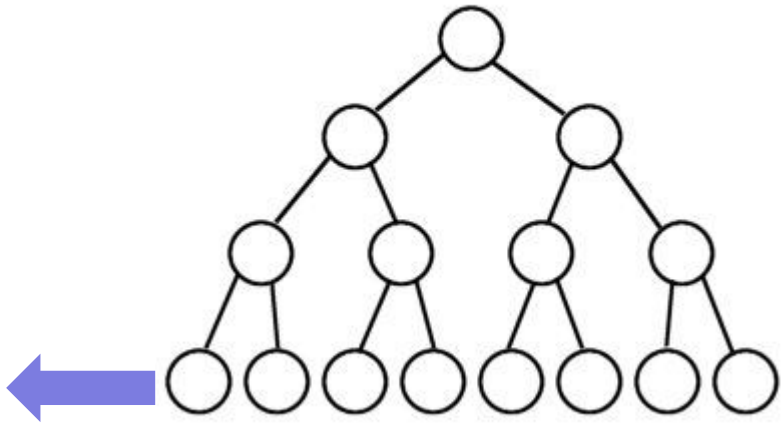
Nós têm 0 ou 2 filhos

Altura

Nível 1:	1	1
Nível 2:	2	3
Nível 3:	4	7
Nível 4:	8	15

Por nível **Total**

Árvore cheia



Sub-árvore vazia no último nível

Nós têm 0 ou 2 filhos

Altura

Nível 1: 1

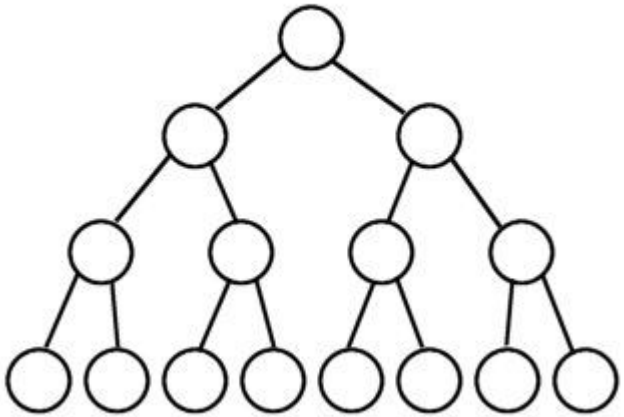
Nível 2: 3

Nível 3: 7

Nível 4: 15

Por nível **Total**

Árvore cheia



Sub-árvore vazia no último nível

Nós têm 0 ou 2 filhos

Altura

Nível 1: 1

Nível 2: 3

Nível 3: 7

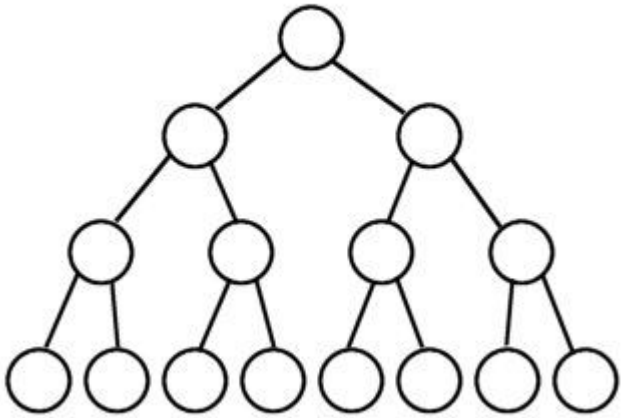
Nível 4: 15

$2^h - 1$

Por nível

Total

Árvore cheia



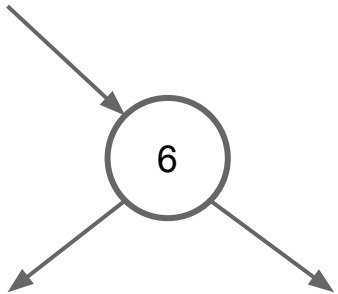
Sub-árvore vazia no último nível

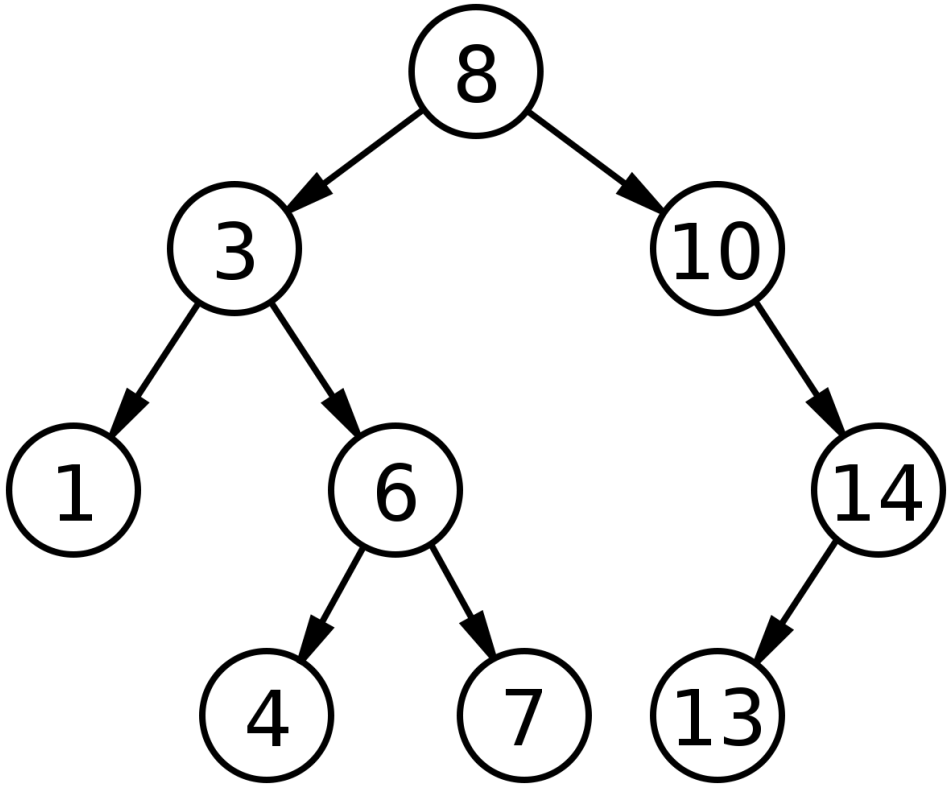
Nós têm 0 ou 2 filhos

Elementos

Nó

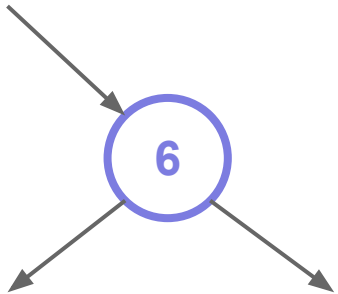
- Valor
- Esquerda
- Direita
- Pai

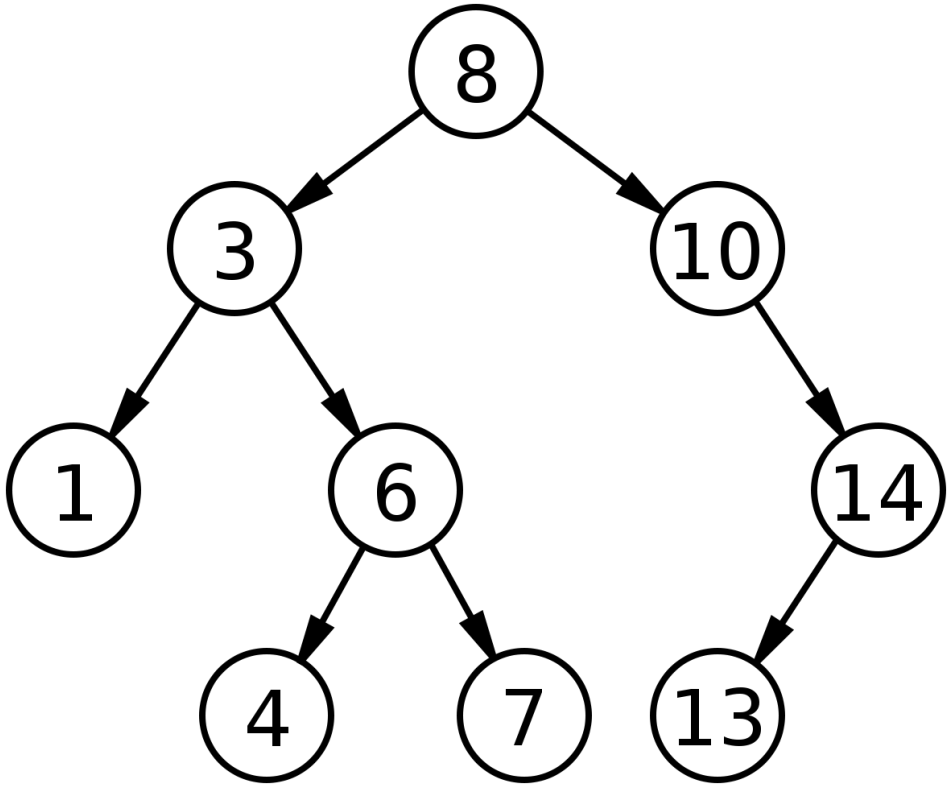




Nó

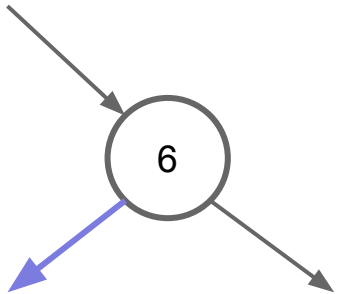
- **Valor**
- Esquerda
- Direita
- Pai

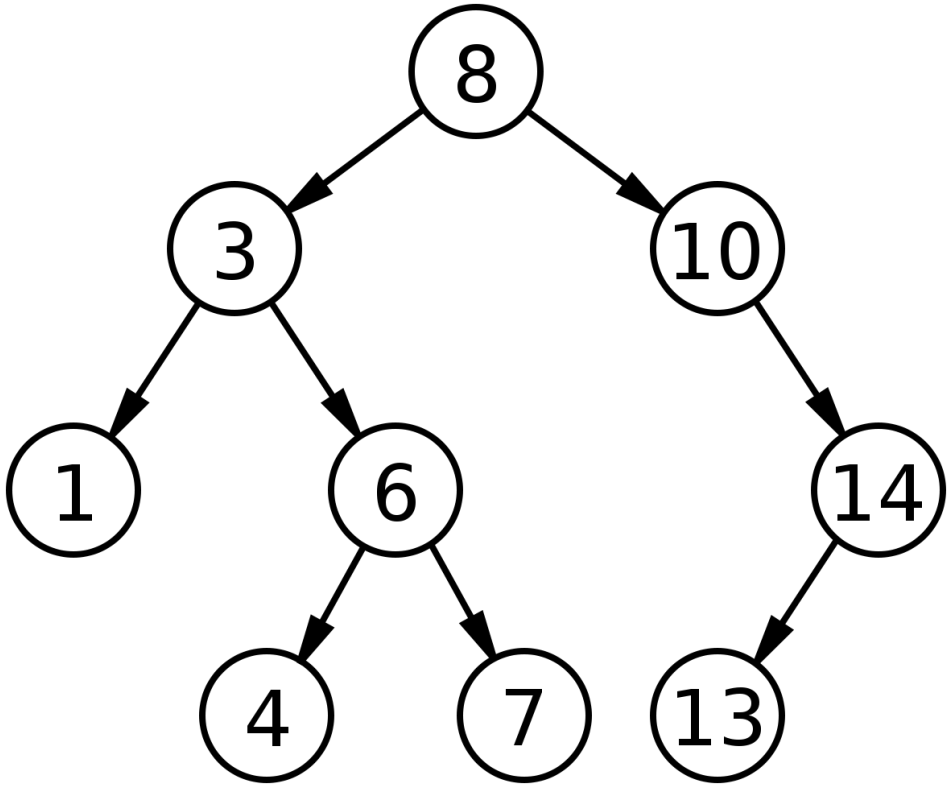




Nó

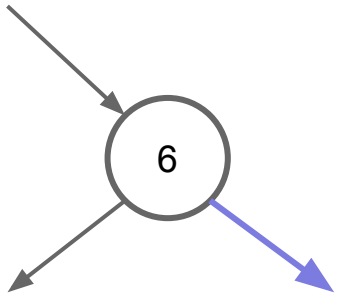
- Valor
- **Esquerda**
- Direita
- Pai

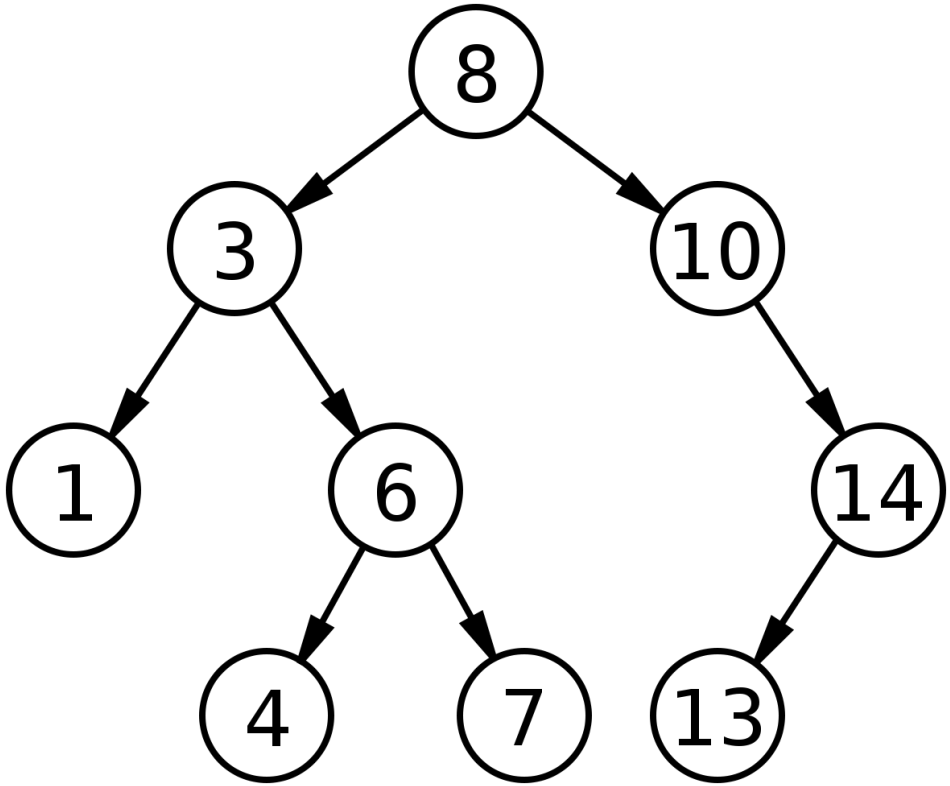




Nó

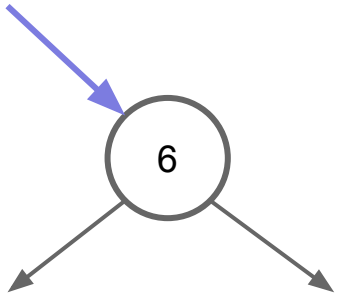
- Valor
- Esquerda
- **Direita**
- Pai

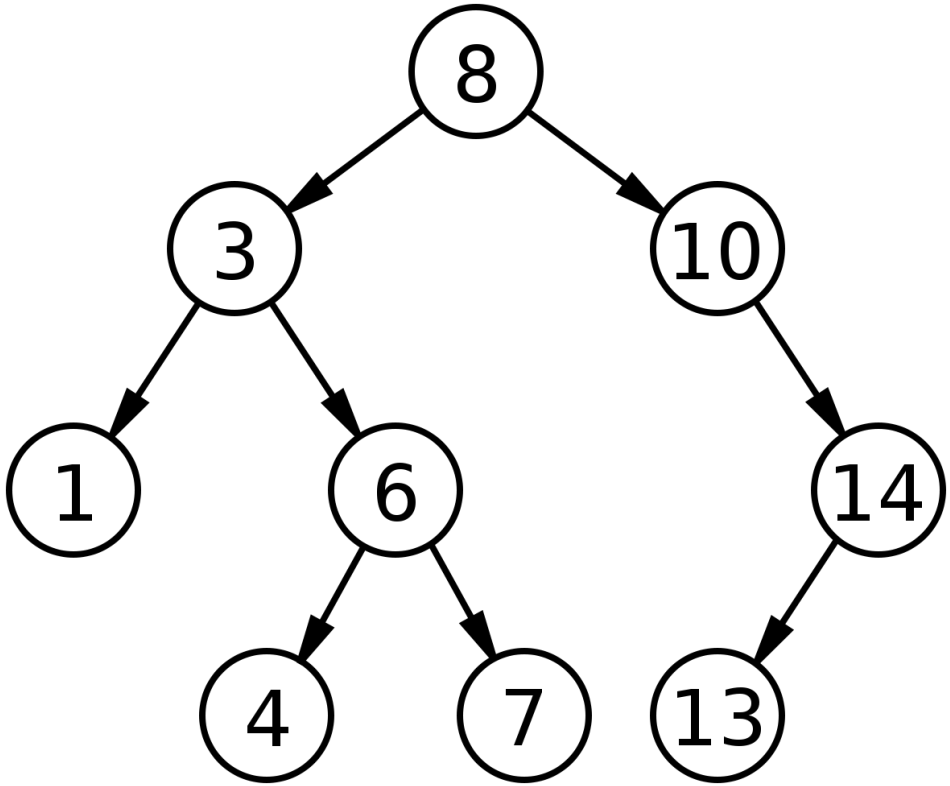




Nó

- Valor
- Esquerda
- Direita
- **Pai**





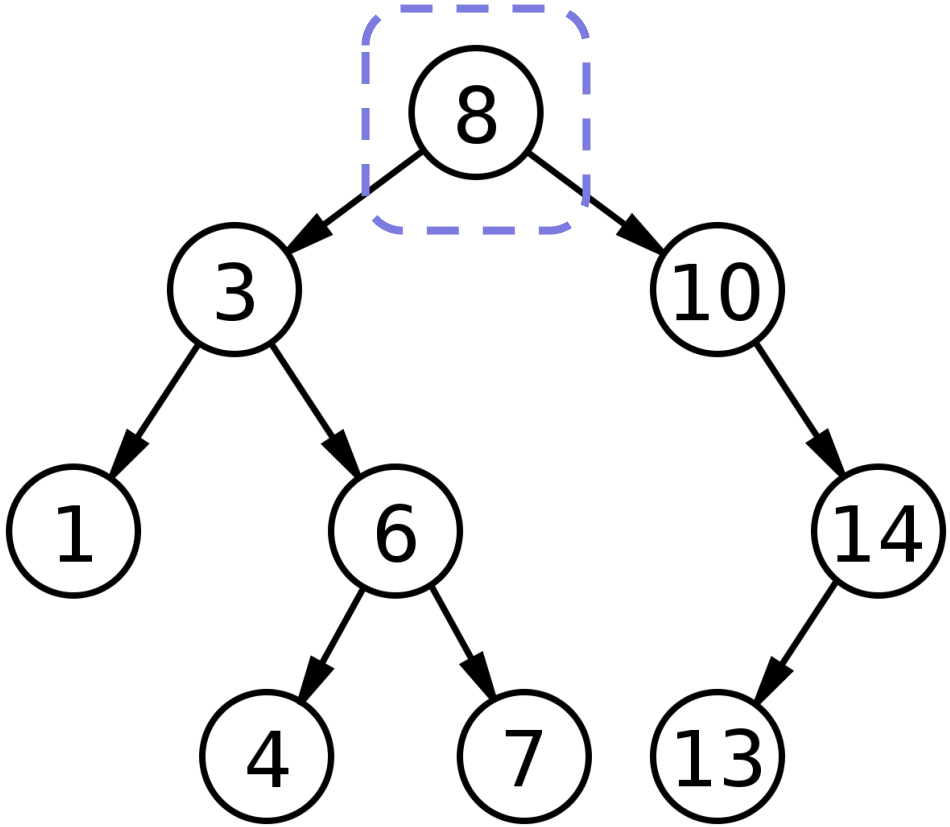
Root/raiz

Início da árvore

Navega para outros itens

É o primeiro nó

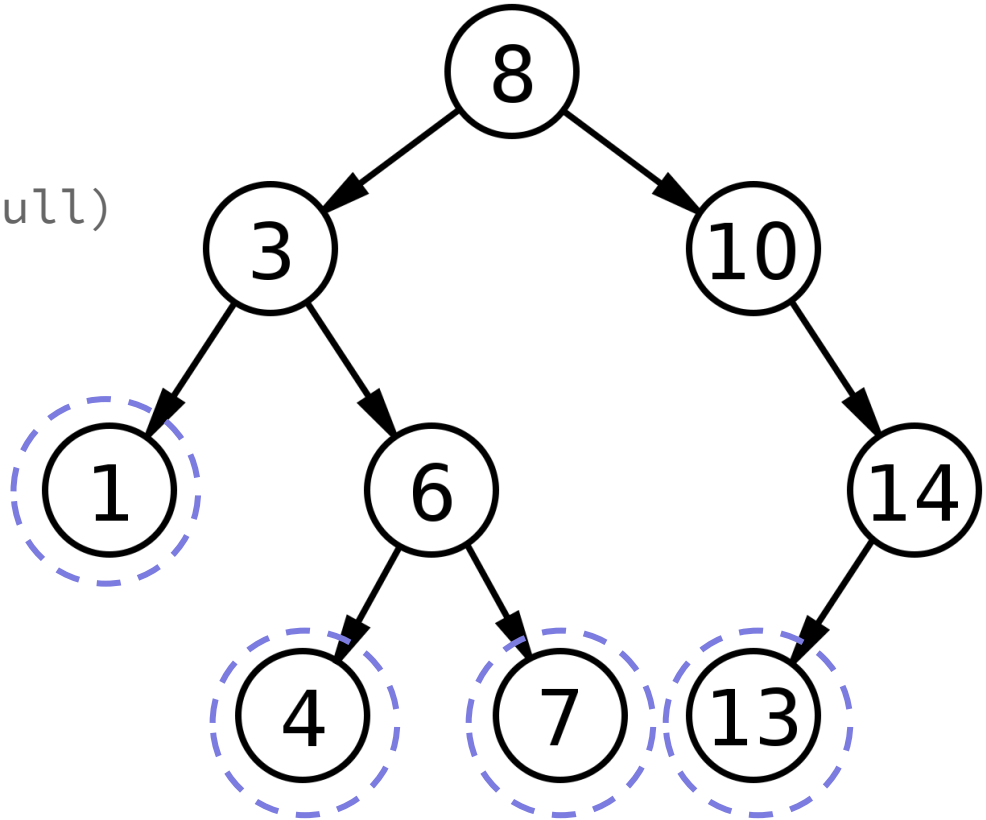
Não tem pai (link para null)



Folhas

Não têm filhos (link para null)

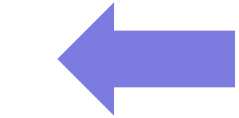
Dead-ends (fim da linha)



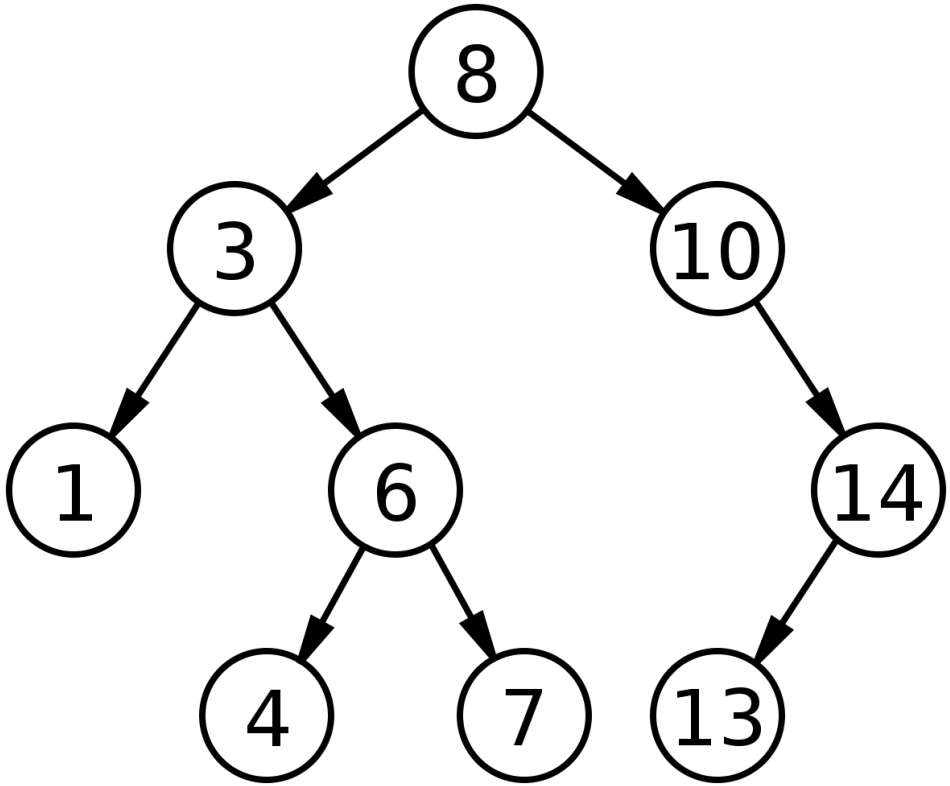
Organização

Organização

Esquerda: valores menores



Direita: valores maiores



Operações

Inserção

Busca

Remoção

visualgo

Criação

{**42**, 16, 57, 48, 63, 35, 8, 11, 5}

- Inserir a raiz

Inserção

{**42**, 16, 57, 48, 63, 35, 8, 11, 5}

Inserção

{42, **16**, 57, 48, 63, 35, 8, 11, 5}

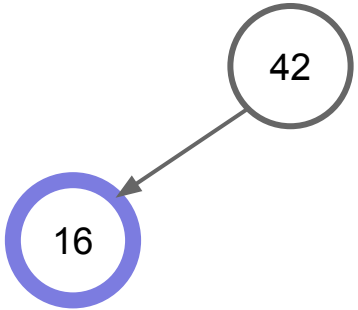
Inserção

{42, **16**, 57, 48, 63, 35, 8, 11, 5}

Esquerda

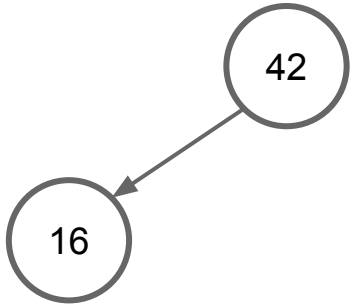
Inserção

{42, **16**, 57, 48, 63, 35, 8, 11, 5}



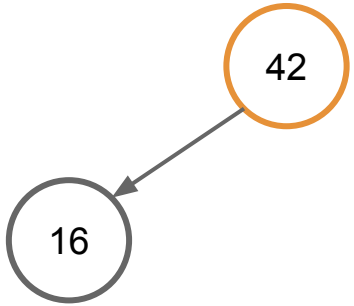
Inserção

{42, 16, **57**, 48, 63, 35, 8, 11, 5}



Inserção

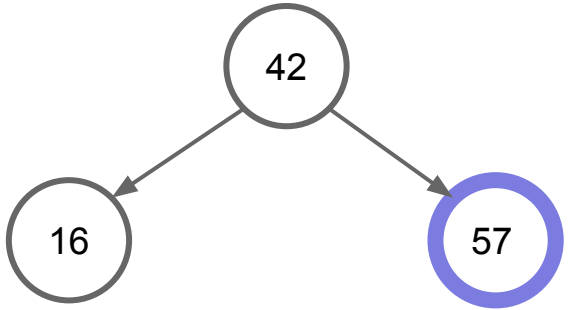
{42, 16, **57**, 48, 63, 35, 8, 11, 5}



Direita

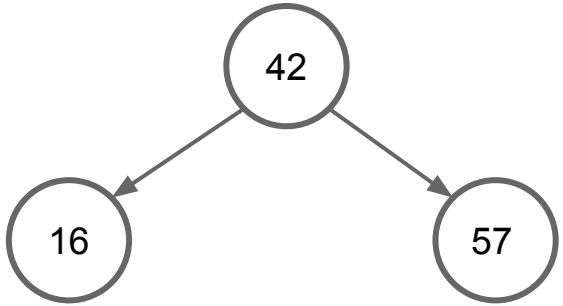
Inserção

{42, 16, **57**, 48, 63, 35, 8, 11, 5}



Inserção

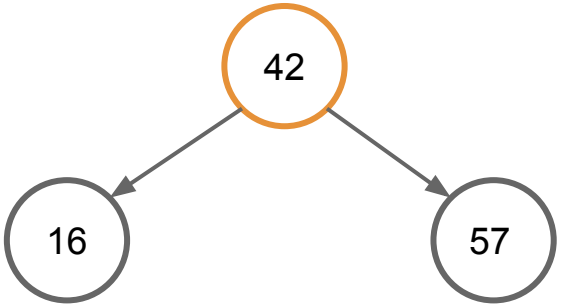
{42, 16, 57, **48**, 63, 35, 8, 11, 5}



Inserção

{42, 16, 57, **48**, 63, 35, 8, 11, 5}

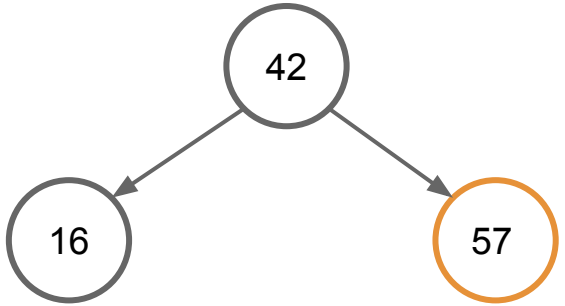
Direita



Inserção

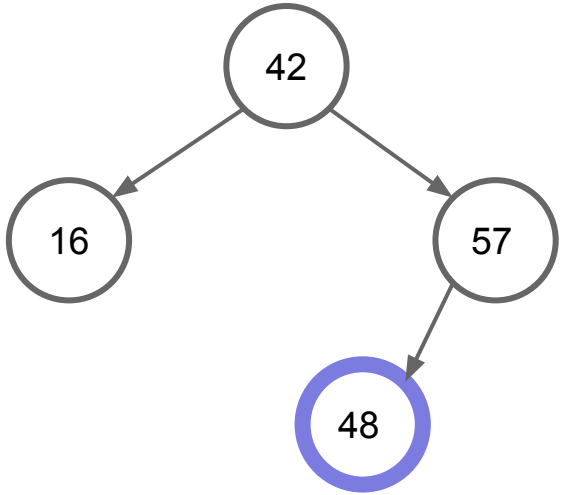
{42, 16, 57, **48**, 63, 35, 8, 11, 5}

Esquerda



Inserção

{42, 16, 57, **48**, 63, 35, 8, 11, 5}

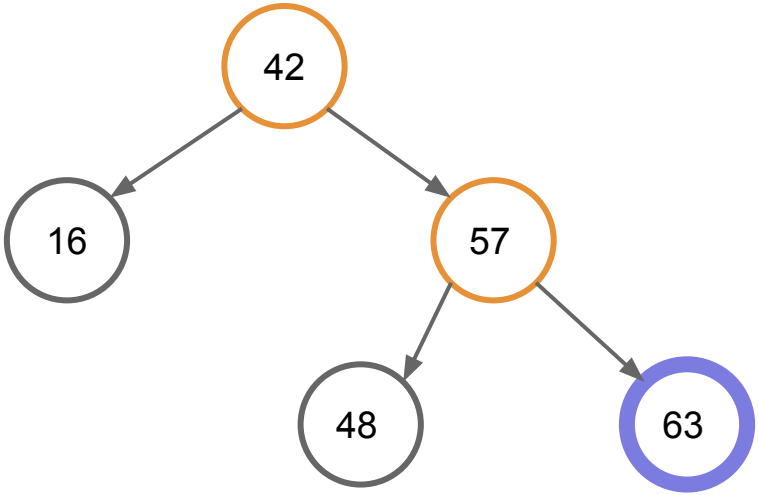


Inserção

{42, 16, 57, 48, **63**, 35, 8, 11, 5}

Direita

Direita

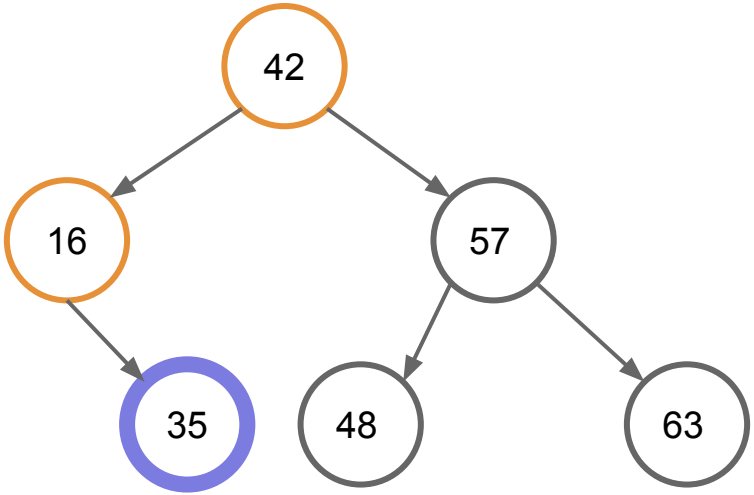


Inserção

{42, 16, 57, 48, 63, **35**, 8, 11, 5}

Esquerda

Direita

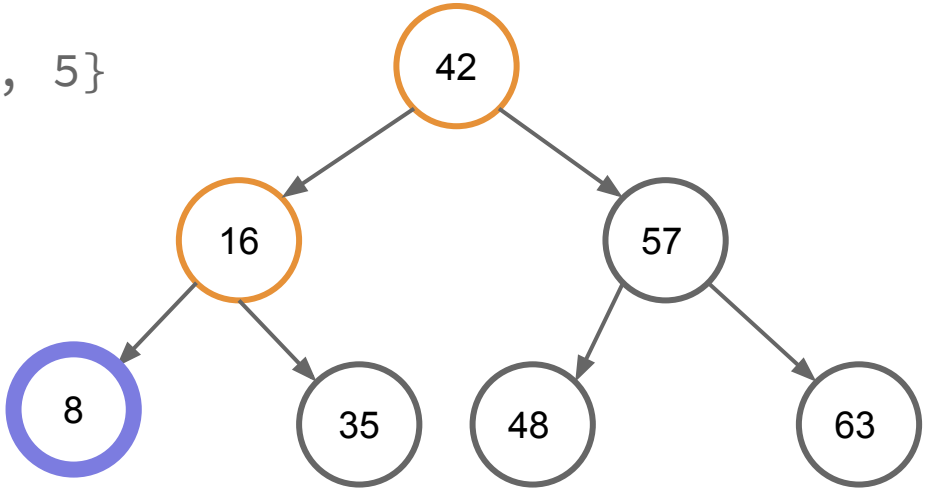


Inserção

{42, 16, 57, 48, 63, 35, **8**, 11, 5}

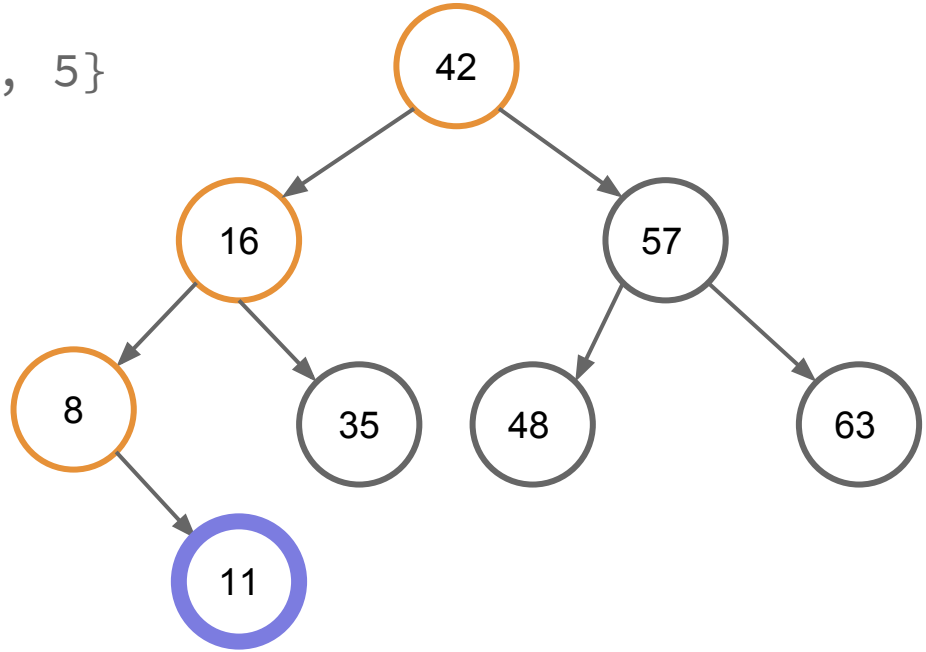
Esquerda

Esquerda



Inserção

{42, 16, 57, 48, 63, 35, 8, **11**, 5}



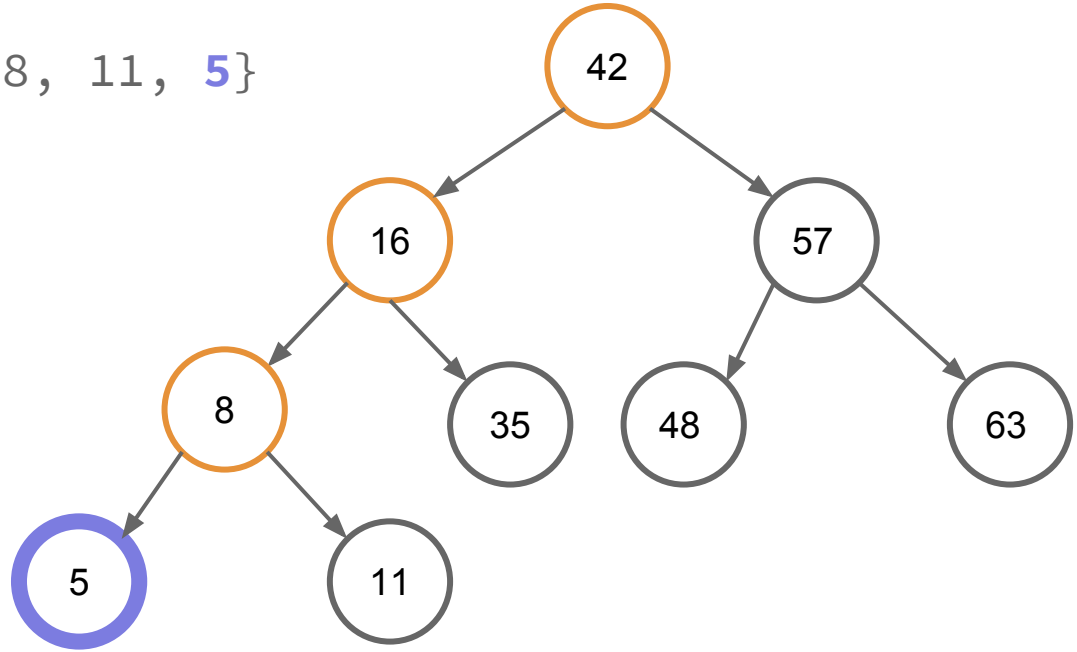
Inserção

{42, 16, 57, 48, 63, 35, 8, 11, **5**}

Esquerda

Esquerda

Esquerda



Inserção

1) no = raiz, valor_inserir = 48
2) SE no == NULL:
    a) raiz = valor_inserir
3) SE no.valor > valor_inserir:
    a) Vai para esquerda
    b) Volta para o 2
4) SE no.valor < valor_inserir:
    a) Vai para direita
    b) Volta para o 2

Inserção

1) no = raiz, valor_inserir = 48
2) SE no == NULL:
    a) raiz = valor_inserir
3) SE no.valor > valor_inserir:
    a) Vai para esquerda
    b) Volta para o 2
4) SE no.valor < valor_inserir:
    a) Vai para direita
    b) Volta para o 2

?

Complexidade

Insere nas folhas

Altura da árvore

Total de nós: $2^h - 1$

Inserção

1) no = raiz, valor_inserir = 48
2) SE no == NULL:
    a) raiz = valor_inserir
3) SE no.valor > valor_inserir:
    a) Vai para esquerda
    b) Volta para o 2
4) SE no.valor < valor_inserir:
    a) Vai para direita
    b) Volta para o 2

O(log n)

Insere nas folhas

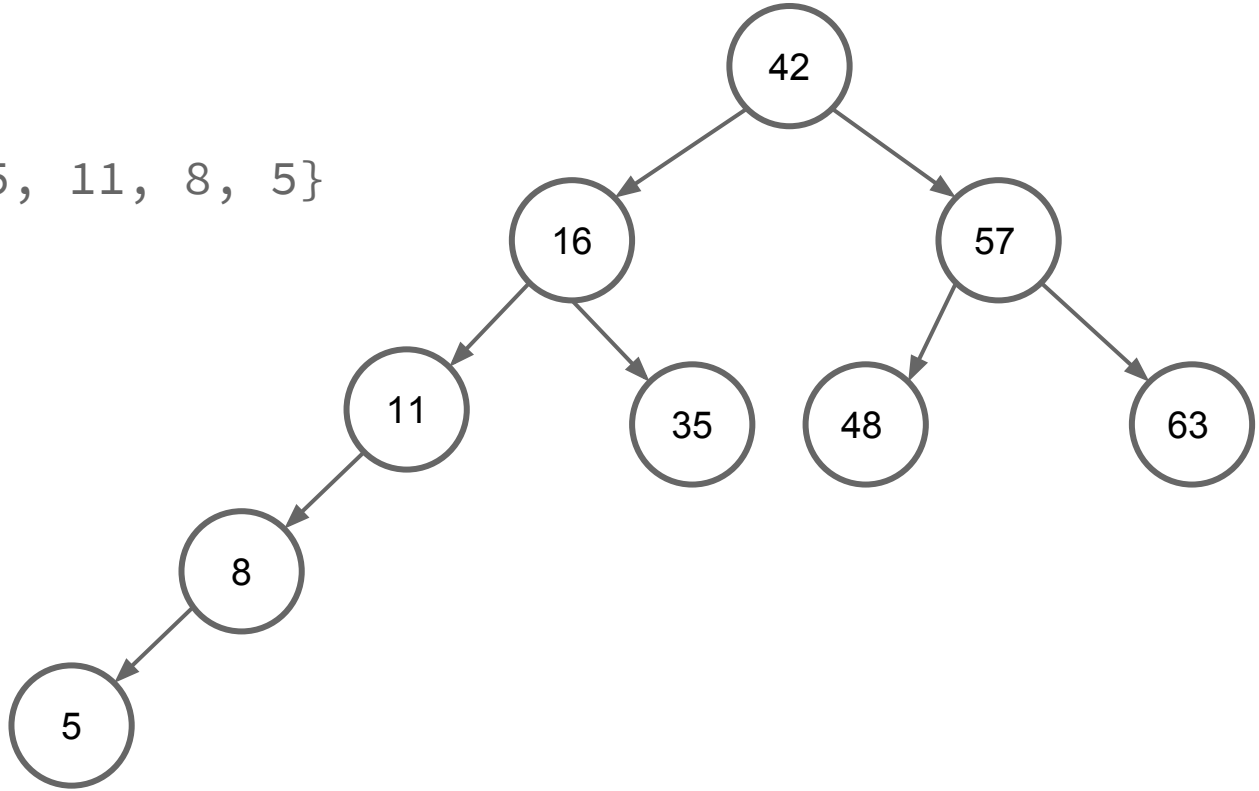
Altura da árvore

Total de nós: $2^h - 1$

É possível que árvores com os mesmos elementos sejam diferentes?

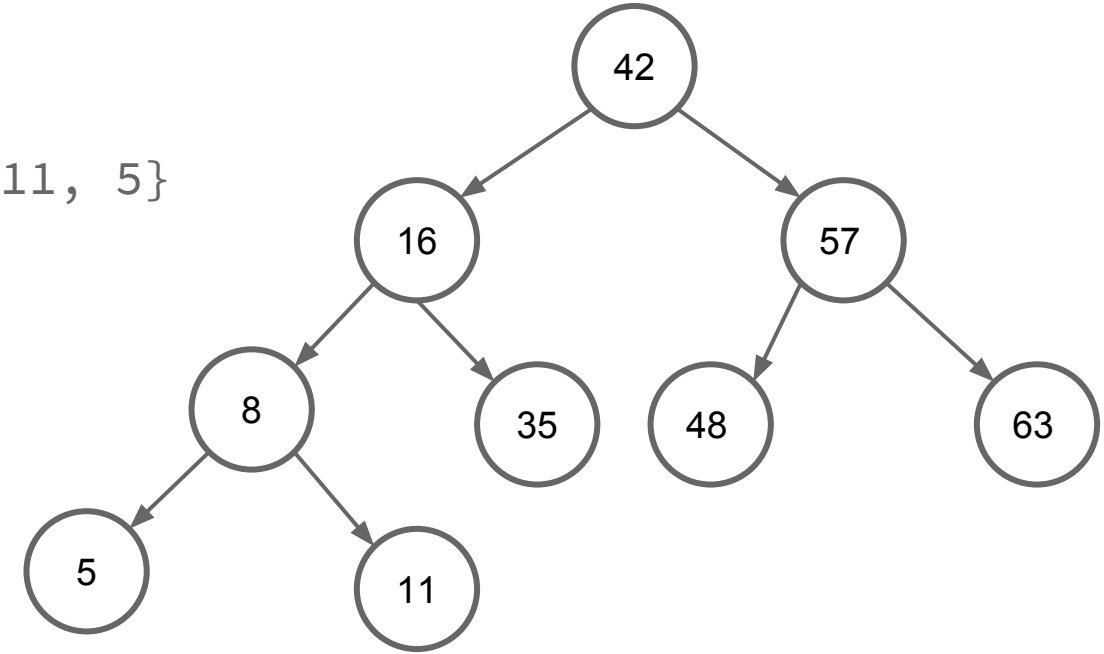
Ordem de inserção

{42, 16, 57, 48, 35, 11, 8, 5}



Ordem de inserção

{42, 16, 57, 48, 35, 8, 11, 5}



Operações

Inserção

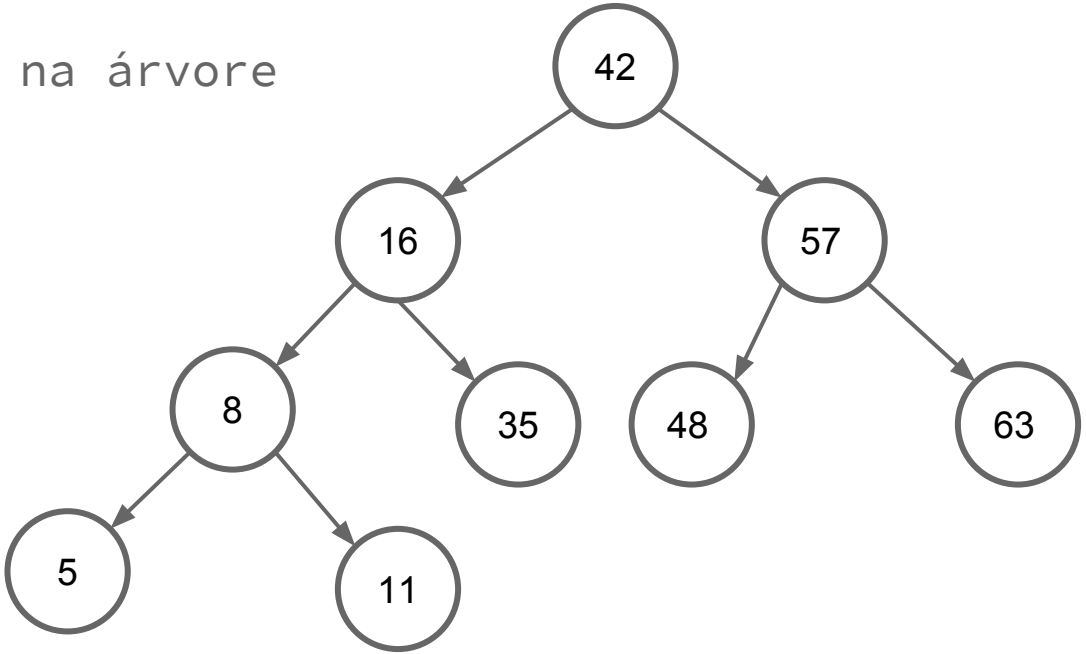
Busca

Remoção

visualgo

Busca

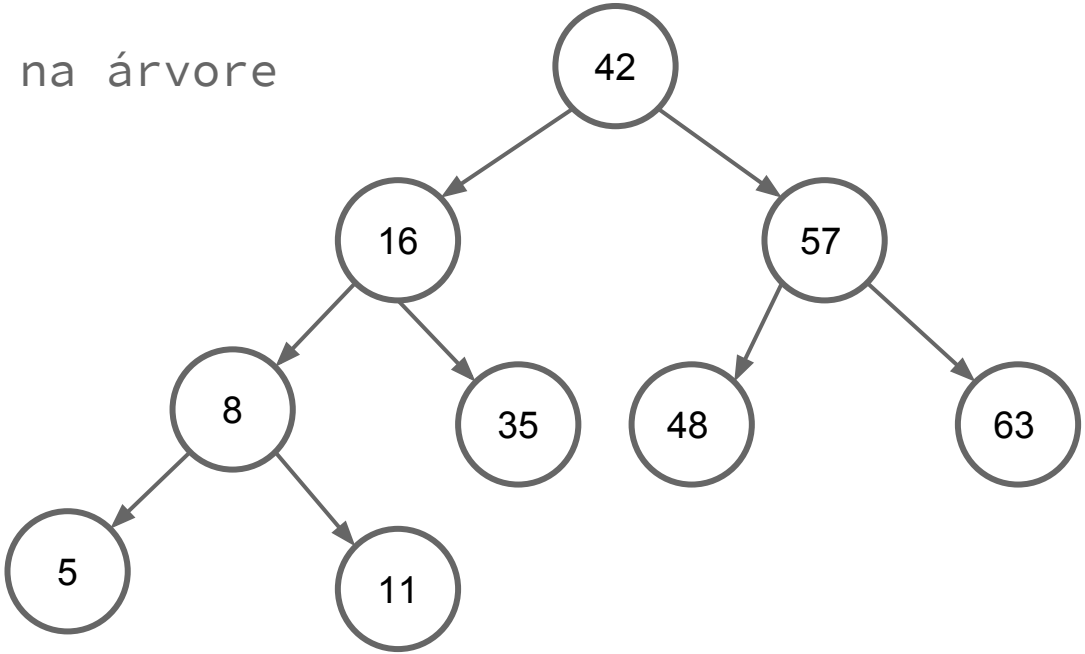
- Buscar número que tem na árvore



Busca

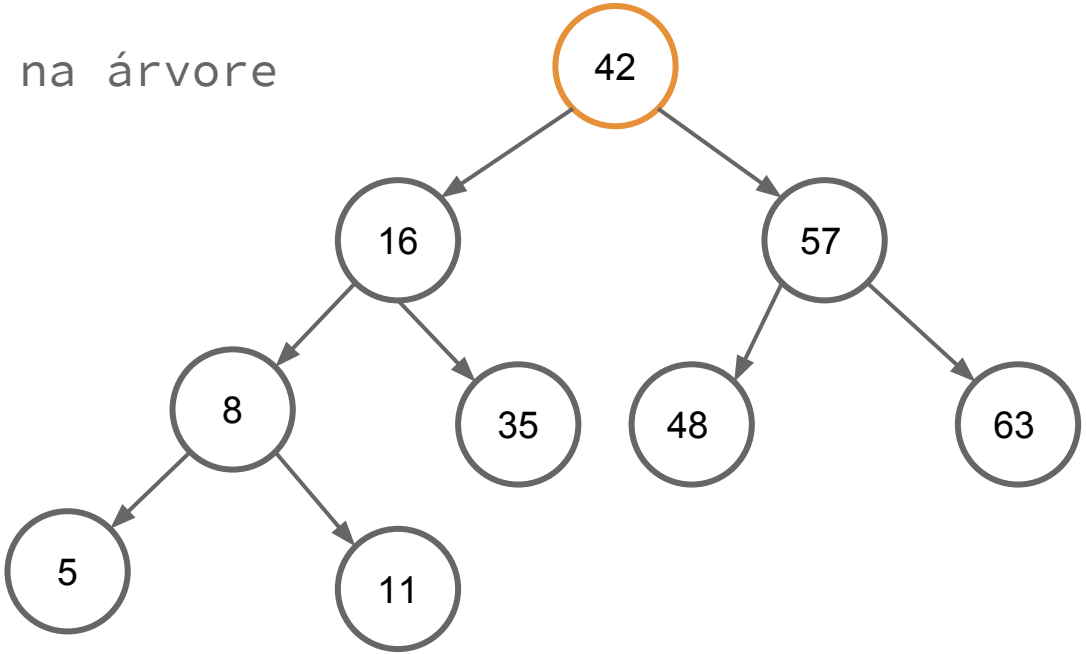
- Buscar número que tem na árvore

48



Busca

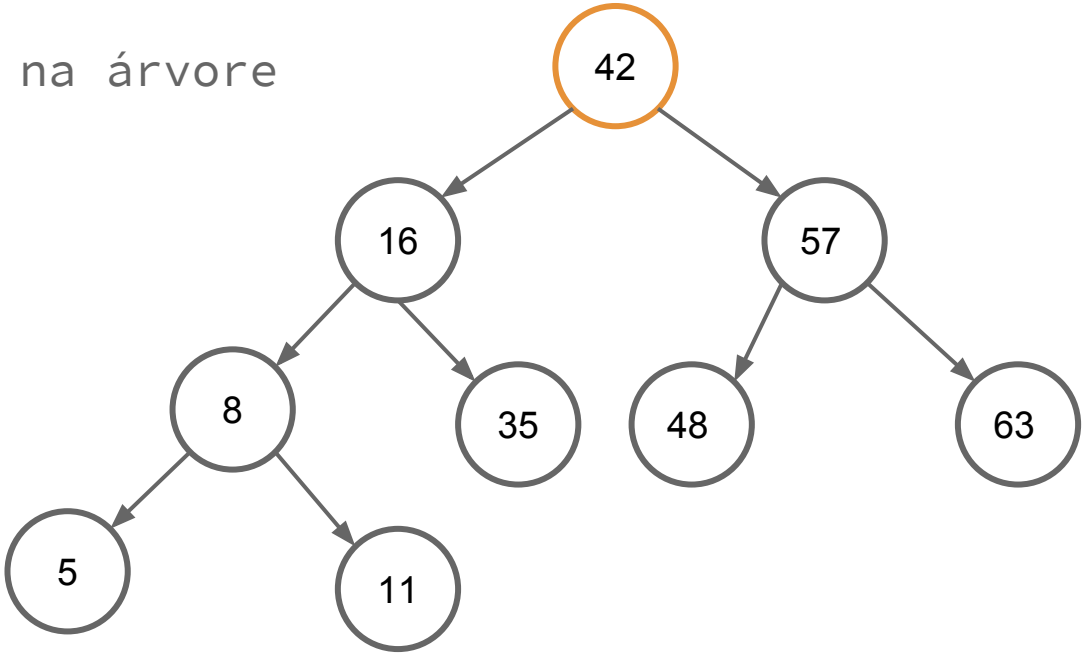
- Buscar número que tem na árvore



Busca

- Buscar número que tem na árvore

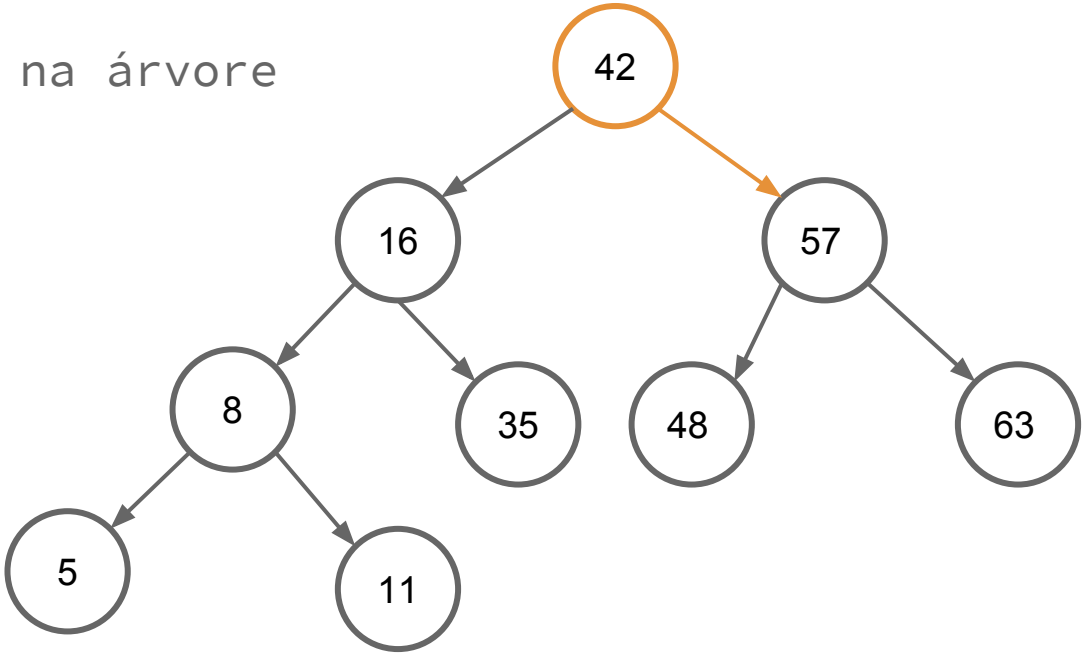
Direita



Busca

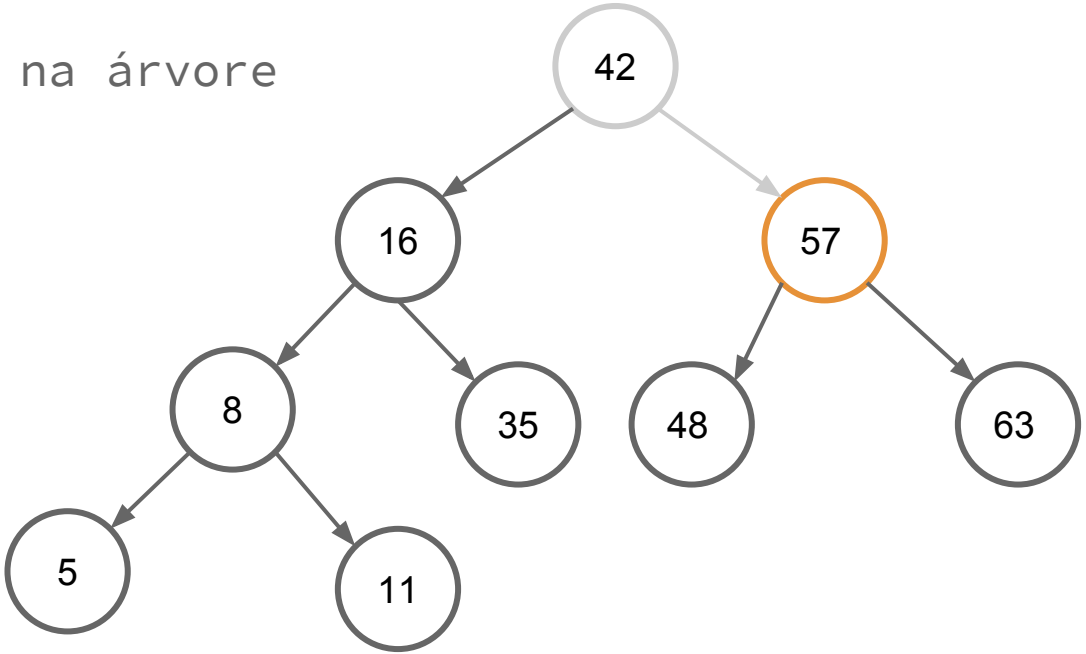
- Buscar número que tem na árvore

Direita



Busca

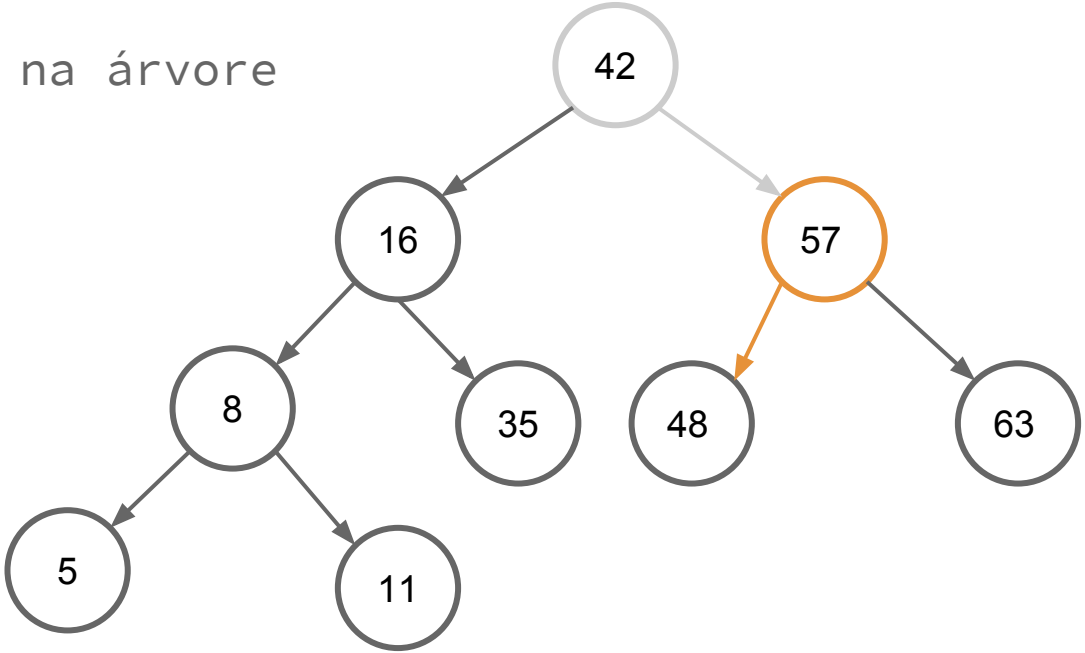
- Buscar número que tem na árvore



Busca

- Buscar número que tem na árvore

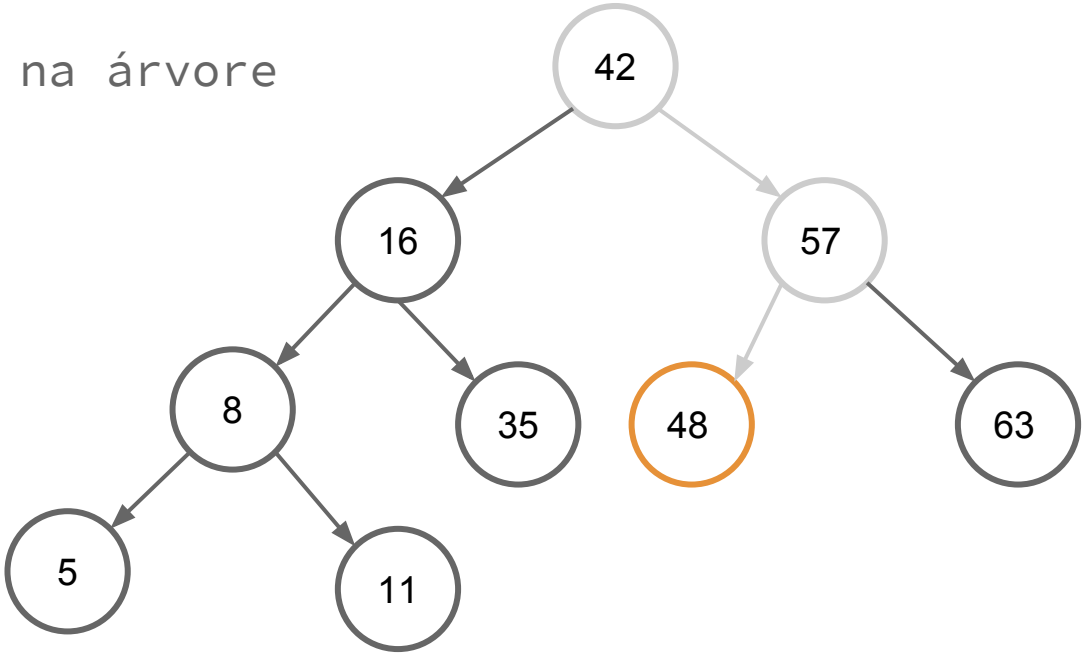
Esquerda



Busca

- Buscar número que tem na árvore

Achou!



Busca

1) no = raiz, valor_busca = 48
2) SE no == NULL:
    a) Não existe o valor na árvore
3) SE no.valor == valor_busca:
    a) Achou!
4) SE no.valor > valor_busca:
    a) Vai para esquerda
    b) Volta para o 2
5) SE no.valor < valor_busca:
    a) Vai para direita
    b) Volta para o 2

Busca

1) no = raiz, valor_busca = 48
2) SE no == NULL:
    a) Não existe o valor na árvore
3) SE no.valor == valor_busca:
    a) Achou!
4) SE no.valor > valor_busca:
    a) Vai para esquerda
    b) Volta para o 2
5) SE no.valor < valor_busca:
    a) Vai para direita
    b) Volta para o 2

Pior caso: folha

Busca

1) no = raiz, valor_busca = 48
2) SE no == NULL:
    a) Não existe o valor na árvore
3) SE no.valor == valor_busca:
    a) Achou!
4) SE no.valor > valor_busca:
    a) Vai para esquerda
    b) Volta para o 2
5) SE no.valor < valor_busca:
    a) Vai para direita
    b) Volta para o 2

Pior caso: folha

Altura da árvore

Busca

1) no = raiz, valor_busca = 48
2) SE no == NULL:
    a) Não existe o valor na árvore
3) SE no.valor == valor_busca:
    a) Achou!
4) SE no.valor > valor_busca:
    a) Vai para esquerda
    b) Volta para o 2
5) SE no.valor < valor_busca:
    a) Vai para direita
    b) Volta para o 2

Pior caso: folha

Altura da árvore

Total de nós: $2^h - 1$

Busca

1) no = raiz, valor_busca = 48
2) SE no == NULL:
    a) Não existe o valor na árvore
3) SE no.valor == valor_busca:
    a) Achou!
4) SE no.valor > valor_busca:
    a) Vai para esquerda
    b) Volta para o 2
5) SE no.valor < valor_busca:
    a) Vai para direita
    b) Volta para o 2

? **Complexidade**

Pior caso: folha

Altura da árvore

Total de nós: $2^h - 1$

Busca

1) no = raiz, valor_busca = 48
2) SE no == NULL:
    a) Não existe o valor na árvore
3) SE no.valor == valor_busca:
    a) Achou!
4) SE no.valor > valor_busca:
    a) Vai para esquerda
    b) Volta para o 2
5) SE no.valor < valor_busca:
    a) Vai para direita
    b) Volta para o 2

O(log n)

Pior caso: folha

Altura da árvore

Total de nós: $2^h - 1$

Operações

Inserção 

Busca 

Remoção

Árvores no C++

Maps

```
#include <map>

map< chave , valor >   nome ;
```

Maps

```
#include <map>

map<string, valor>  nome ;
```

Maps

```
#include <map>

map<string, int> nome;
```

Maps

```
#include <map>

map<string, int> estados;
```

Maps

```
#include <map>

map<string, int> estados;
```

nome[chave] = valor;

Inserção

Maps

```
#include <map>

map<string, int> estados;

estados[ chave ] = valor ;
```

Inserção

Maps

```
#include <map>

map<string, int> estados;

estados["Mato Grosso"] = valor ;
```

Inserção

Maps

```
#include <map>

map<string, int> estados;

estados["Mato Grosso"] = 66  ;
```

Inserção

Maps

```
#include <map>

map<string, int> estados;

estados["Mato Grosso"] = 66;

estados["Bahia"] = 75;

estados["Amazonas"] = 97;
```

Inserção

Maps

```
cout << nome[ chave ] << "\n";
```

Consulta

Maps

```
cout << estados["Mato Grosso"] << "\n";
```

Consulta

Maps

```
cout << estados["Mato Grosso"] << "\n";

cout << estados["Bahia"] << "\n";
```

```
66
75
```

Consulta

Maps

```
if (estados.find("Amazonas") != estados.end()) {

    cout << "Estado encontrado!\n";

}
```

Busca

Sets

```
#include <set>

set< valor >  nome ;
```

Sets

```
#include <set>

set<int > nome;
```

Sets

```
#include <set>

set<int > identificadores;

nome.insert valor;
```

Inserção

Sets

```
#include <set>

set<int > identificadores;

identificadores.insert(68);
```

Inserção

Sets

```
#include <set>

set<int > identificadores;

identificadores.insert(68);

identificadores.insert(33);

identificadores.insert(108);
```

Inserção

Sets

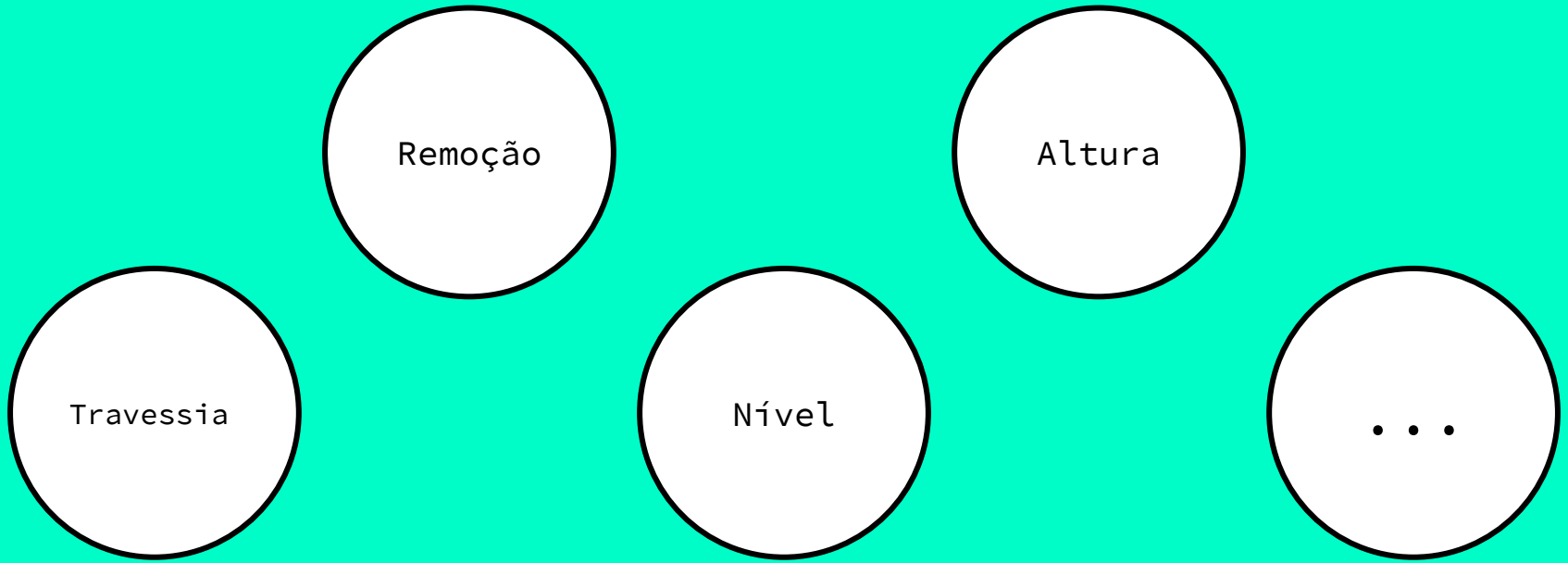
```
if (identificadores.count(108) != 0) {

    cout << "Encontrado!\n";

}
```

Busca

Próxima aula



Operações

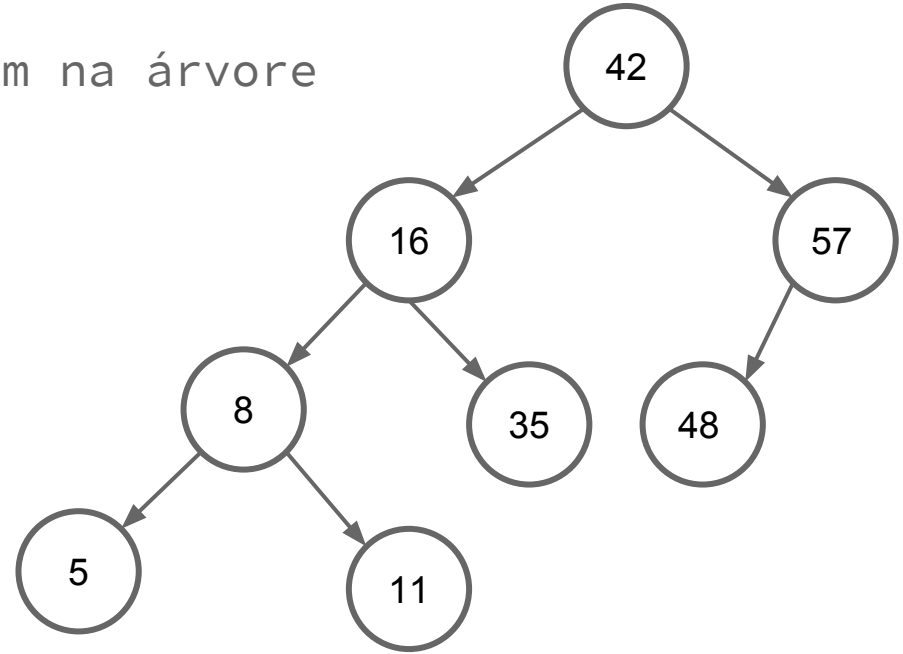
Inserção ✓

Busca ✓

Remoção

Remoção

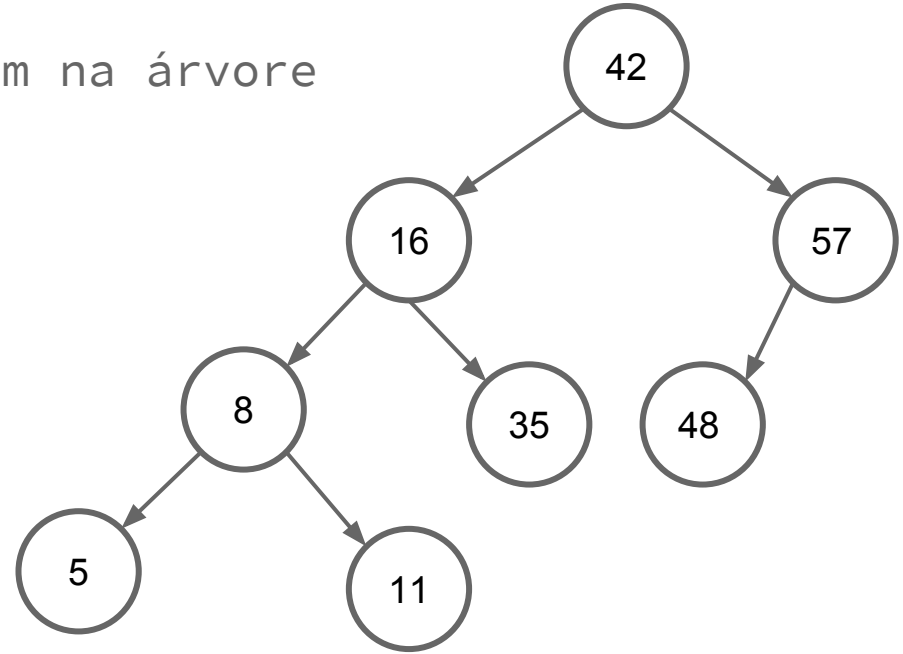
- Remover número que tem na árvore



Remoção

- Remover número que tem na árvore

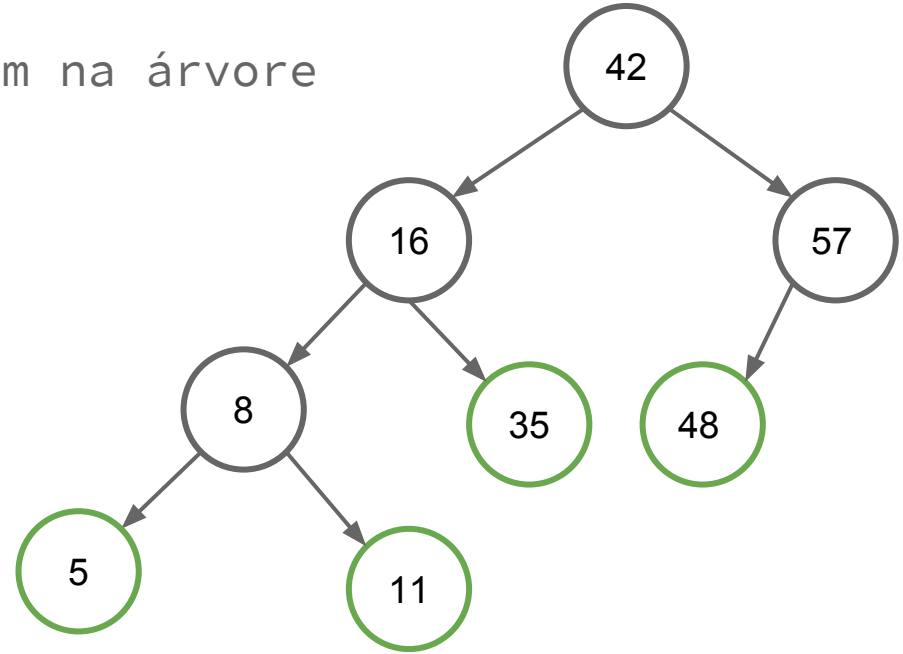
Folha



Remoção

- Remover número que tem na árvore

Folha

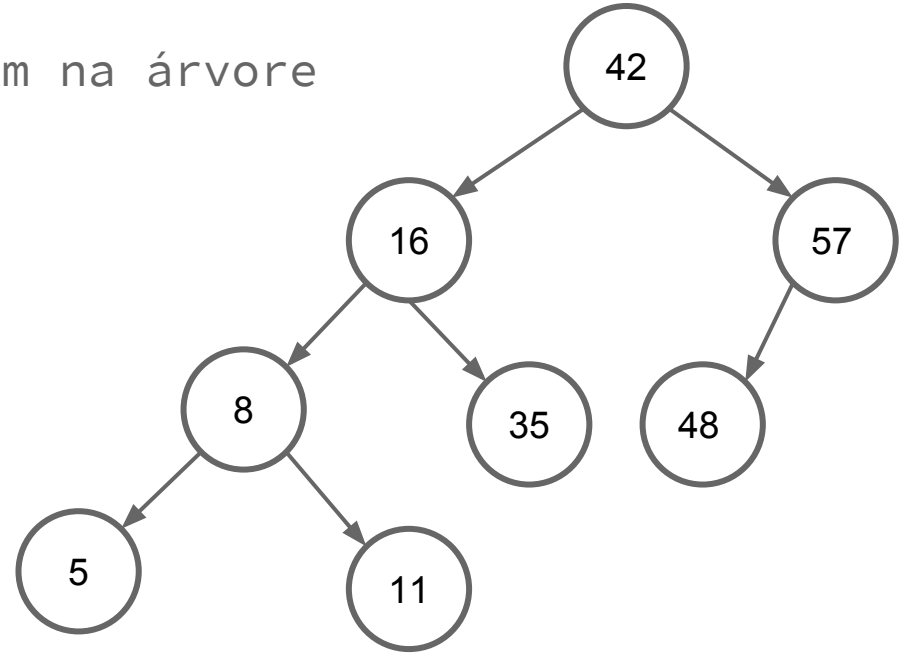


Remoção

- Remover número que tem na árvore

Folha

1 filho

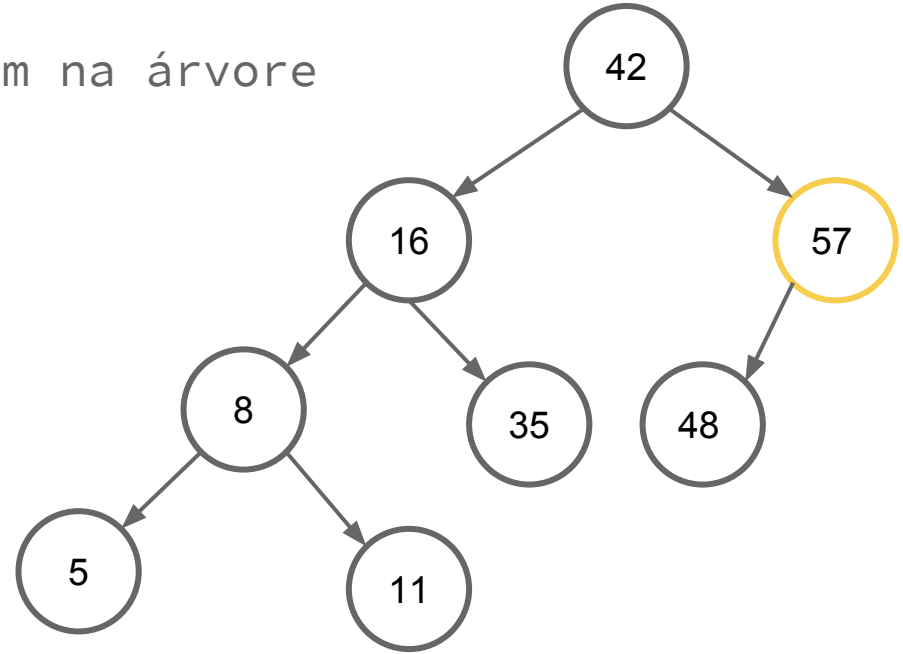


Remoção

- Remover número que tem na árvore

Folha

1 filho



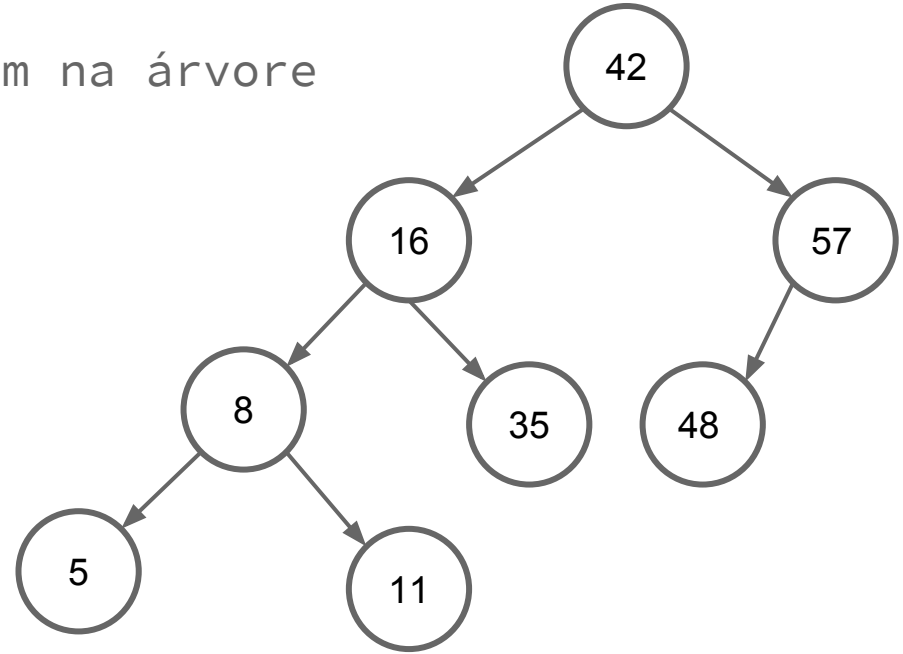
Remoção

- Remover número que tem na árvore

Folha

1 filho

2 filhos



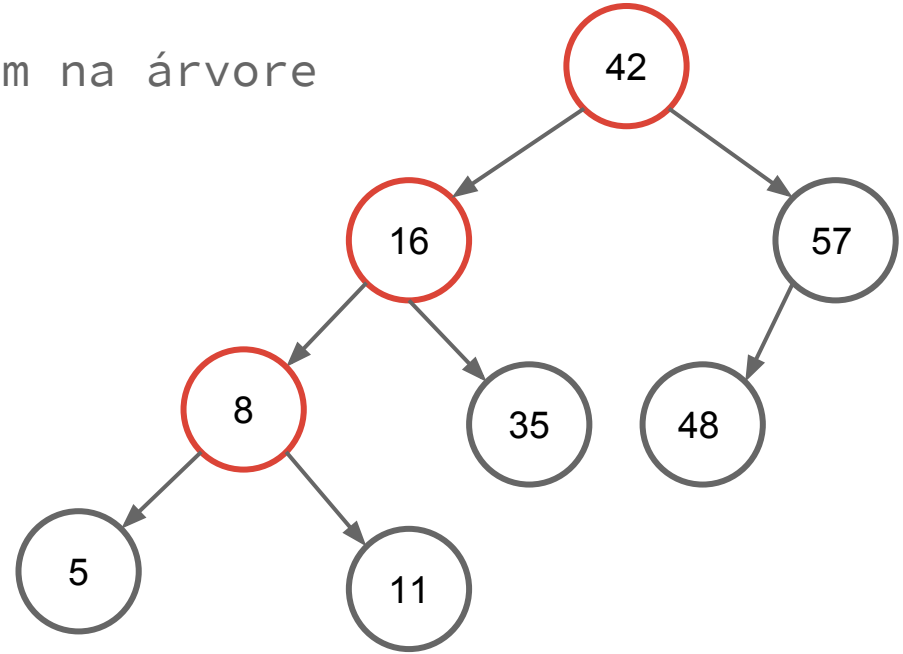
Remoção

- Remover número que tem na árvore

Folha

1 filho

2 filhos



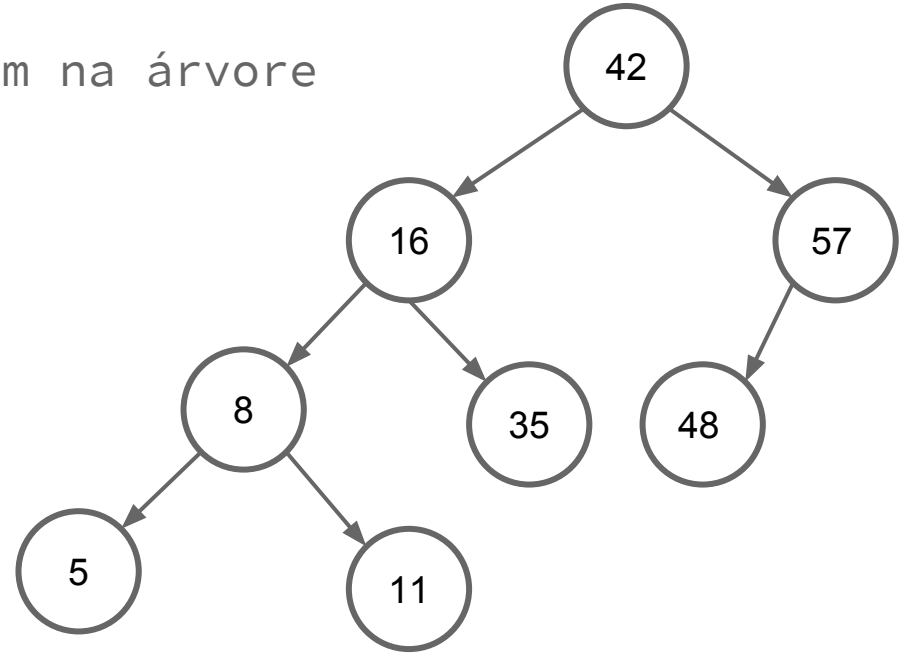
Remoção

- Remover número que tem na árvore

Folha

1 filho

2 filhos



Remoção

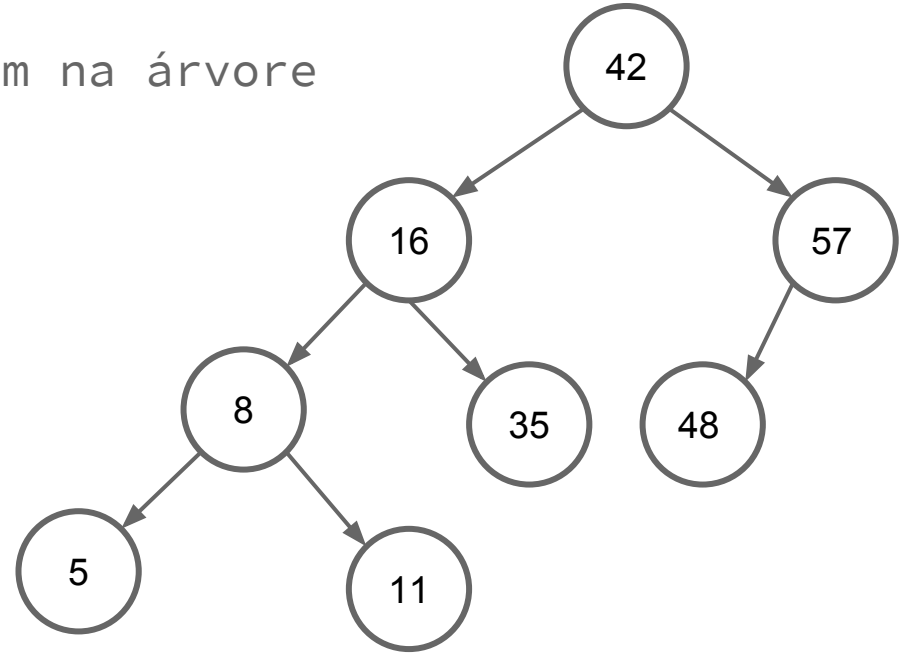
- Remover número que tem na árvore

Folha

11

1 filho

2 filhos



Remoção

- Remover número que tem na árvore

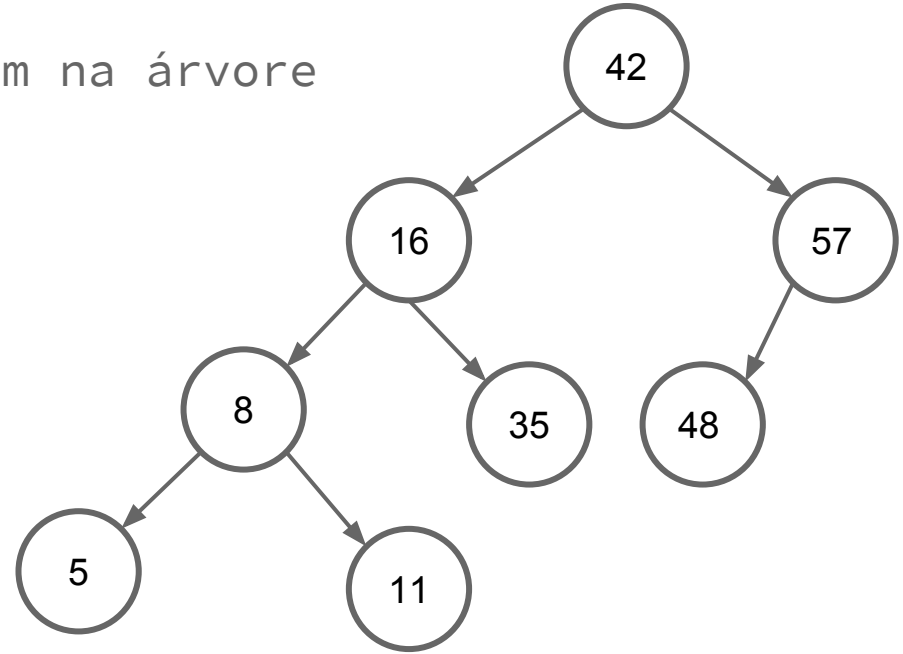
Folha

1 filho

2 filhos

11

Buscar



Remoção

- Remover número que tem na árvore

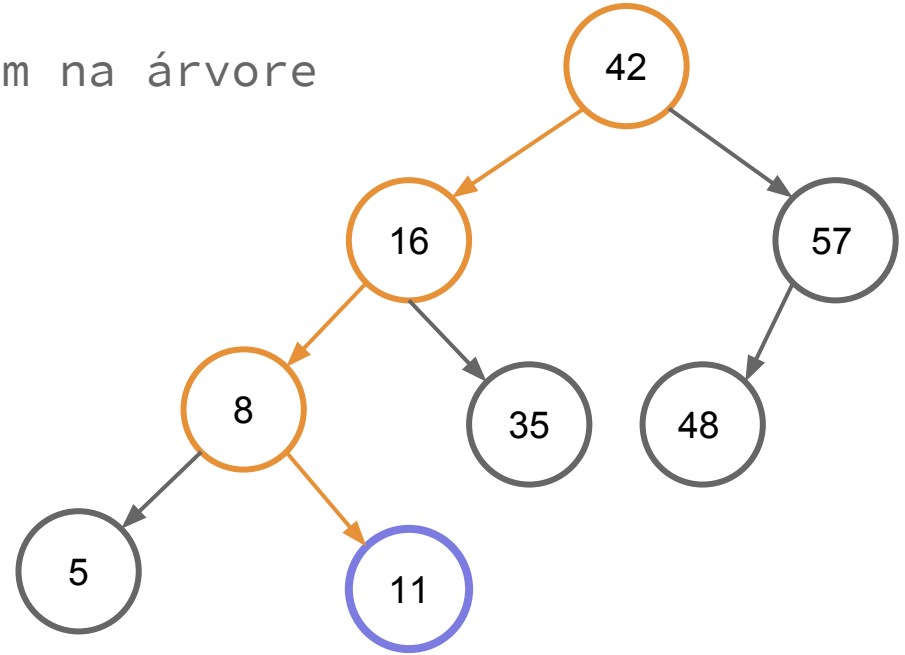
Folha

1 filho

2 filhos

11

Achou



Remoção

- Remover número que tem na árvore

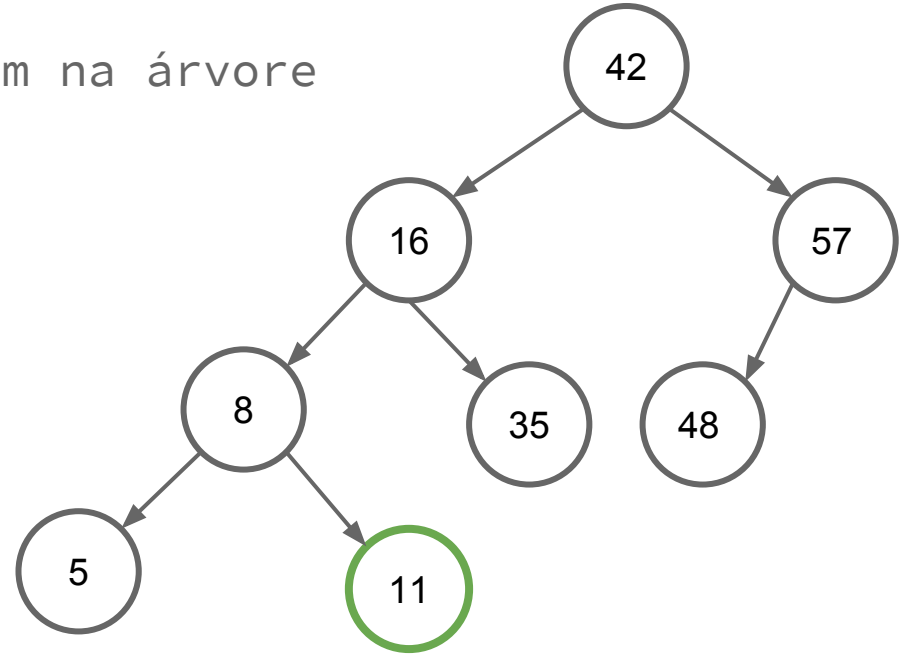
Folha

1 filho

2 filhos

11

É folha?



Remoção

- Remover número que tem na árvore

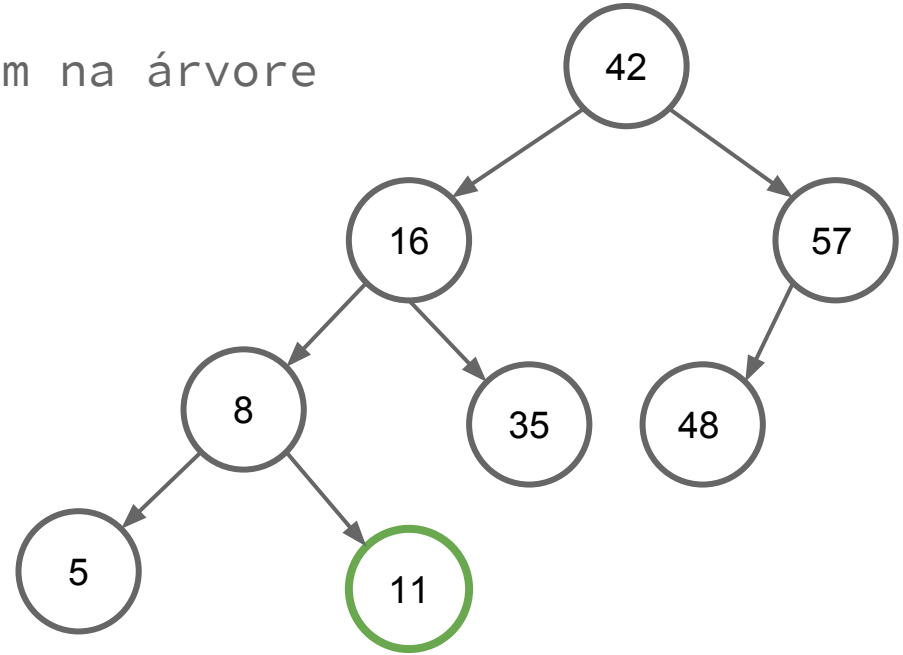
Folha

1 filho

2 filhos

11

É folha?
Sim



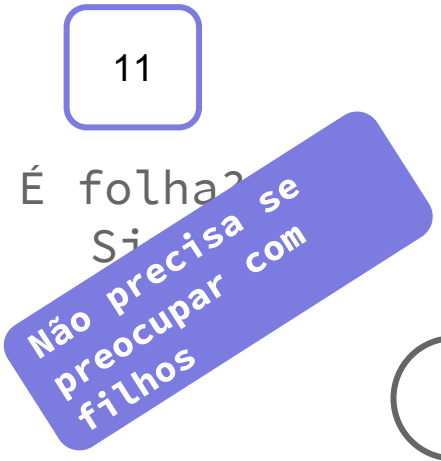
Remoção

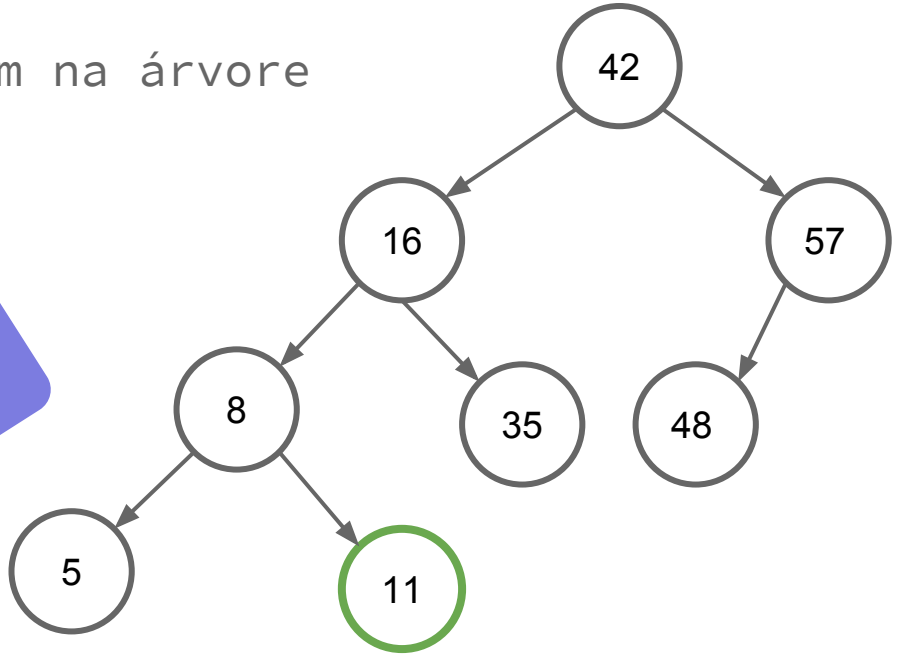
- Remover número que tem na árvore

Folha

1 filho

2 filhos





Remoção

- Remover número que tem na árvore

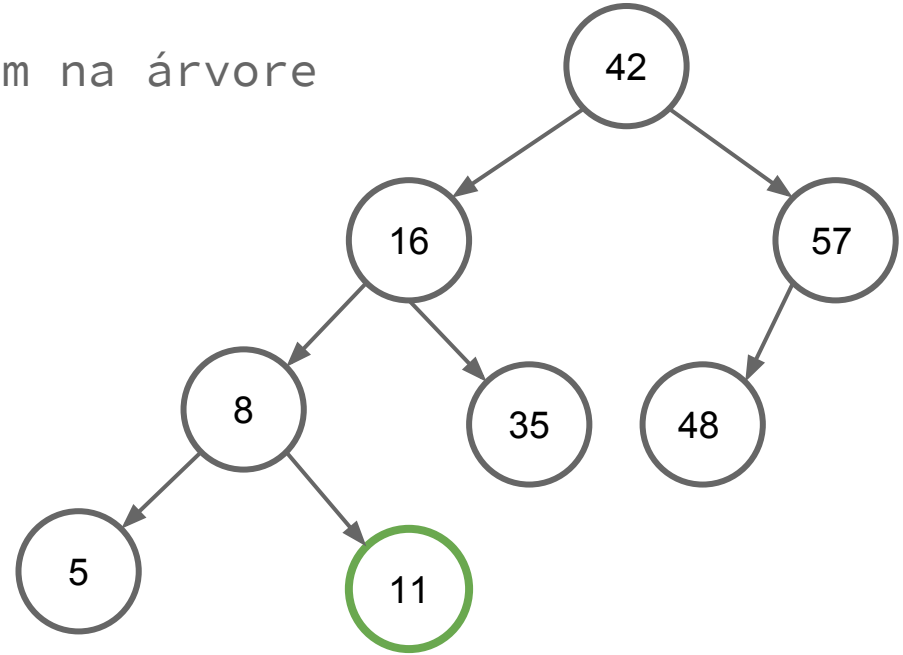
Folha

11

E o pai?

1 filho

2 filhos



Remoção

- Remover número que tem na árvore

Folha

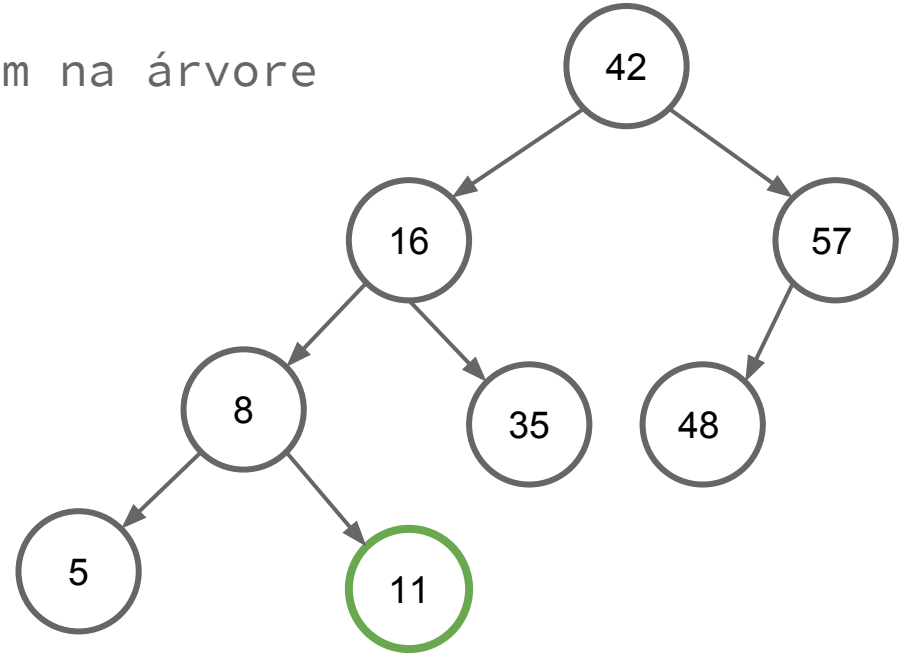
1 filho

2 filhos

11

E o pai?

Perde o filho direito



Remoção

- Remover número que tem na árvore

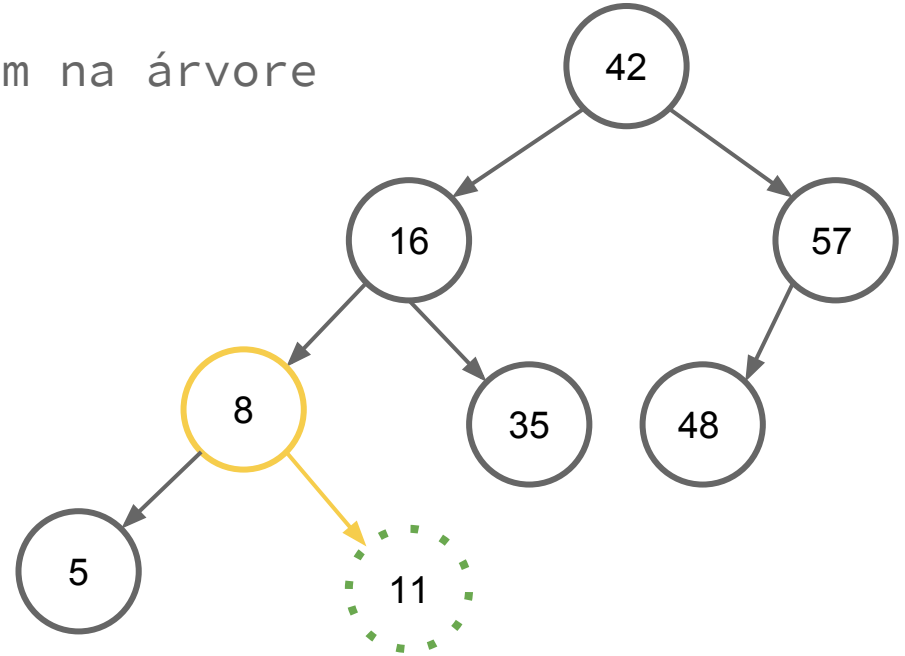
Folha

11

E o pai?
Perde o filho direito

1 filho

2 filhos



Remoção

- Remover número que tem na árvore

Folha

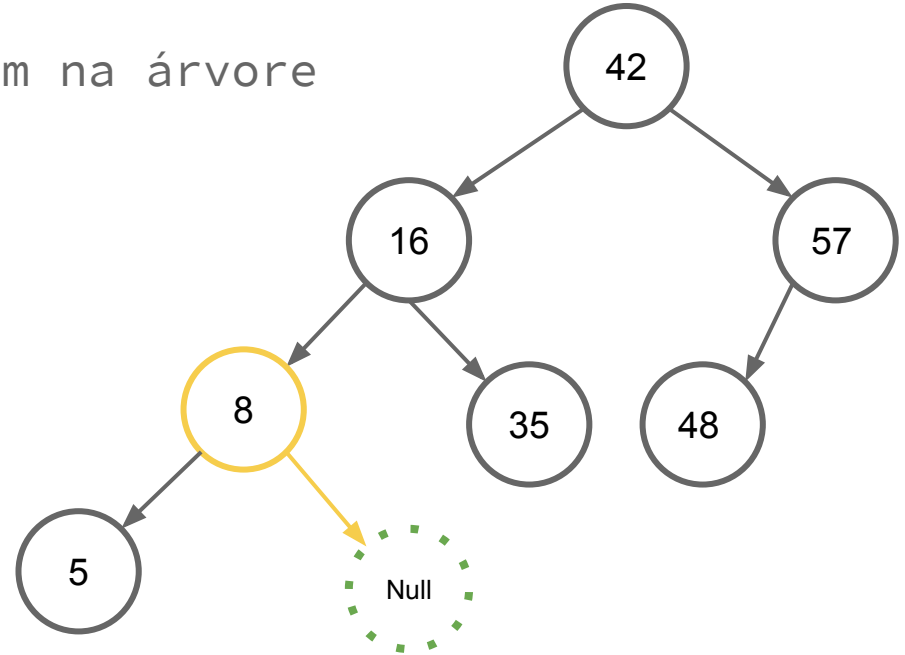
1 filho

2 filhos

11

E o pai?

Perde o filho direito



Remoção

- Remover número que tem na árvore

Folha

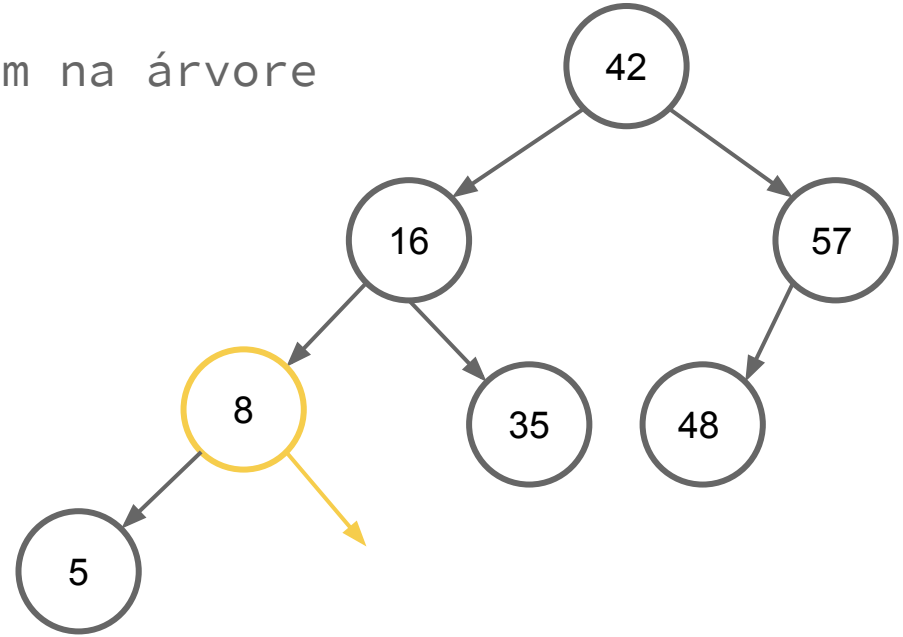
1 filho

2 filhos

11

E o pai?

Perde o filho direito



Remoção

- Remover número que tem na árvore

Folha

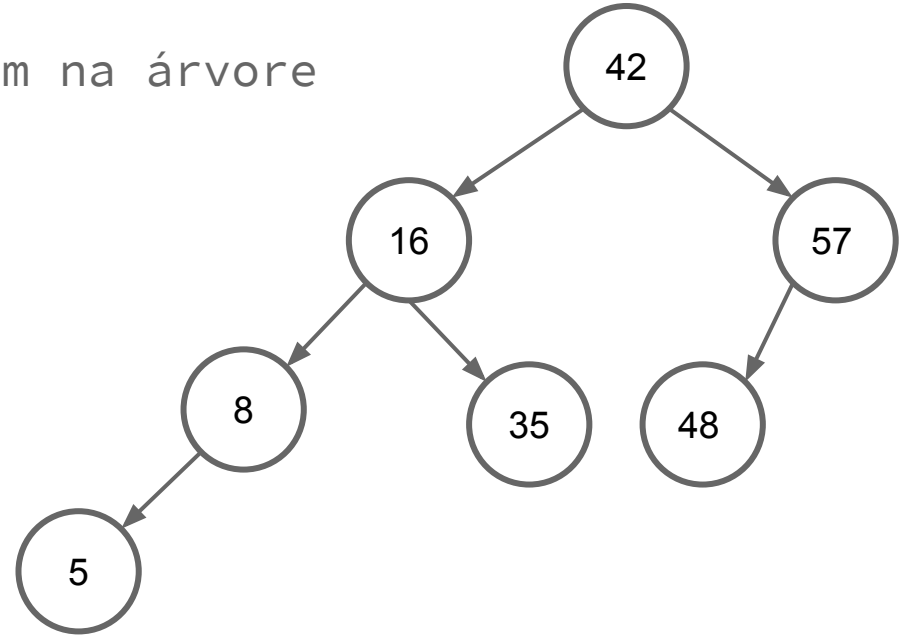
1 filho

2 filhos

11

E o pai?

Perde o filho direito



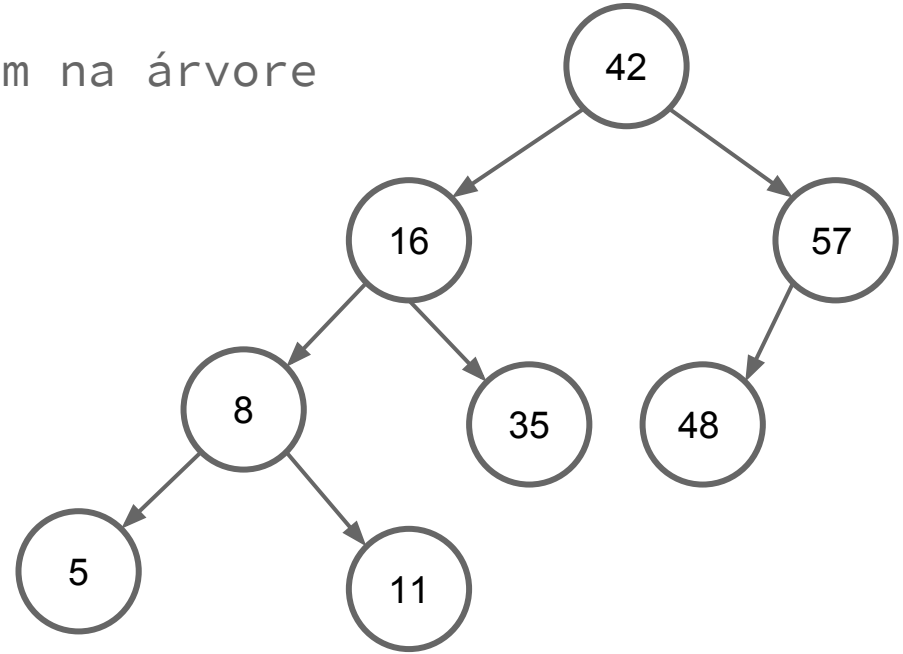
Remoção

- Remover número que tem na árvore

Folha

1 filho

2 filhos



Remover folha

```
1)  no = raiz, valor_remocao = 48
2)  no_remocao = busca(no, valor_remocao)
3)  SE no_remocao.esq == null && no_remocao.dir == null:
    a)  SE no_remocao.pai.valor > no_remocao:
        i)  no_remocao.pai.esq = null
    b)  SE no_remocao.pai.valor < no_remocao:
        i)  no_remocao.pai.dir = null
```

Remover folha

```
1)  no = raiz, valor_remocao = 48
2)  no_remocao = busca(no, valor_remocao)
3)  SE no_remocao.esq == null && no_remocao.dir == null:
    a)  SE no_remocao.pai.valor > no_remocao:
        i)  no_remocao.pai.esq = null
    b)  SE no_remocao.pai.valor < no_remocao:
        i)  no_remocao.pai.dir = null
```

Busca

Remover folha

```
1)  no = raiz, valor_remocao = 48
2)  no_remocao = busca(no, valor_remocao)
3)  SE no_remocao.esq == null && no_remocao.dir == null:
    a)  SE no_remocao.pai.valor > no_remocao:
        i)  no_remocao.pai.esq = null
    b)  SE no_remocao.pai.valor < no_remocao:
        i)  no_remocao.pai.dir = null
```

Busca

Acesso ao pai

Remover folha

```
1)  no = raiz, valor_remocao = 48
2)  no_remocao = busca(no, valor_remocao)
3)  SE no_remocao.esq == null && no_remocao.dir == null:
    a)  SE no_remocao.pai.valor > no_remocao:
        i)  no_remocao.pai.esq = null
    b)  SE no_remocao.pai.valor < no_remocao:
        i)  no_remocao.pai.dir = null
```

Busca

Acesso ao pai

O(log n)

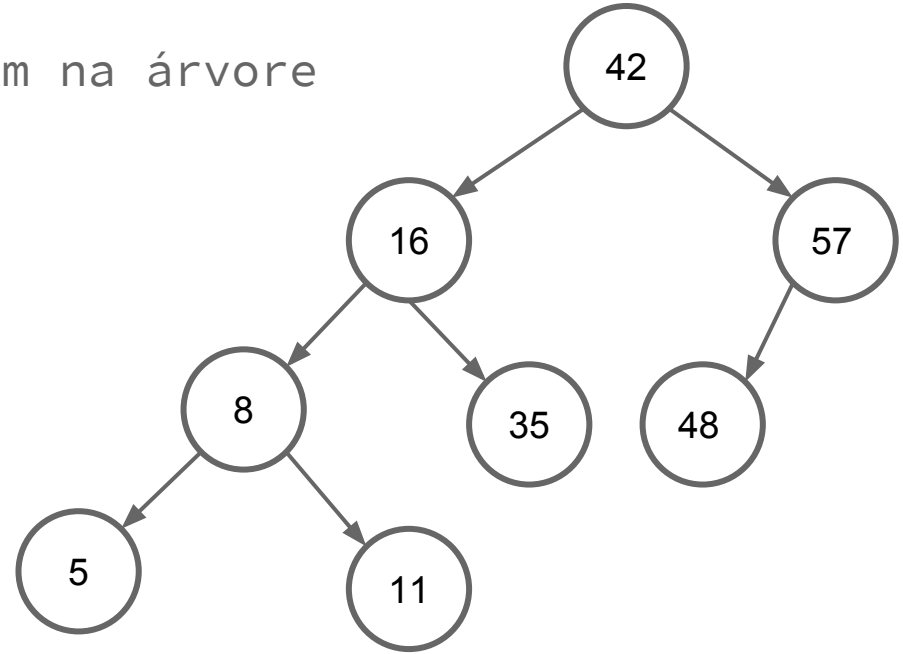
Remoção

- Remover número que tem na árvore

Folha

1 filho

2 filhos



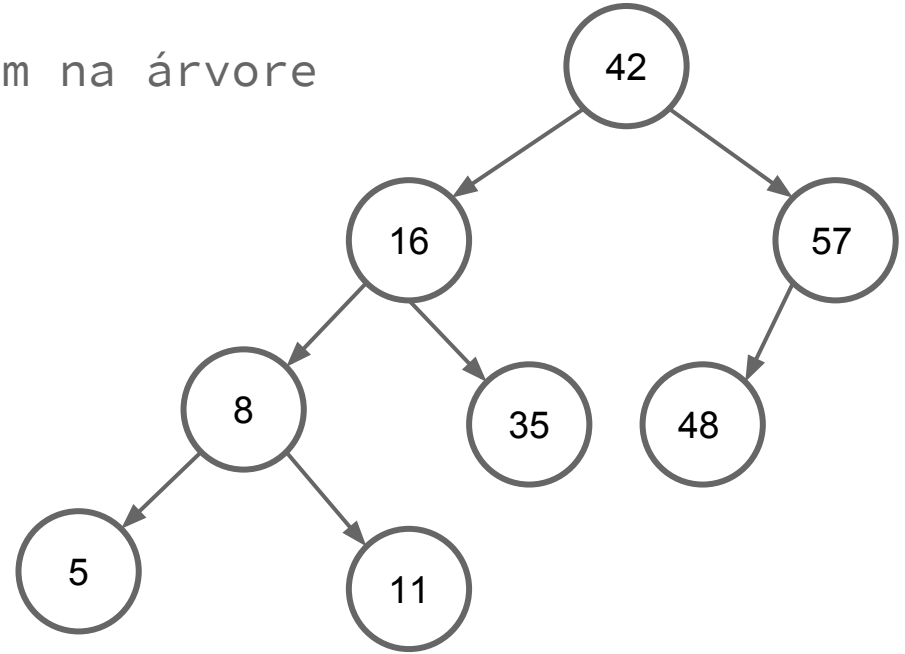
Remoção

- Remover número que tem na árvore

Folha

1 filho

2 filhos



Remoção

- Remover número que tem na árvore

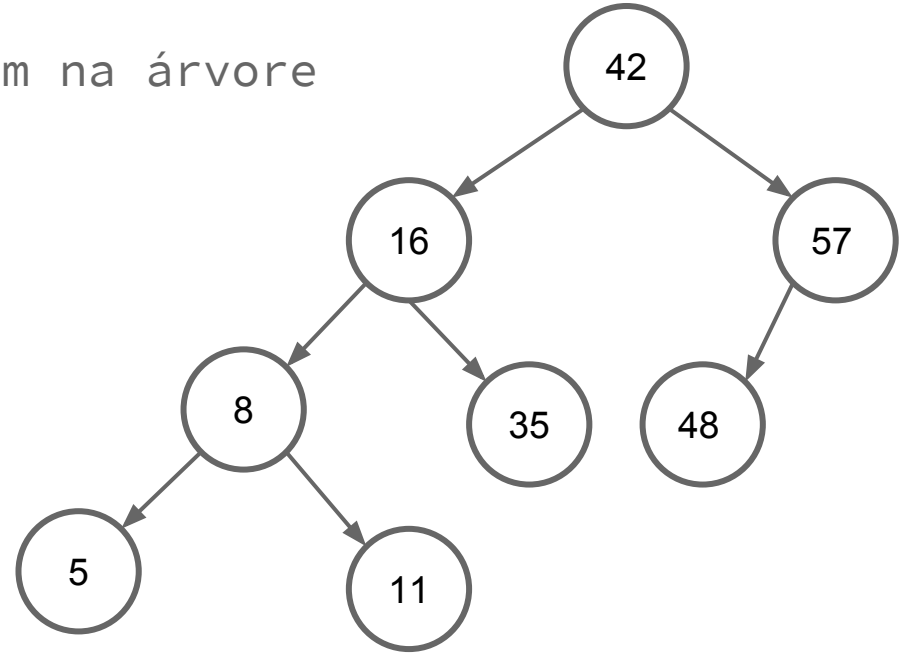
Folha

1 filho

2 filhos

57

Buscar



Remoção

- Remover número que tem na árvore

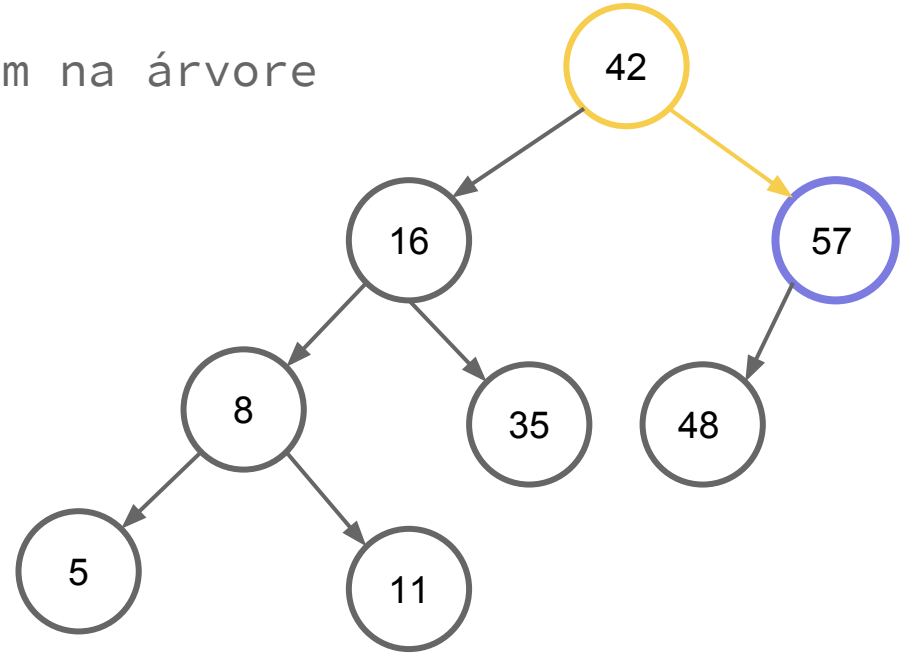
Folha

1 filho

2 filhos

57

Achou



Remoção

- Remover número que tem na árvore

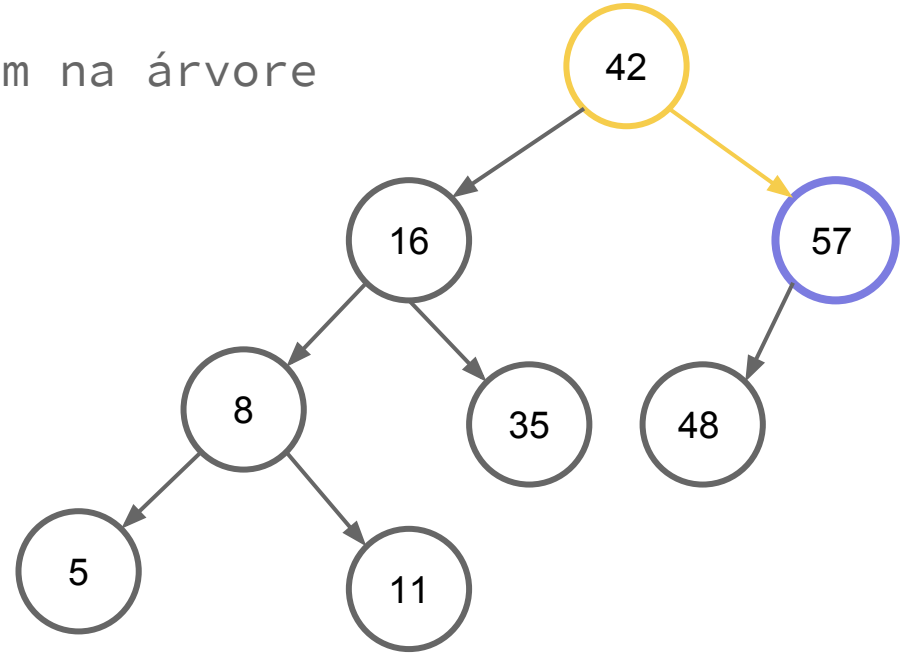
Folha

1 filho

2 filhos

57

Tem filho?



Remoção

- Remover número que tem na árvore

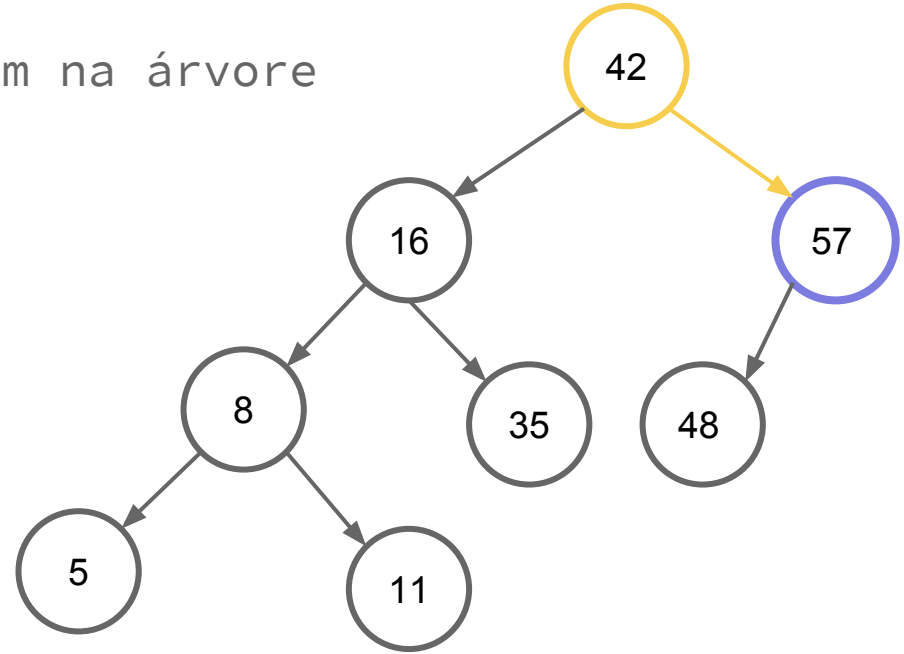
Folha

1 filho

2 filhos

57

Tem filho?
Sim



Remoção

- Remover número que tem na árvore

Folha

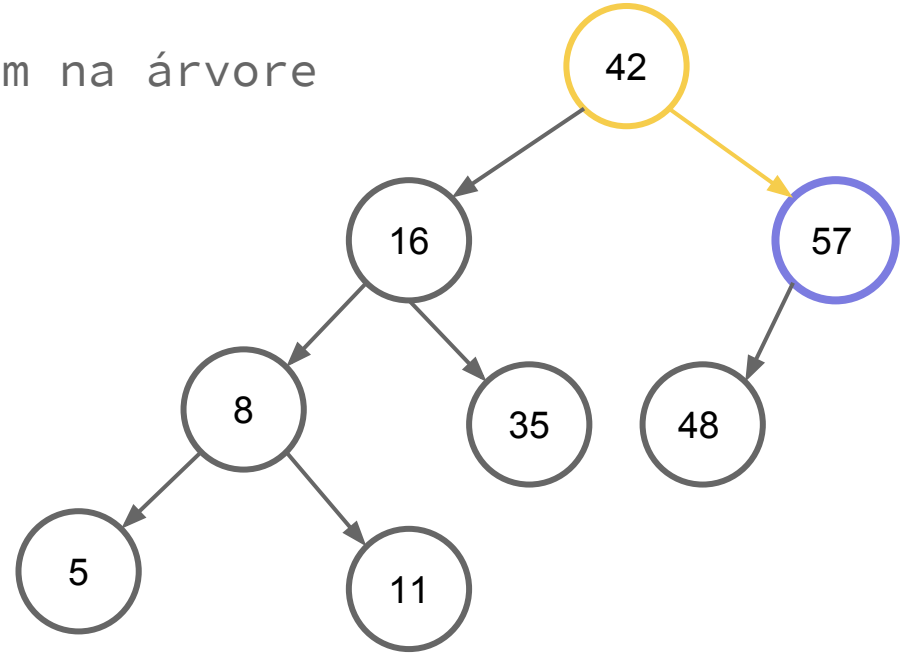
1 filho

2 filhos

57

Tem filho?

Sim



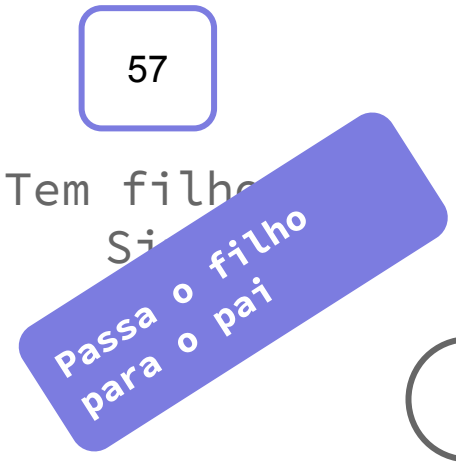
Remoção

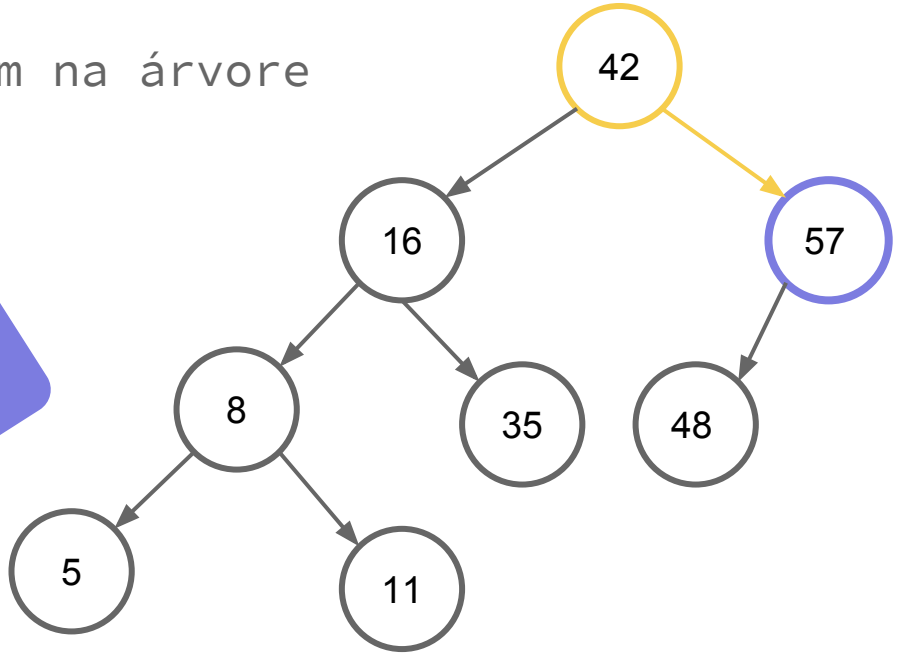
- Remover número que tem na árvore

Folha

1 filho

2 filhos





Remoção

- Remover número que tem na árvore

Folha

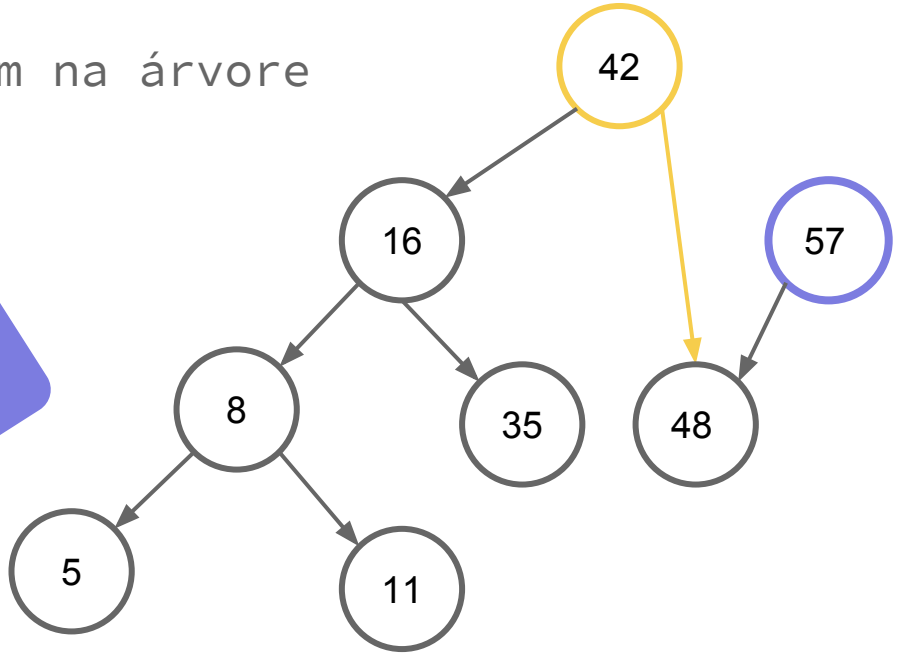
1 filho

2 filhos

57

Tem filho
Si

Passa o filho para o pai



Remoção

- Remover número que tem na árvore

Folha

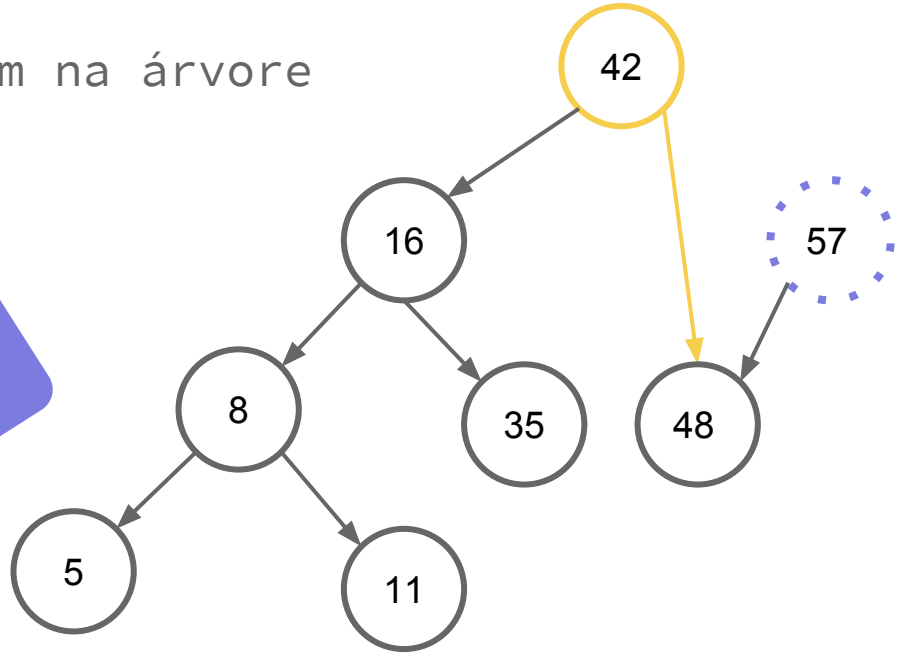
1 filho

2 filhos

57

Tem filho
Si

Passa o filho para o pai



Remoção

- Remover número que tem na árvore

Folha

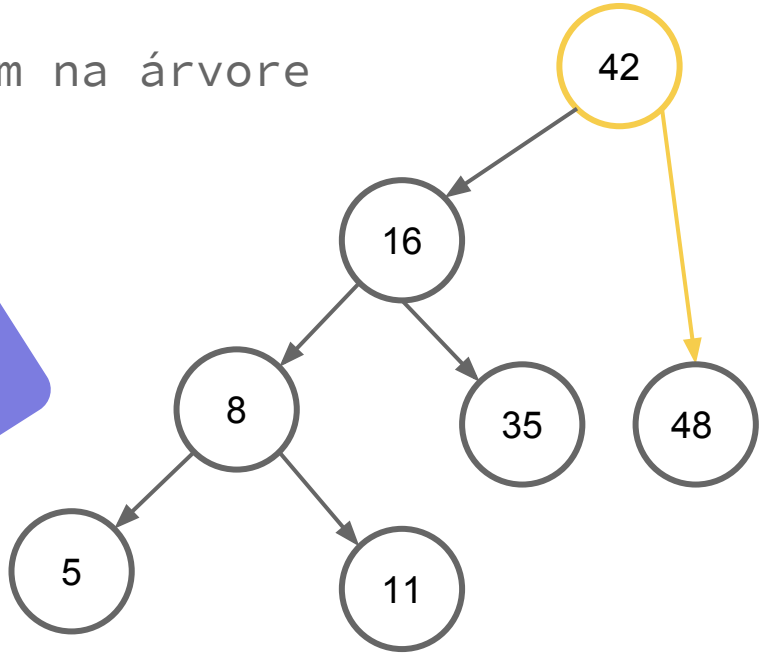
1 filho

2 filhos

57

Tem filho
Si

Passa o filho para o pai



Remoção

- Remover número que tem na árvore

Folha

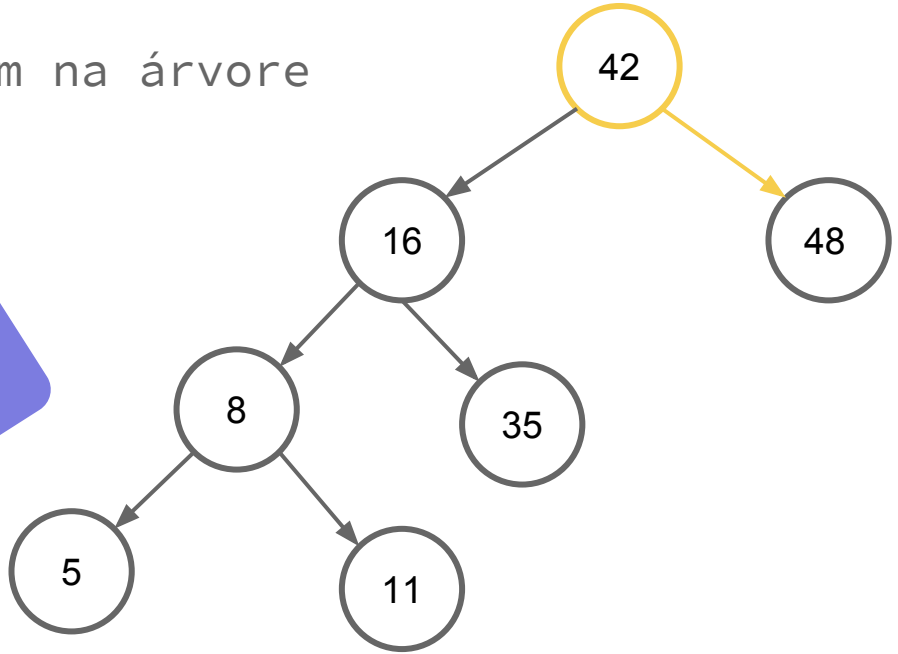
1 filho

2 filhos

57

Tem filho
Si

Passa o filho para o pai



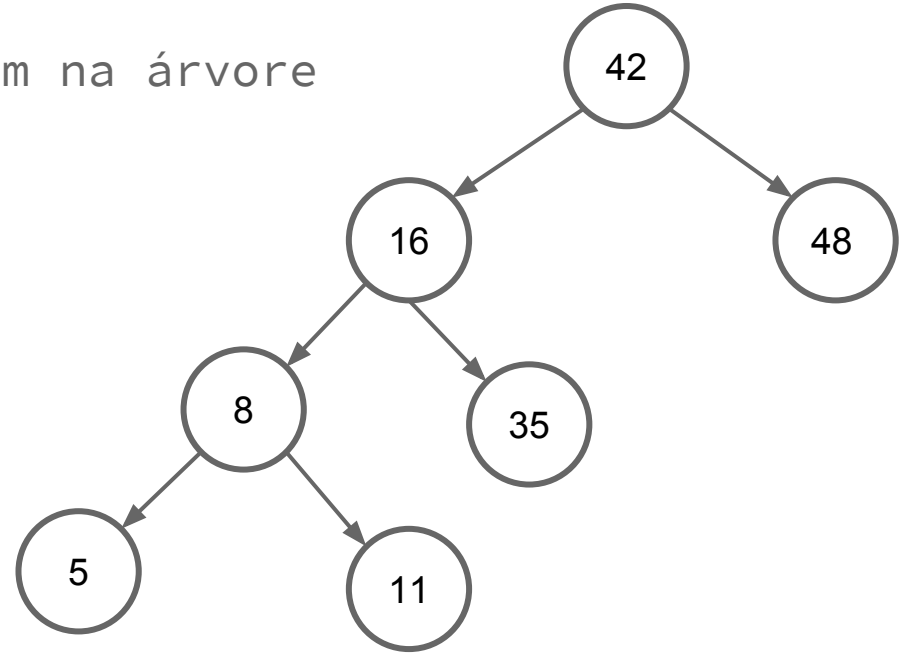
Remoção

- Remover número que tem na árvore

Folha

1 filho

2 filhos



Remover com 1 filho

1) no = raiz, valor_remocao = 48
2) no_remocao = busca(no, valor_remocao)
3) SE no_remocao.esq != null && no_remocao.dir == null:
    a) SE no_remocao.pai.valor > no_remocao:
        i) no_remocao.pai.esq = no_remocao.esq
    b) SE no_remocao.pai.valor < no_remocao:
        i) no_remocao.pai.dir = no_remocao.esq
4) SE no_remocao.esq == null && no_remocao.dir != null:
    a) SE no_remocao.pai.valor > no_remocao:
        i) no_remocao.pai.esq = no_remocao.dir
    b) SE no_remocao.pai.valor < no_remocao:
        i) no_remocao.pai.dir = no_remocao.dir

Remover com 1 filho

```
1)  no = raiz, valor_remocao = 48
2)  no_remocao = busca(no, valor_remocao)
3)  SE no_remocao.esq != null && no_remocao.dir == null:
    a)  SE no_remocao.pai.valor > no_remocao:
        i)  no_remocao.pai.esq = no_remocao.esq
    b)  SE no_remocao.pai.valor < no_remocao:
        i)  no_remocao.pai.dir = no_remocao.esq
4)  SE no_remocao.esq == null && no_remocao.dir != null:
    a)  SE no_remocao.pai.valor > no_remocao:
        i)  no_remocao.pai.esq = no_remocao.dir
    b)  SE no_remocao.pai.valor < no_remocao:
        i)  no_remocao.pai.dir = no_remocao.dir
```

Remover com 1 filho

```
1)  no = raiz, valor_remocao = 48
2)  no_remocao = busca(no, valor_remocao)
3)  SE no_remocao.esq != null && no_remocao.dir == null:
    a)  SE no_remocao.pai.valor > no_remocao:
        i)  no_remocao.pai.esq = no_remocao.esq
    b)  SE no_remocao.pai.valor < no_remocao:
        i)  no_remocao.pai.dir = no_remocao.esq
4)  SE no_remocao.esq == null && no_remocao.dir != null:
    a)  SE no_remocao.pai.valor > no_remocao:
        i)  no_remocao.pai.esq = no_remocao.dir
    b)  SE no_remocao.pai.valor < no_remocao:
        i)  no_remocao.pai.dir = no_remocao.dir
```

Busca

Acesso ao pai

Remover com 1 filho

```
1)  no = raiz, valor_remocao = 48
2)  no_remocao = busca(no, valor_remocao)
3)  SE no_remocao.esq != null && no_remocao.dir == null:
    a)  SE no_remocao.pai.valor > no_remocao:
        i)  no_remocao.pai.esq = no_remocao.esq
    b)  SE no_remocao.pai.valor < no_remocao:
        i)  no_remocao.pai.dir = no_remocao.esq
4)  SE no_remocao.esq == null && no_remocao.dir != null:
    a)  SE no_remocao.pai.valor > no_remocao:
        i)  no_remocao.pai.esq = no_remocao.dir
    b)  SE no_remocao.pai.valor < no_remocao:
        i)  no_remocao.pai.dir = no_remocao.dir
```

O(log n)

Busca

Acesso ao pai

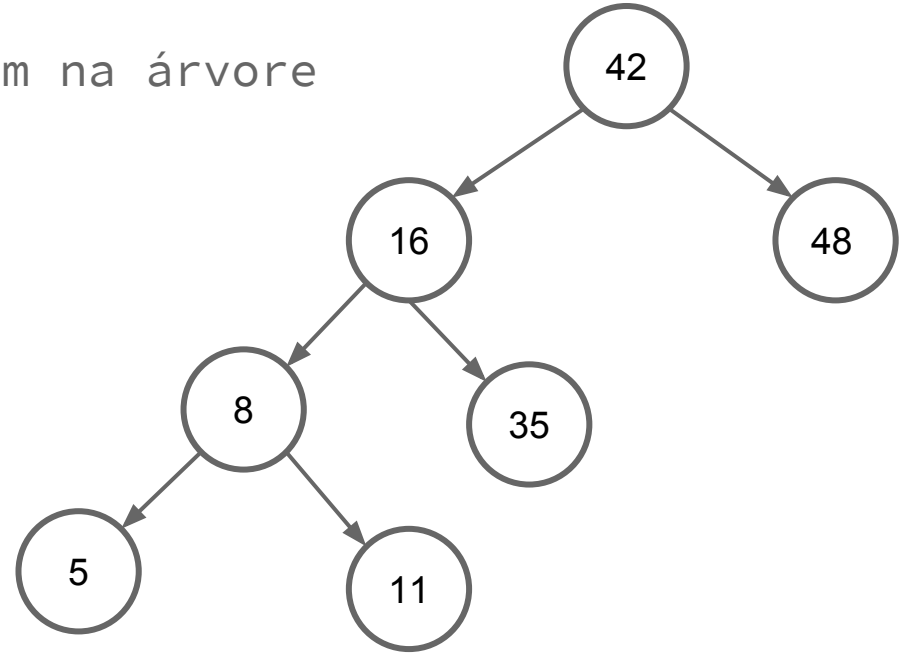
Remoção

- Remover número que tem na árvore

Folha

1 filho

2 filhos



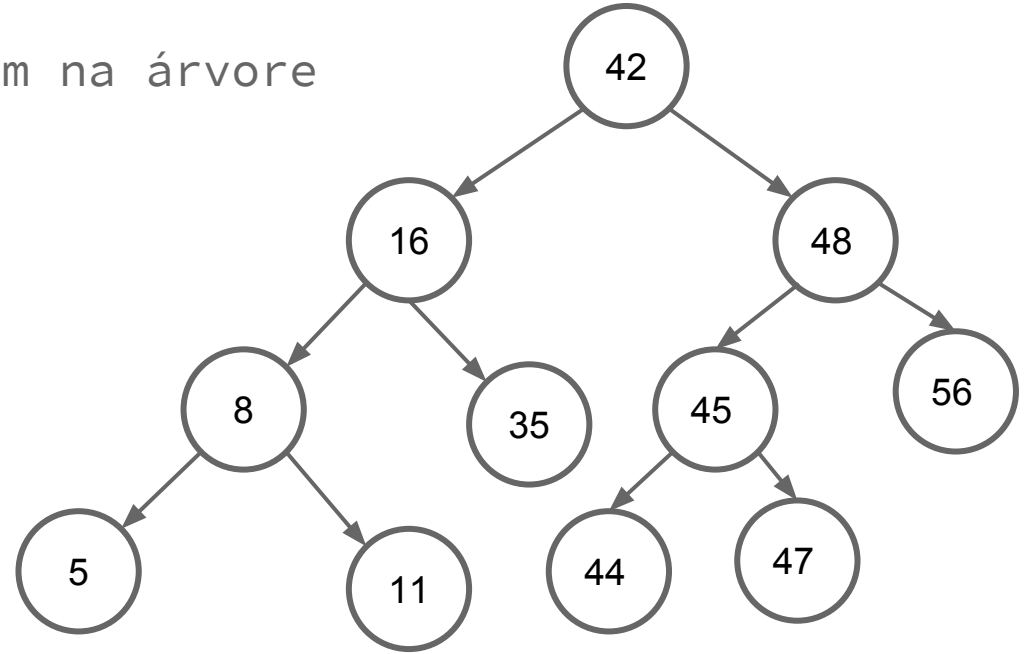
Remoção

- Remover número que tem na árvore

Folha

1 filho

2 filhos



Remoção

- Remover número que tem na árvore

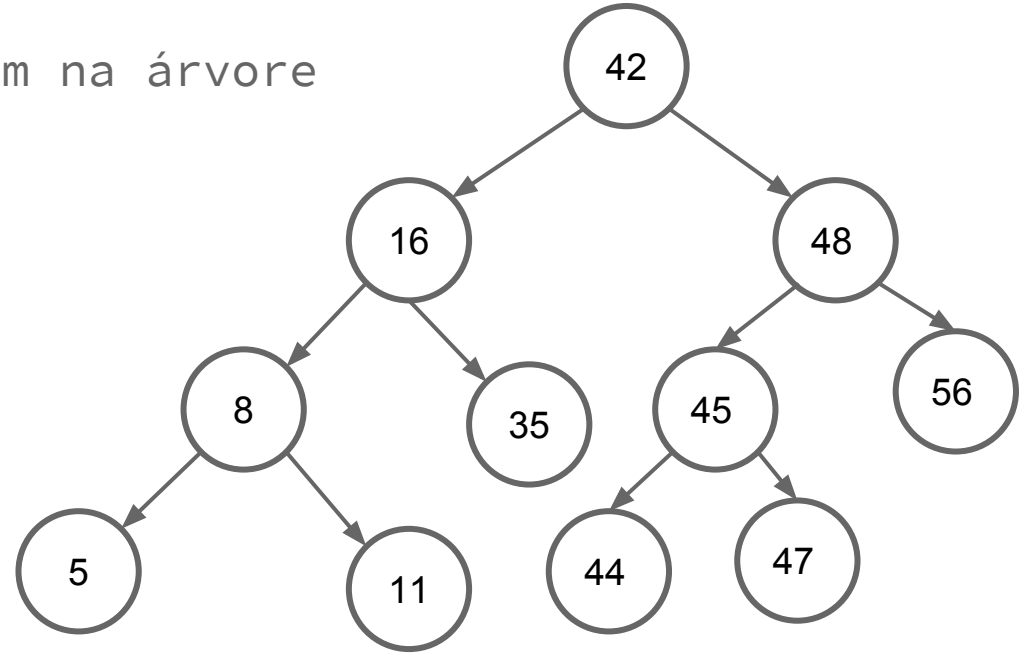
Folha

1 filho

2 filhos

42

Buscar



Remoção

- Remover número que tem na árvore

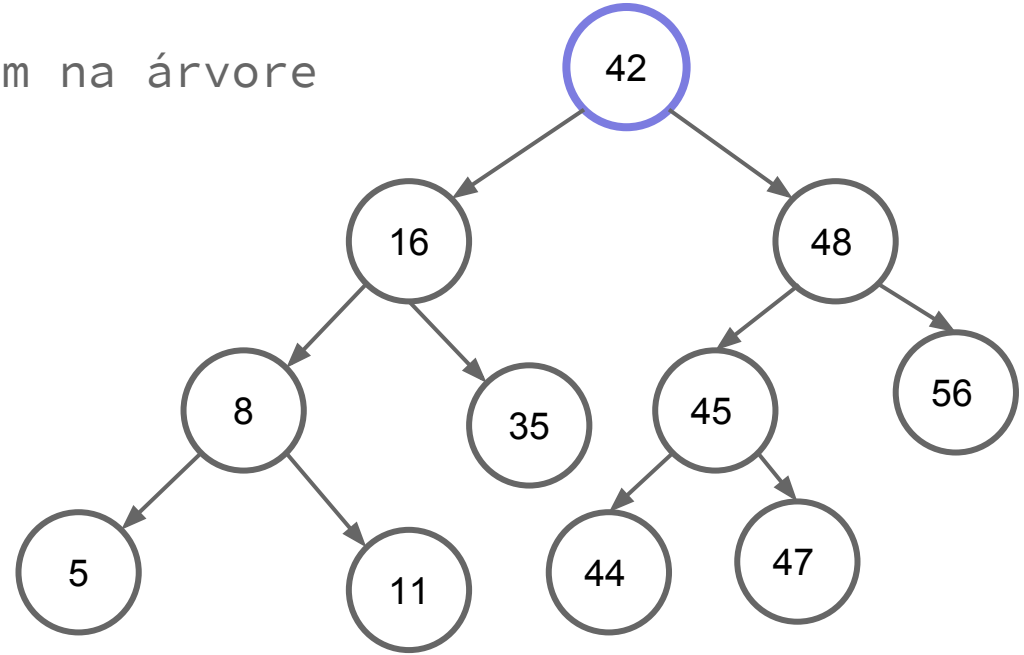
Folha

1 filho

2 filhos

42

Achou



Remoção

- Remover número que tem na árvore

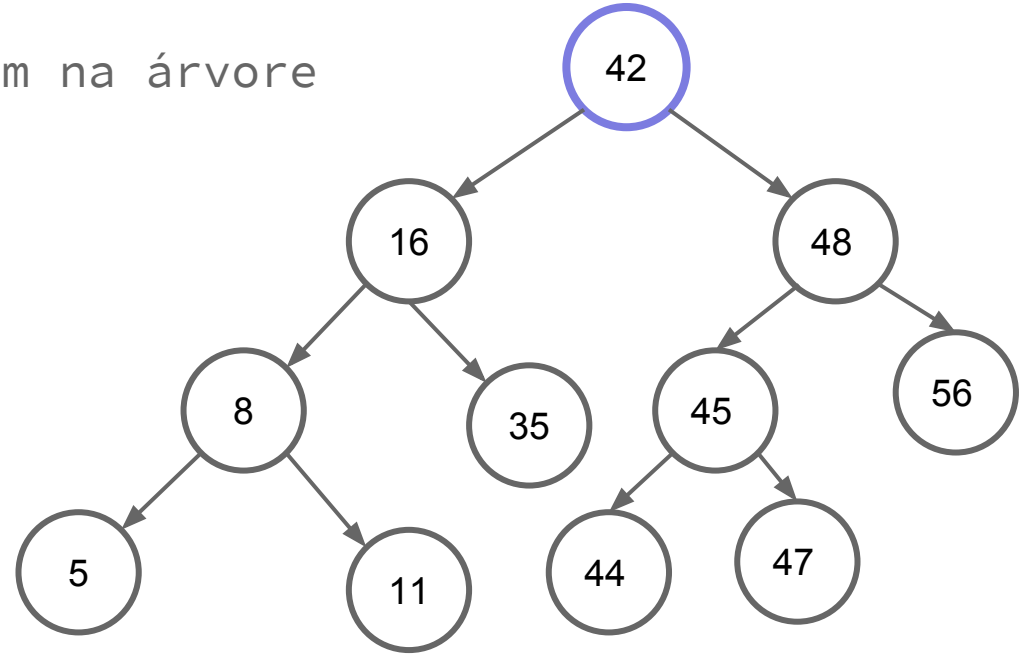
Folha

1 filho

2 filhos

42

Tem filho?
Sim



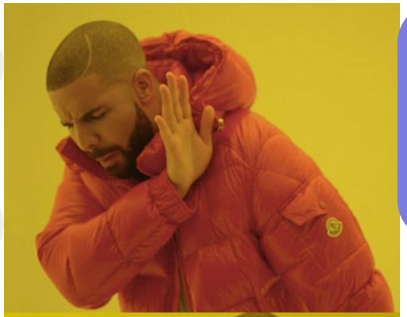
Remoção

- Remover número que tem na árvore

Folha

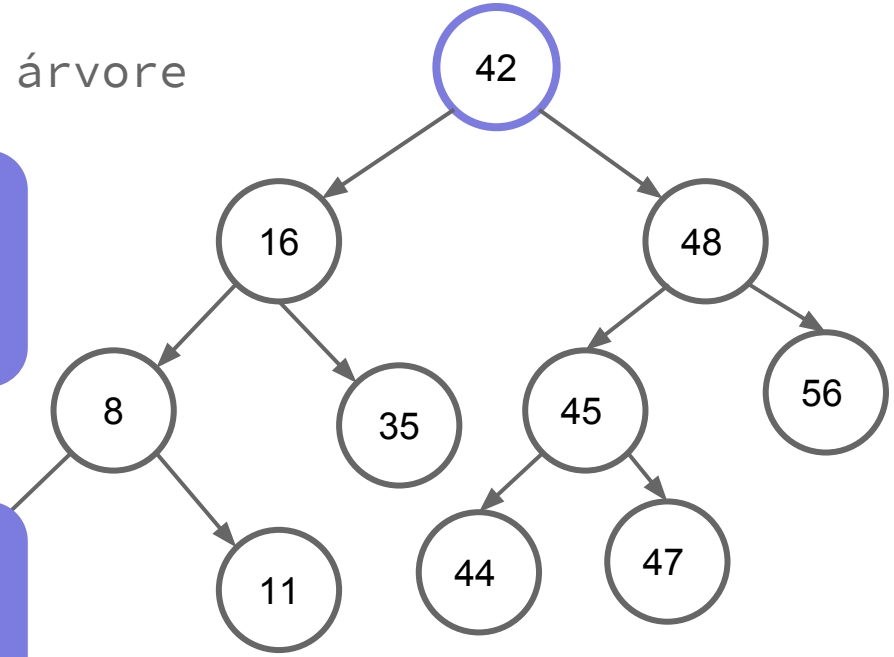
1 filho

2 filhos



Passa o filho para o pai

Acha o sucessor



Remoção

- Remover número que tem na árvore

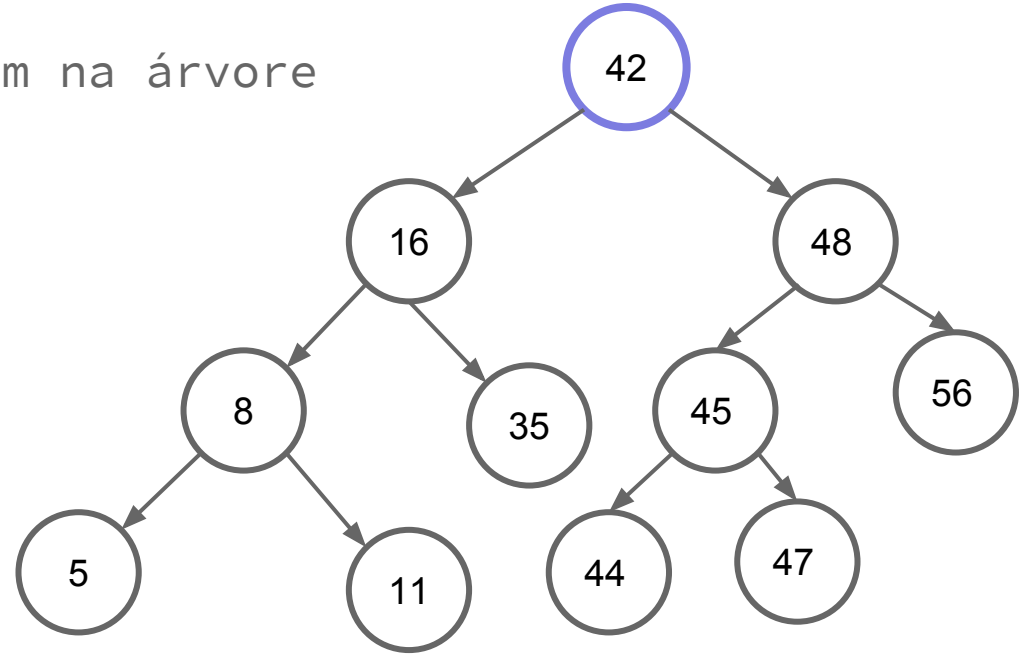
Folha

1 filho

2 filhos

42

Quem é o sucessor?



Remoção

- Remover número que tem na árvore

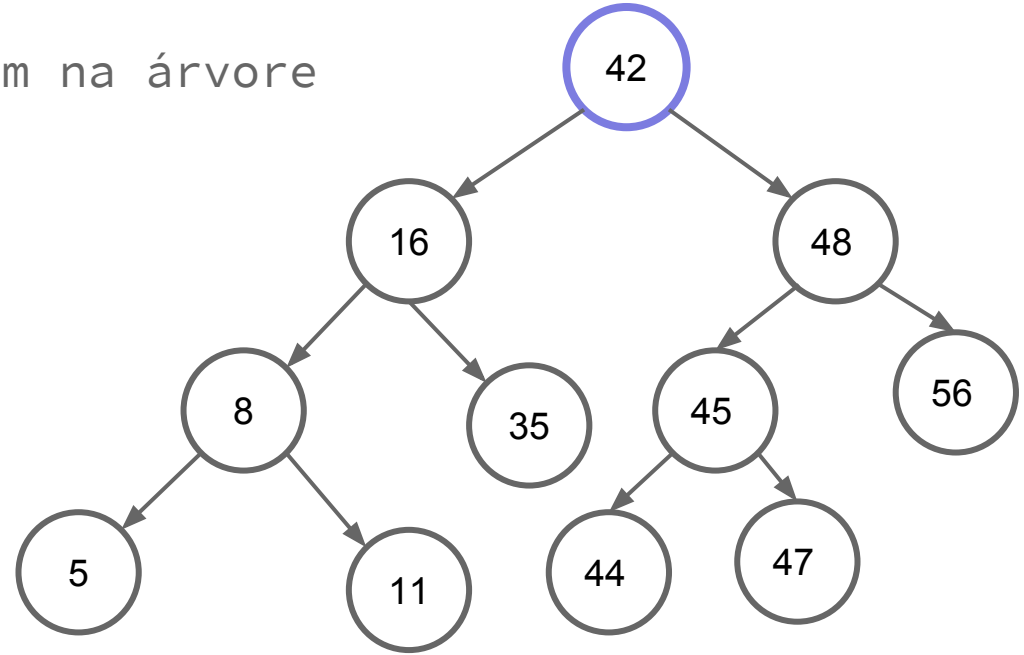
Folha

1 filho

2 filhos

42

Quem é o sucessor?
44



Remoção

- Remover número que tem na árvore

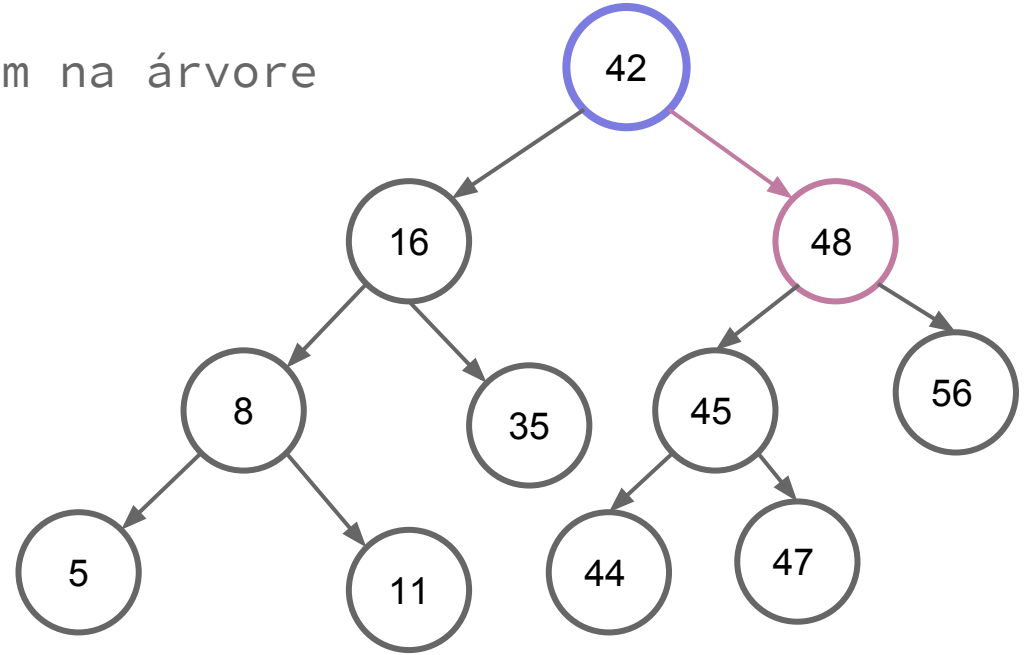
Folha

1 filho

2 filhos

42

Quem é o sucessor?
44



Remoção

- Remover número que tem na árvore

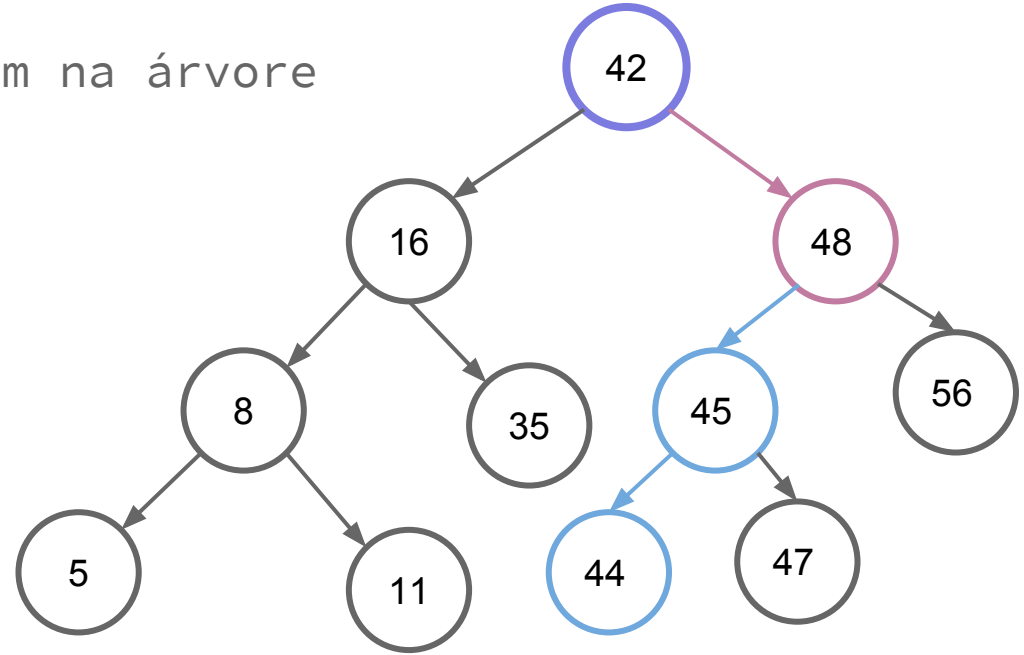
Folha

1 filho

2 filhos

42

Quem é o sucessor?
44



Remoção

- Remover número que tem na árvore

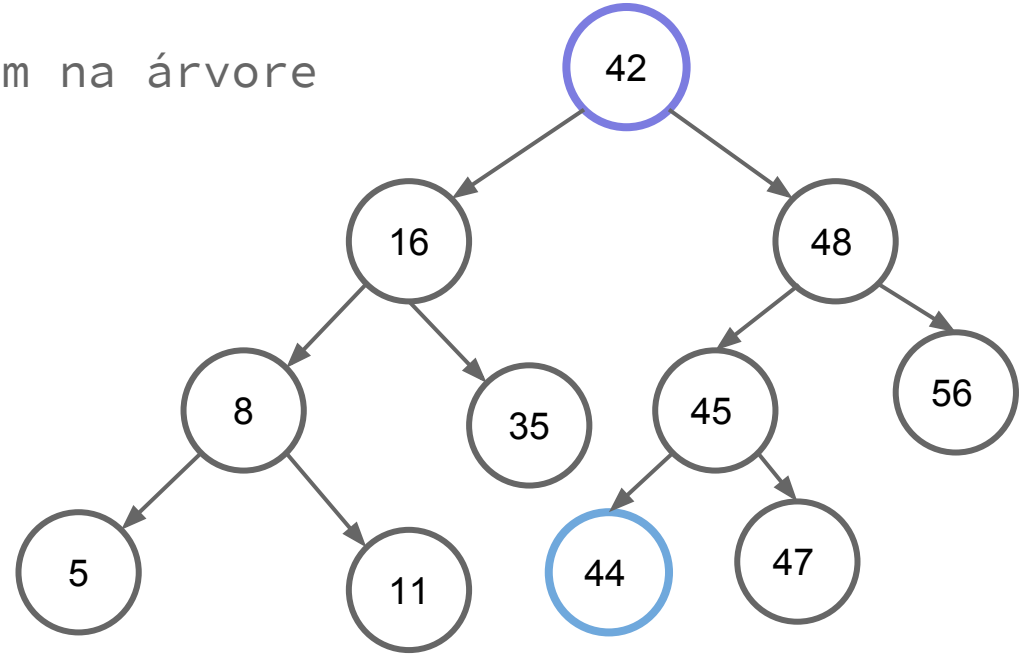
Folha

1 filho

2 filhos

42

Quem é o sucessor?
44



Remoção

- Remover número que tem na árvore

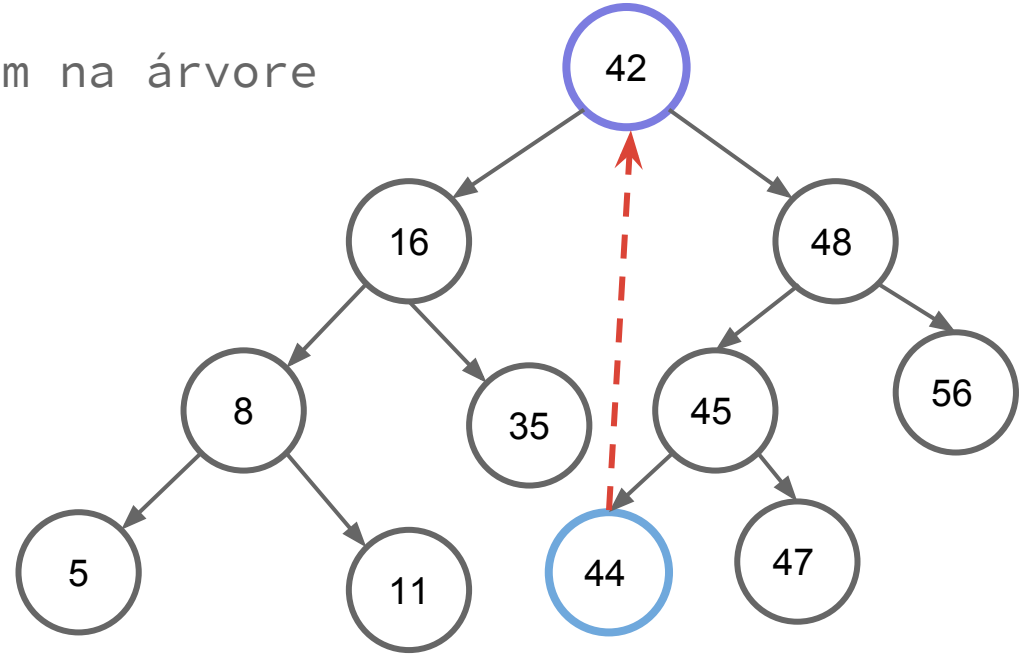
Folha

1 filho

2 filhos

42

Quem é o sucessor?
44



Remoção

- Remover número que tem na árvore

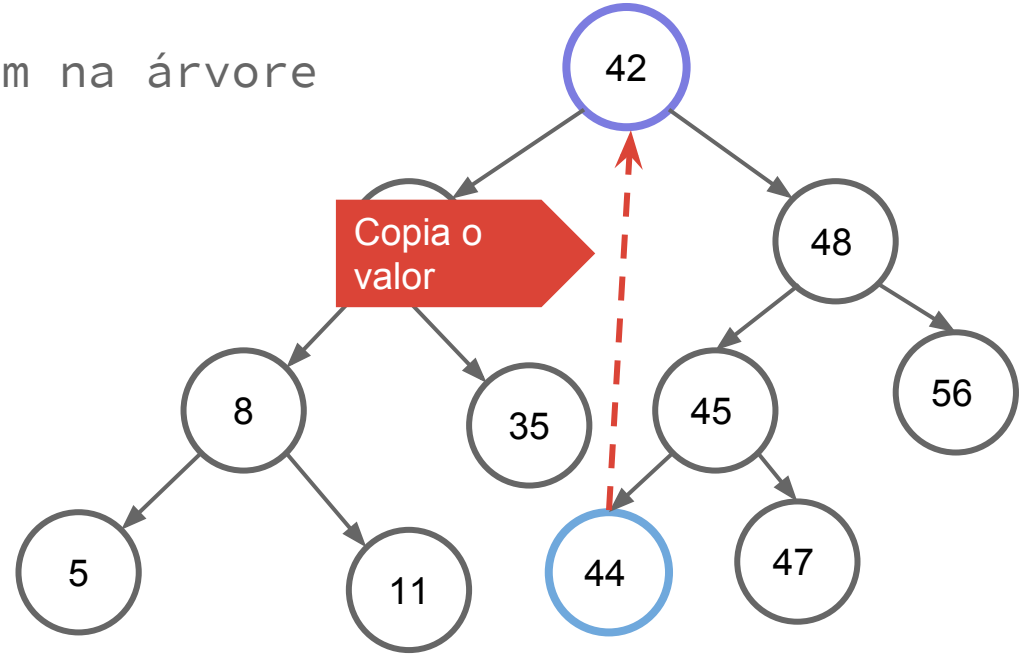
Folha

1 filho

2 filhos

42

Quem é o sucessor?
44



Remoção

- Remover número que tem na árvore

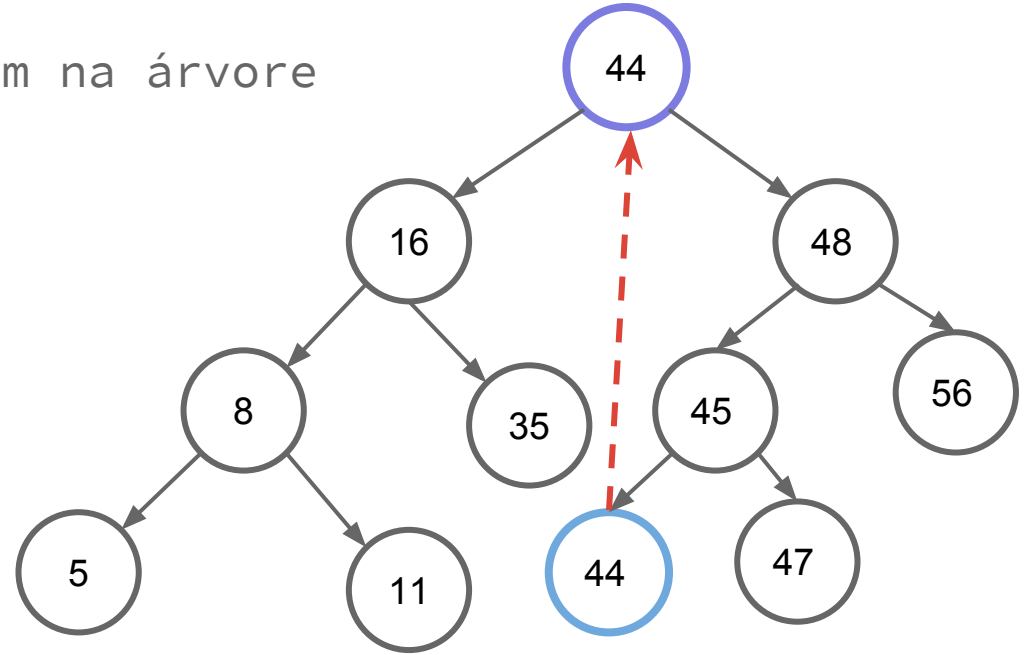
Folha

1 filho

2 filhos

42

Quem é o sucessor?
44



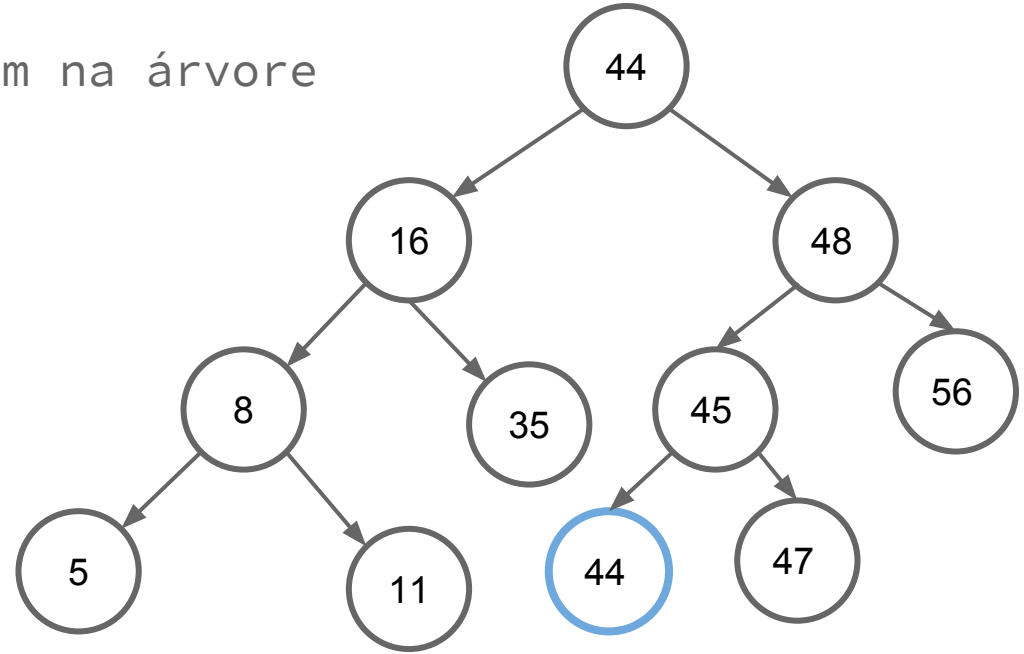
Remoção

- Remover número que tem na árvore

Folha

1 filho

2 filhos



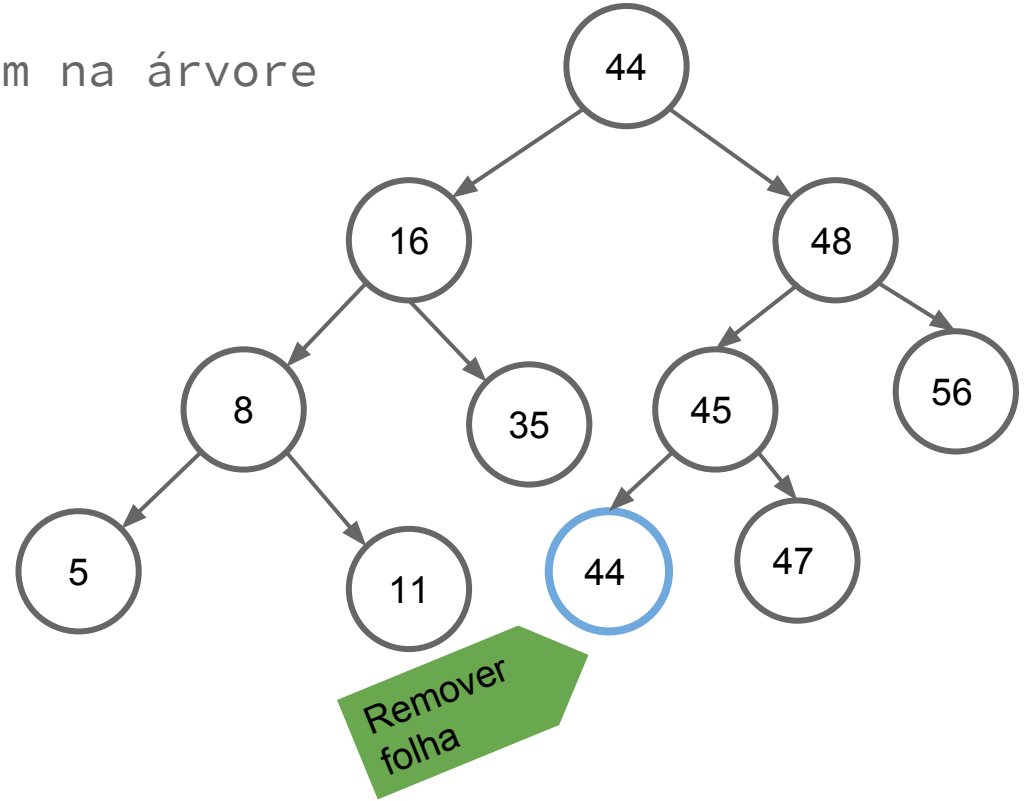
Remoção

- Remover número que tem na árvore

Folha

1 filho

2 filhos



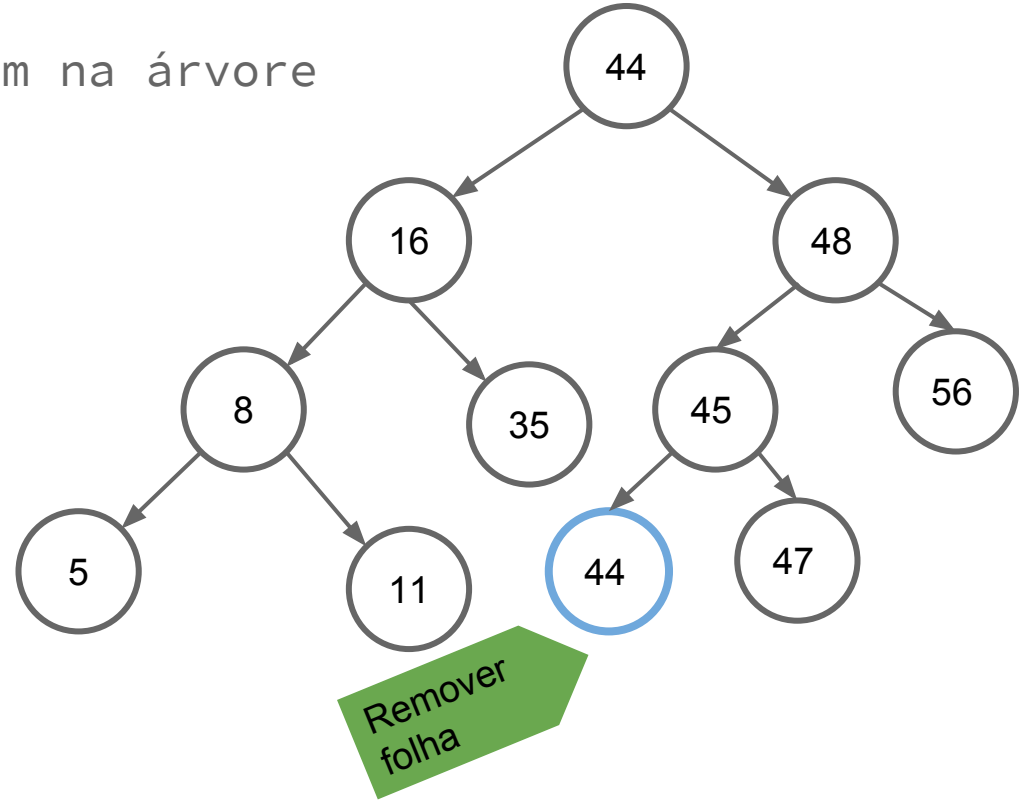
Remoção

- Remover número que tem na árvore

Folha

1 filho

2 filhos



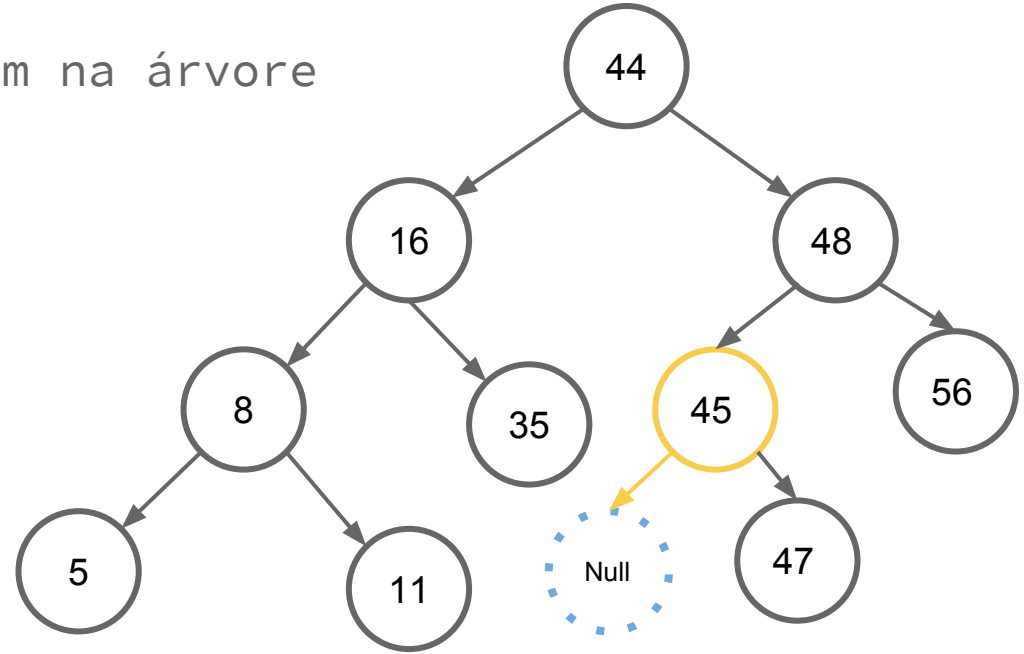
Remoção

- Remover número que tem na árvore

Folha

1 filho

2 filhos



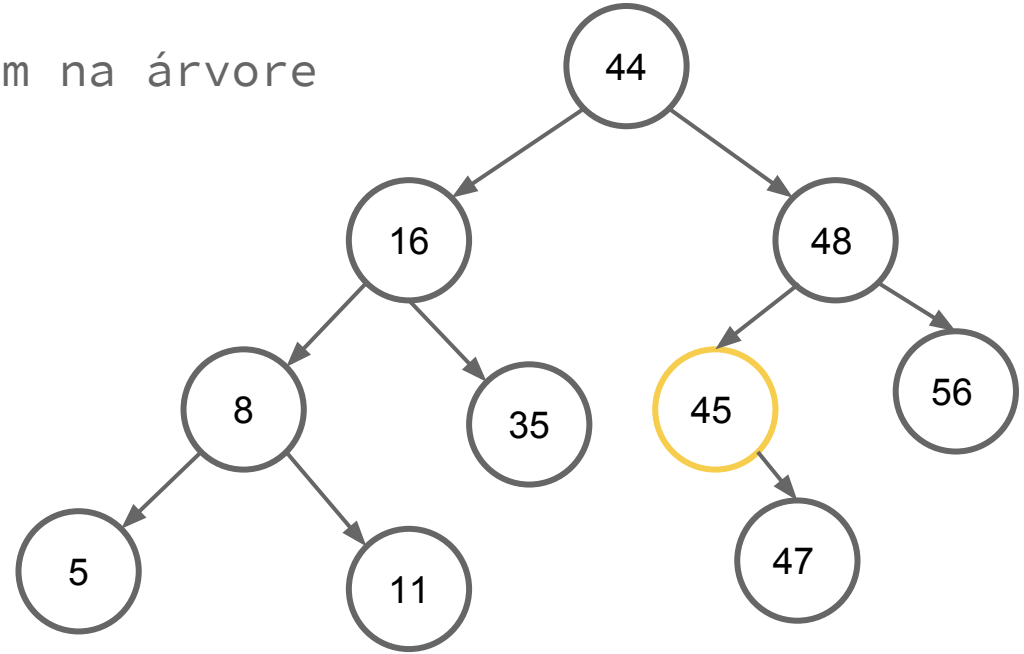
Remoção

- Remover número que tem na árvore

Folha

1 filho

2 filhos



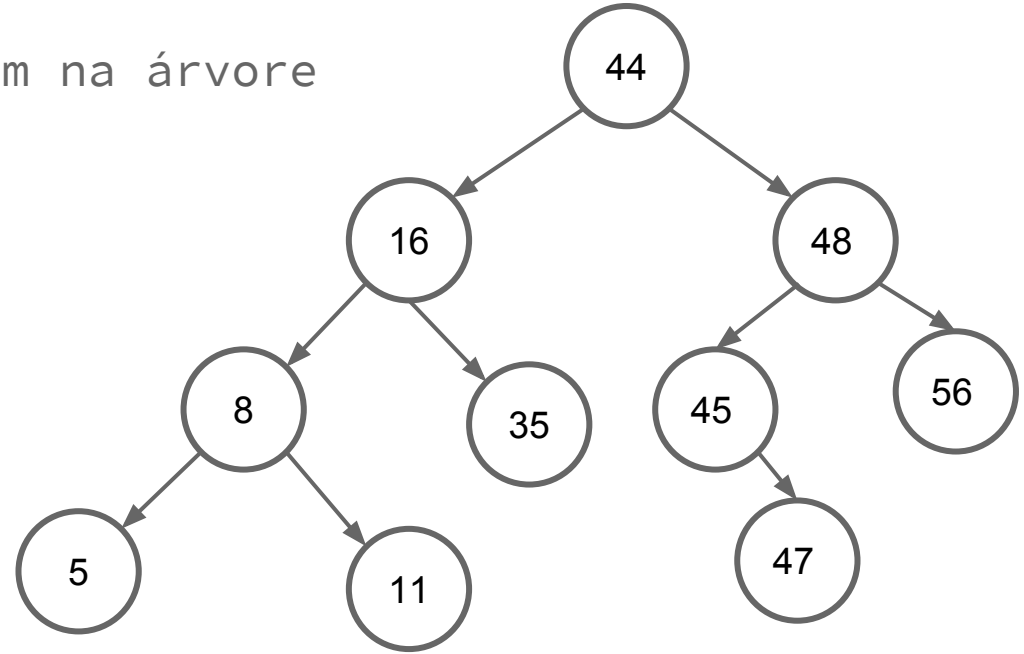
Remoção

- Remover número que tem na árvore

Folha

1 filho

2 filhos



Remover com 2 filhos

1) no = raiz, valor_remocao = 48
2) no_remocao = busca(no, valor_remocao)
3) SE no_remocao.esq != null && no_remocao.dir != null:
    a) Vai para a direita
    b) Vai para esquerda
    c) Se no não é folha, volta 3.b
    d) Se no é folha:
        i) no_remocao.valor = no.valor
        ii) remover_folha(no)

Remover com 2 filhos

```
1)  no = raiz, valor_remocao = 48
2)  no_remocao = busca(no, valor_remocao)
3)  SE no_remocao.esq != null && no_remocao.dir != null:
    a)  Vai para a direita
    b)  Vai para esquerda
    c)  Se no não é folha, volta 3.b
    d)  Se no é folha:
        i)  no_remocao.valor = no.valor
       ii)  remover_folha(no)
```

Remover com 2 filhos

1) no = raiz, valor_remocao = 48
2) no_remocao = busca(no, valor_remocao)
3) SE no_remocao.esq != null && no_remocao.dir != null:
    a) Vai para a direita
    b) Vai para esquerda
    c) Se no não é folha, volta 3.b
    d) Se no é folha:
        i) no_remocao.valor = no.valor
        ii) remover_folha(no)

O(log n)

Busca

Operações

Inserção 

Busca 

Remoção 

Operações

Inserção 

Busca 

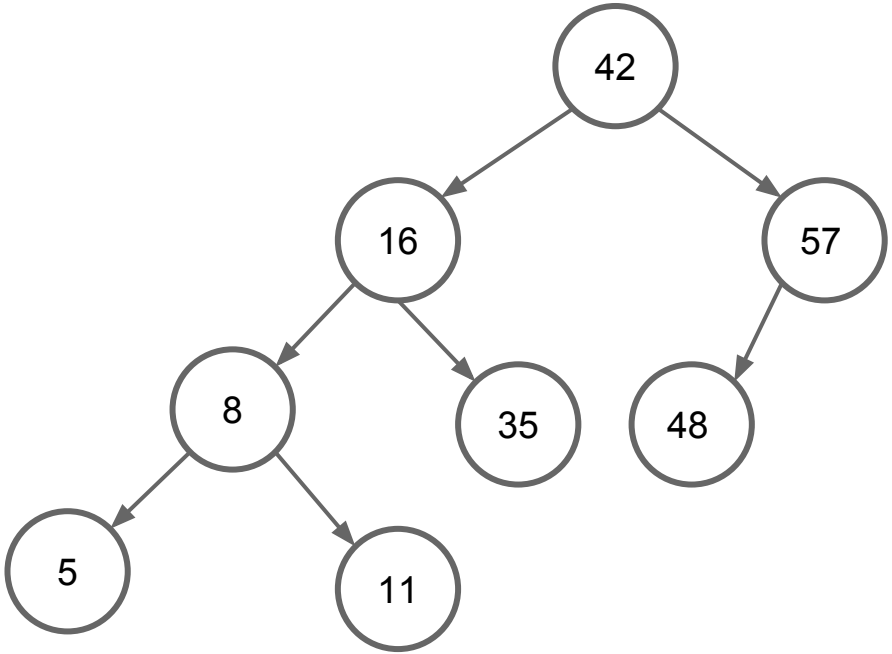
Remoção 

Travessia

visualgo

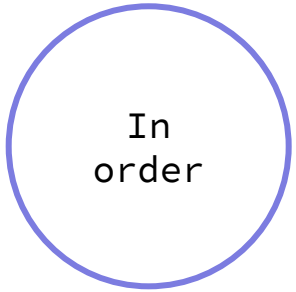
Travessia

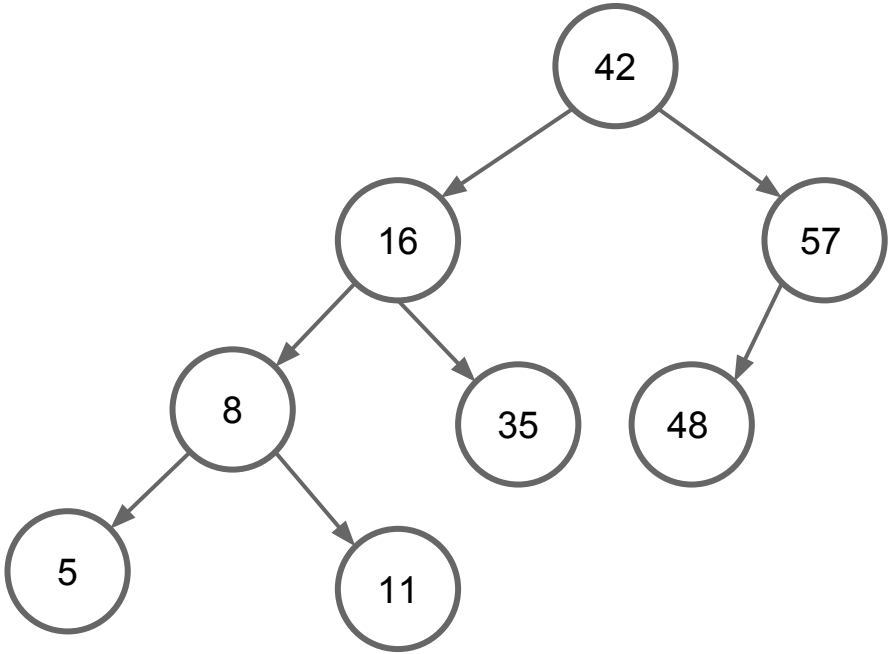
- Percorrer a árvore



Travessia

- Percorrer a árvore



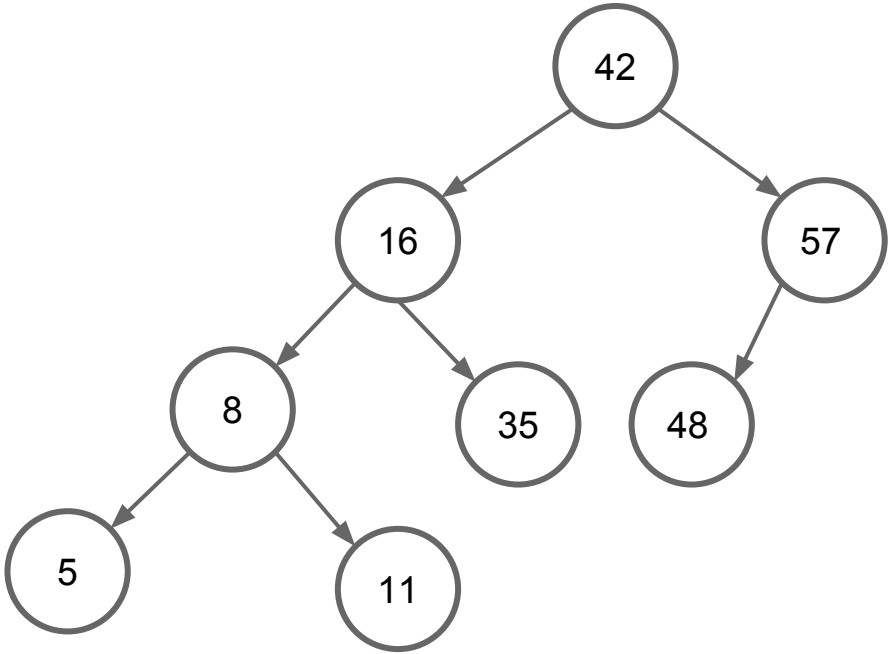


Travessia

- Percorrer a árvore

In order

Pre order



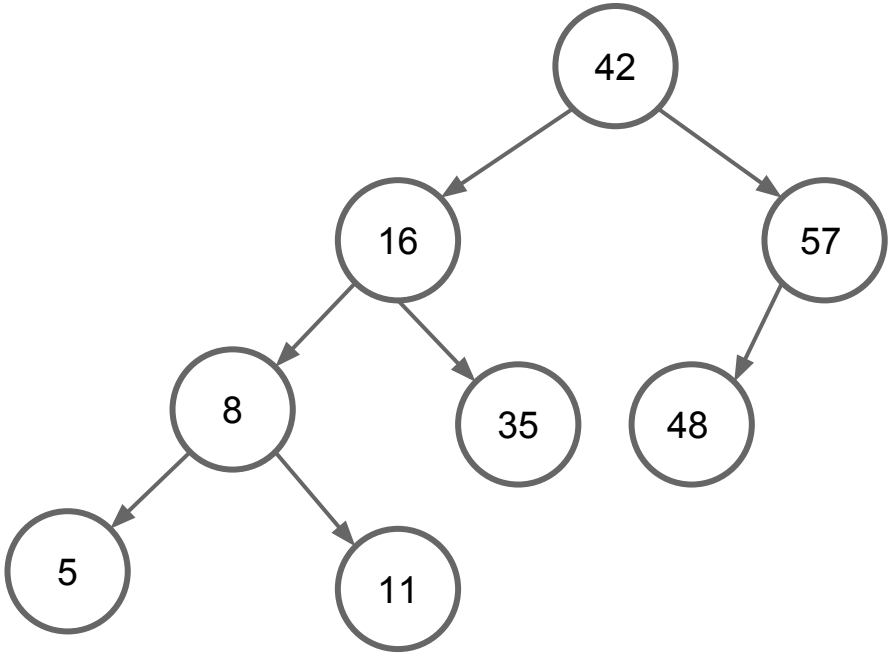
Travessia

- Percorrer a árvore

In order

Pre order

Post order



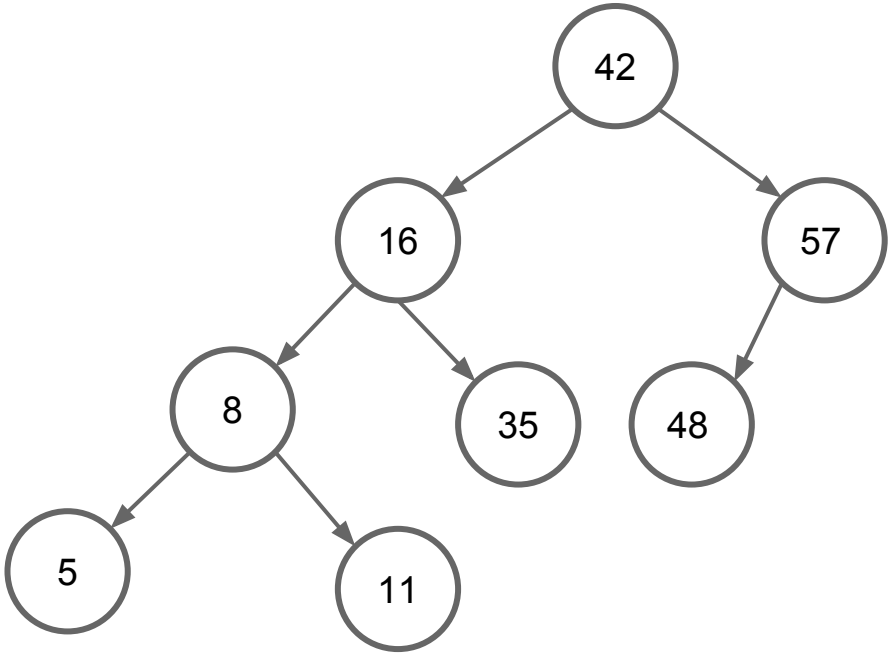
Travessia

- Percorrer a árvore

In order

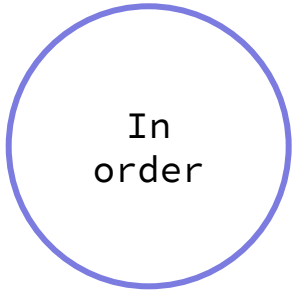
Pre order

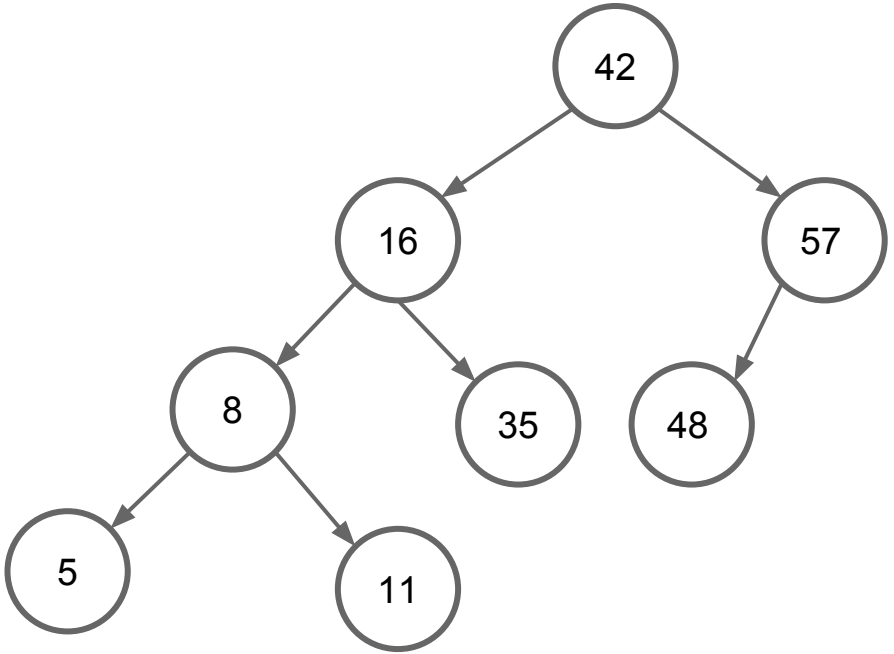
Post order



Travessia

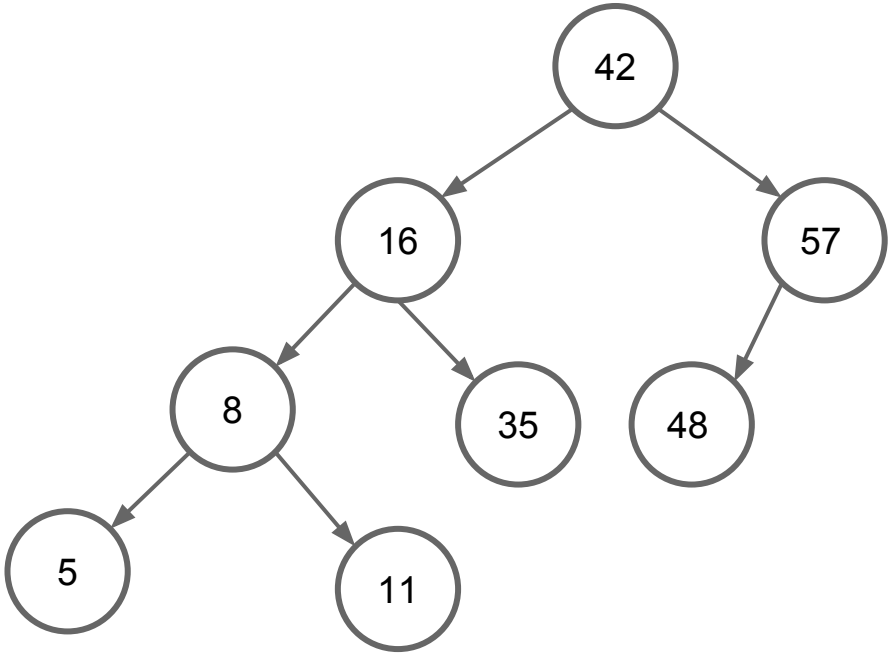
- Percorrer a árvore





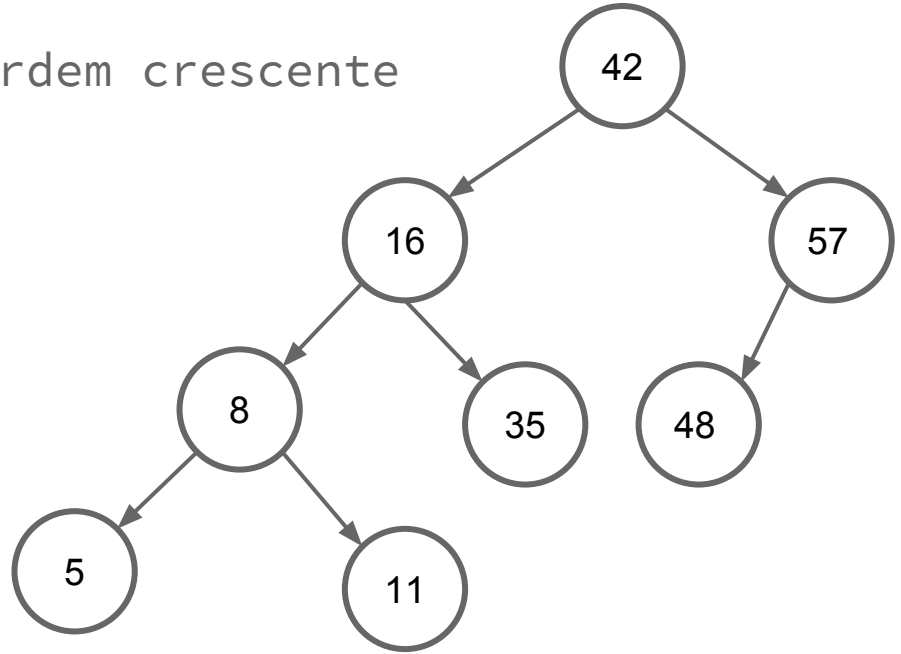
Travessia In-order

- Percorrer a árvore



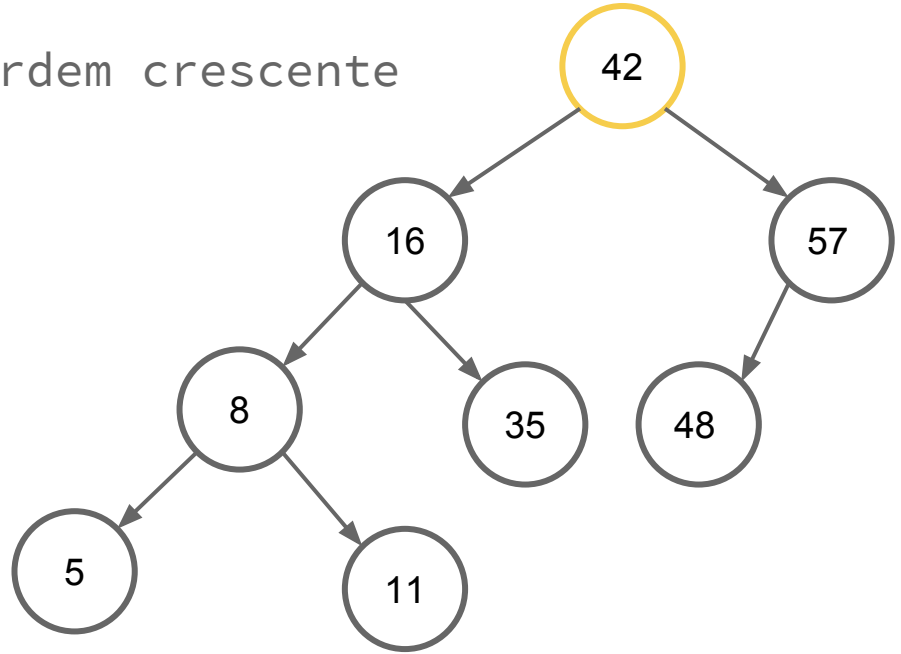
Travessia In-order

- Percorrer a árvore: ordem crescente



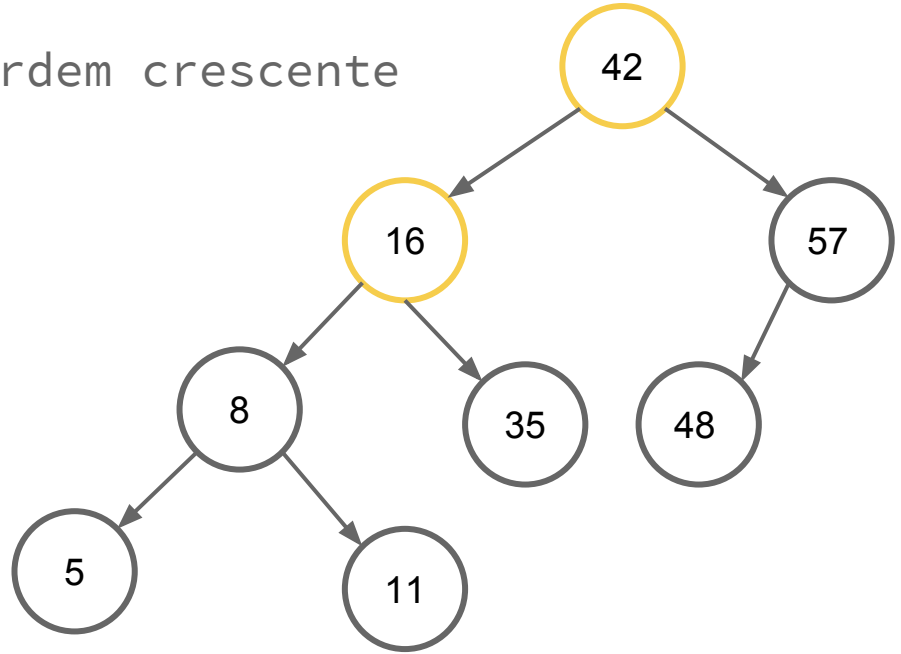
Travessia In-order

- Percorrer a árvore: ordem crescente



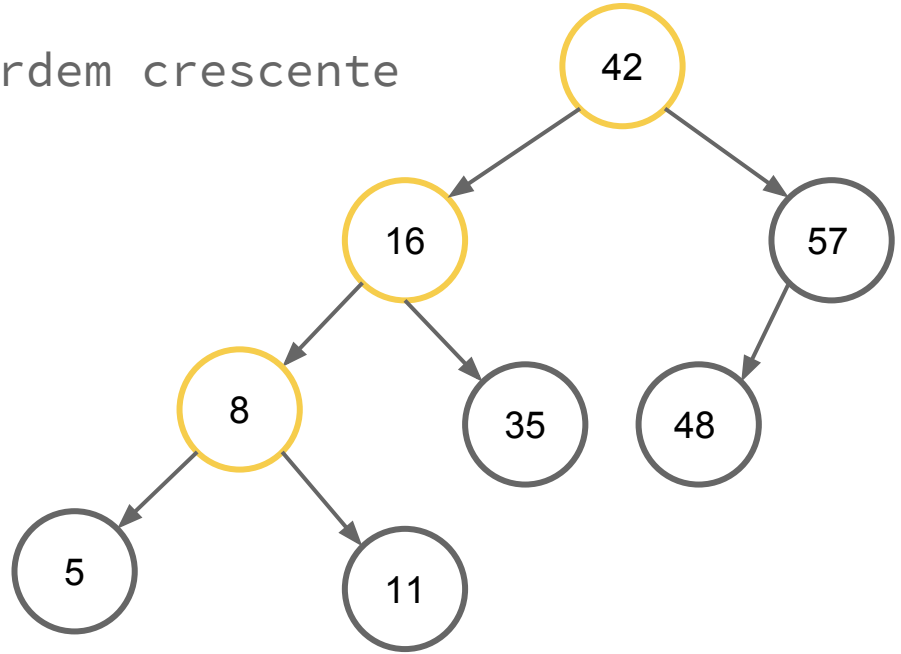
Travessia In-order

- Percorrer a árvore: ordem crescente



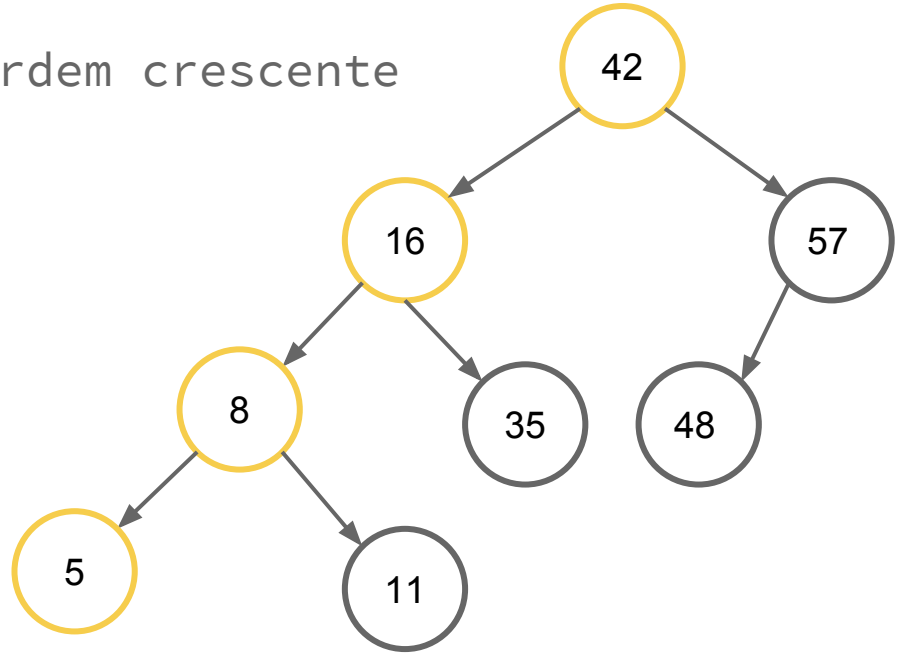
Travessia In-order

- Percorrer a árvore: ordem crescente



Travessia In-order

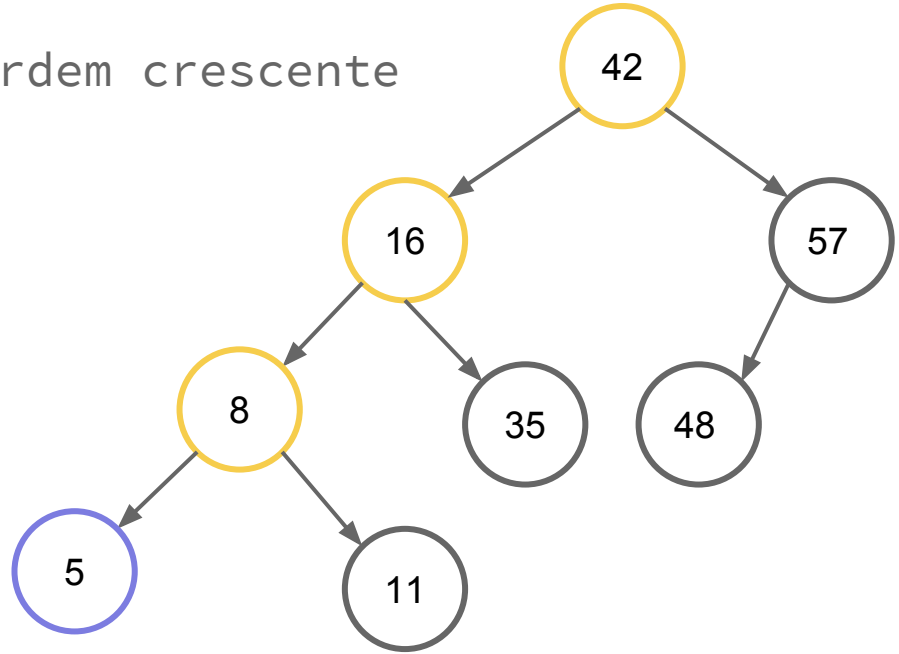
- Percorrer a árvore: ordem crescente



Travessia In-order

- Percorrer a árvore: ordem crescente

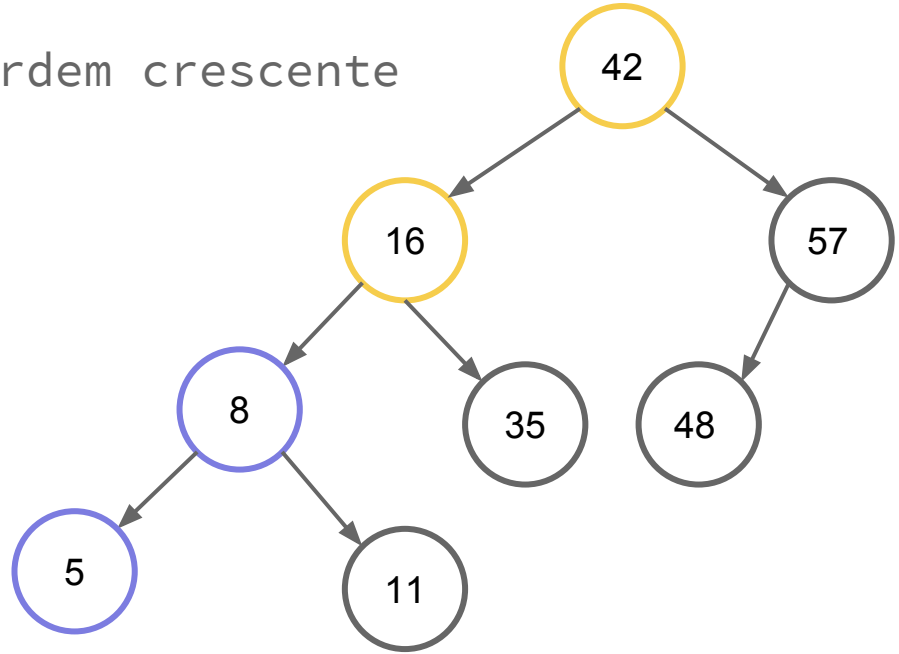
```
5,
```



Travessia In-order

- Percorrer a árvore: ordem crescente

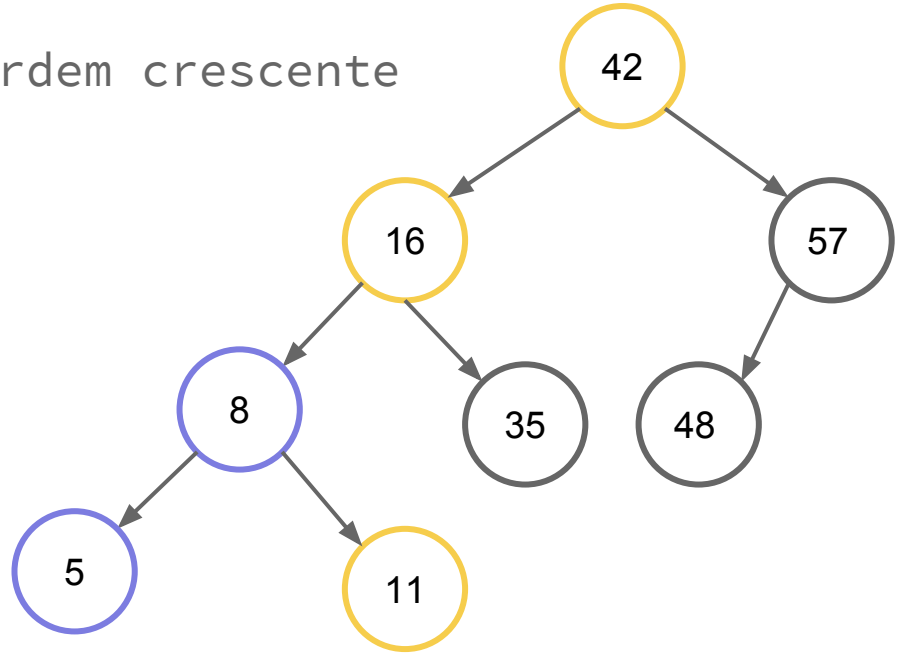
```
5, 8
```



Travessia In-order

- Percorrer a árvore: ordem crescente

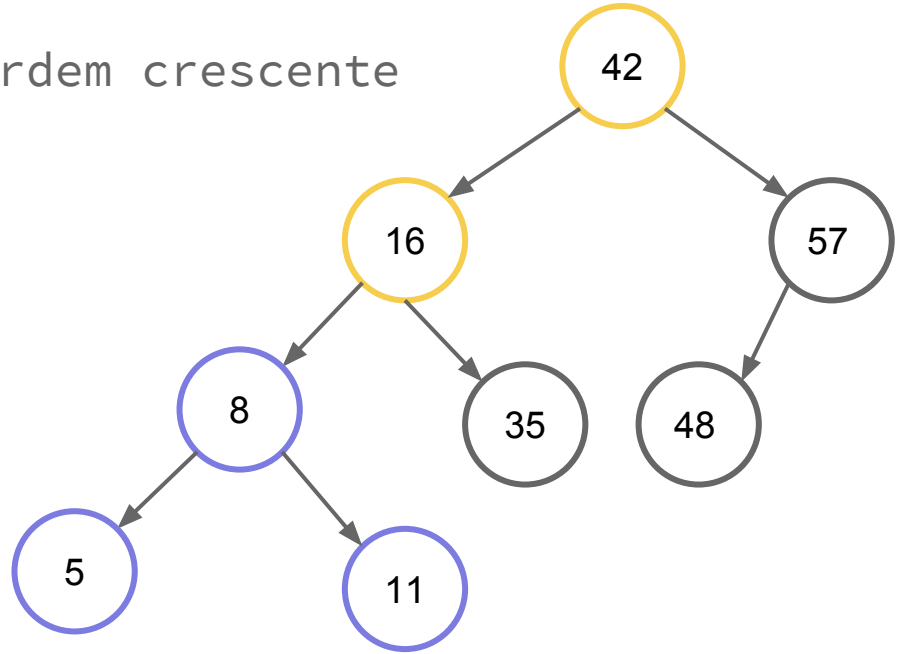
```
5, 8
```



Travessia In-order

- Percorrer a árvore: ordem crescente

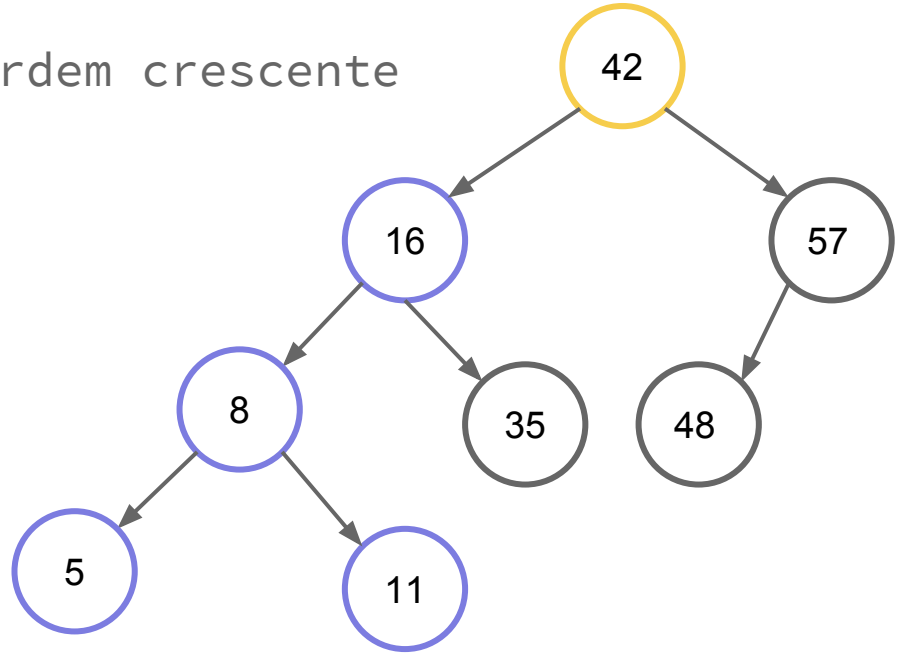
```
5, 8, 11
```



Travessia In-order

- Percorrer a árvore: ordem crescente

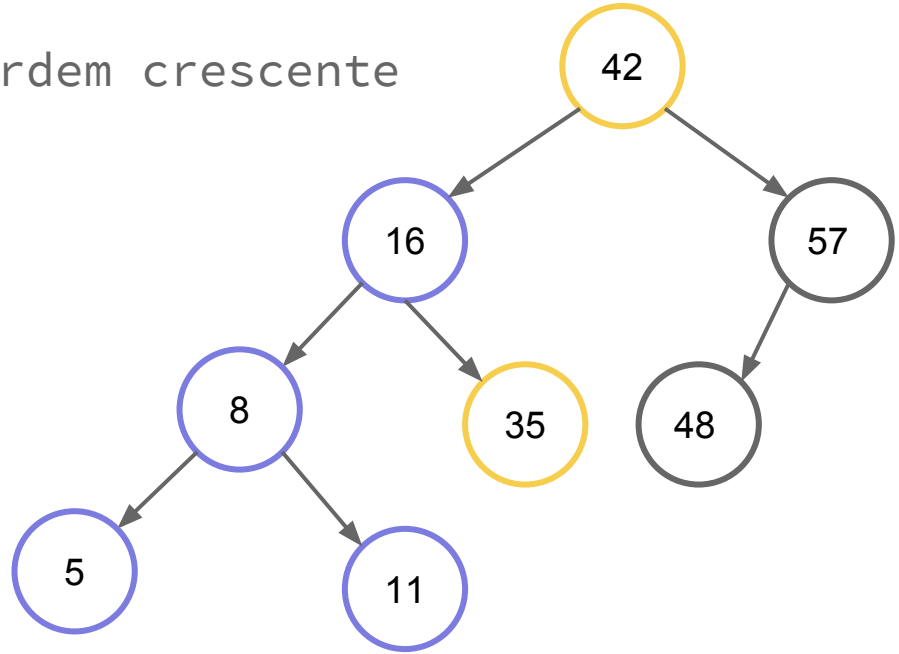
```
5, 8, 11, 16
```



Travessia In-order

- Percorrer a árvore: ordem crescente

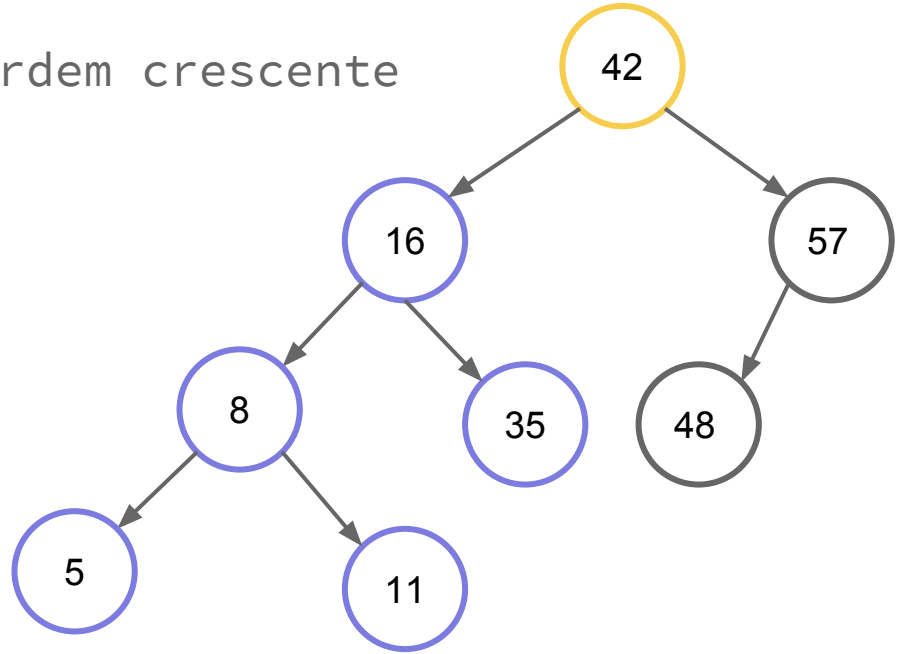
```
5, 8, 11, 16
```



Travessia In-order

- Percorrer a árvore: ordem crescente

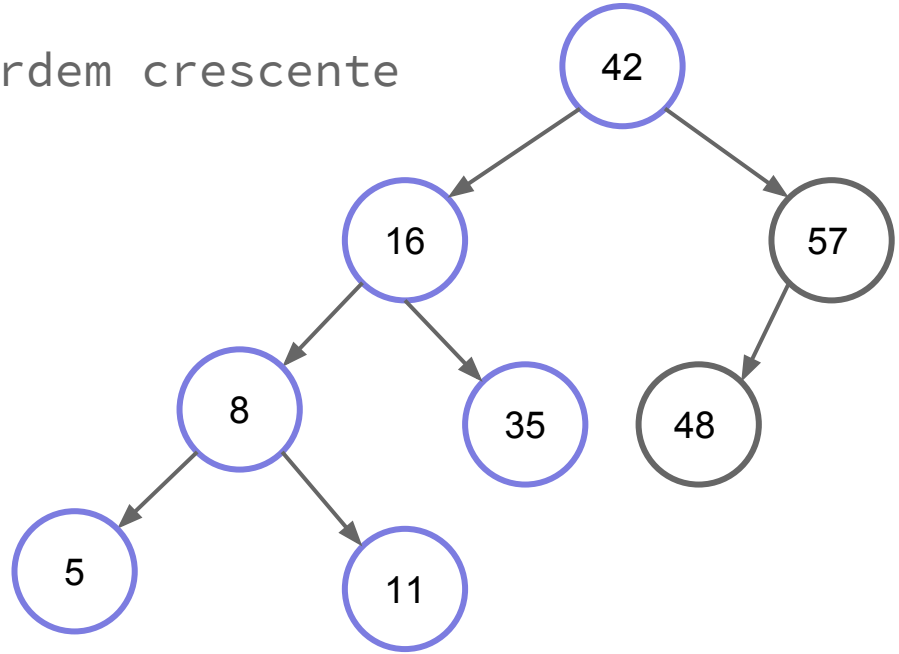
```
5, 8, 11, 16, 35
```



Travessia In-order

- Percorrer a árvore: ordem crescente

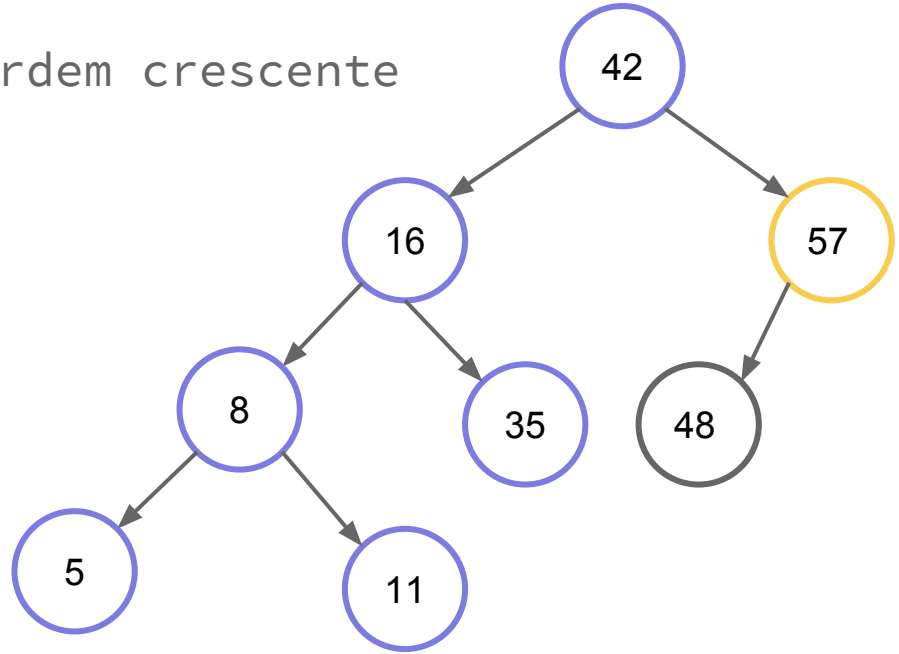
```
5, 8, 11, 16, 35,
42
```



Travessia In-order

- Percorrer a árvore: ordem crescente

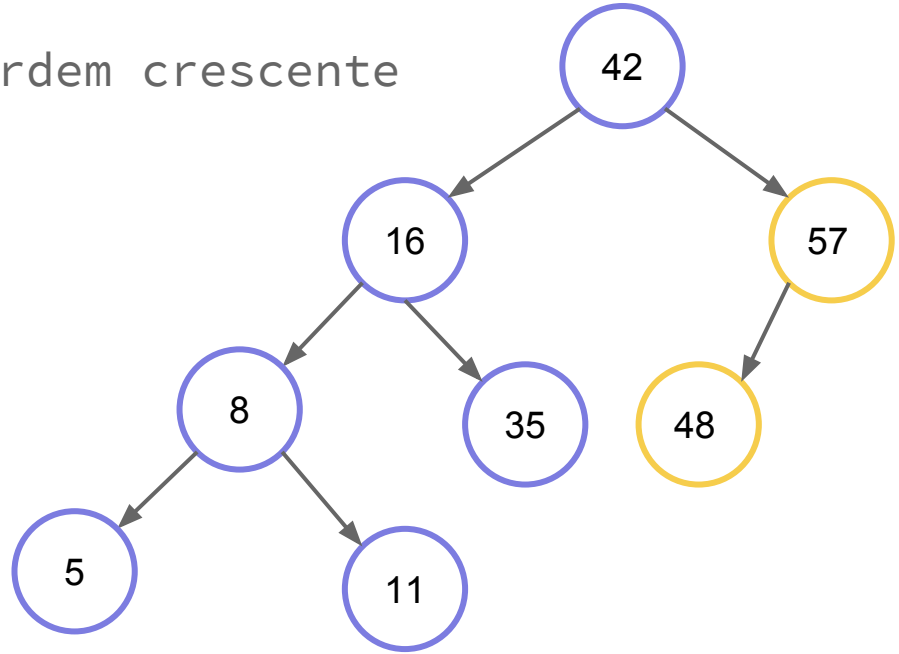
```
5, 8, 11, 16, 35,
42
```



Travessia In-order

- Percorrer a árvore: ordem crescente

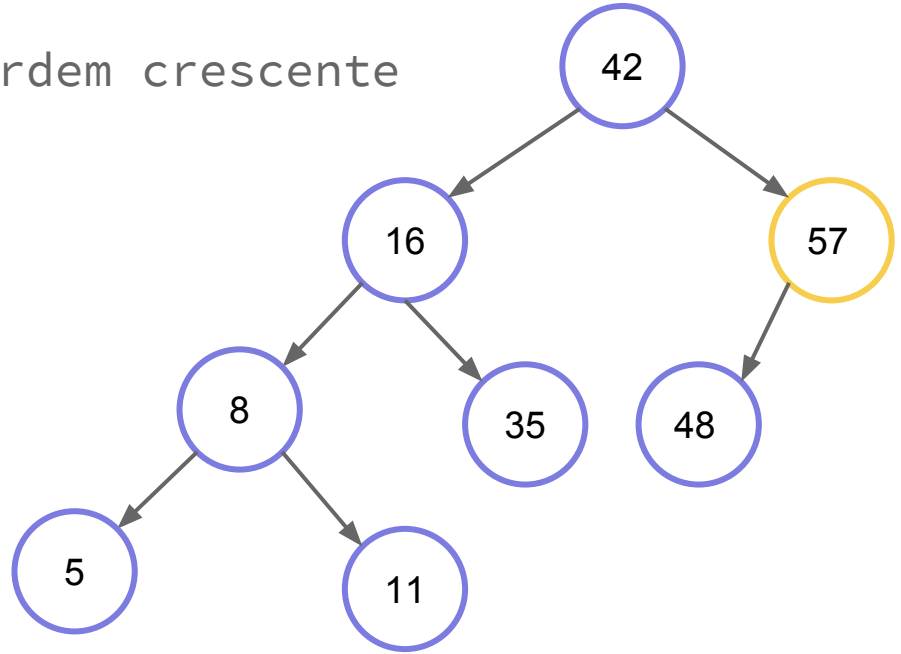
```
5, 8, 11, 16, 35,
42
```



Travessia In-order

- Percorrer a árvore: ordem crescente

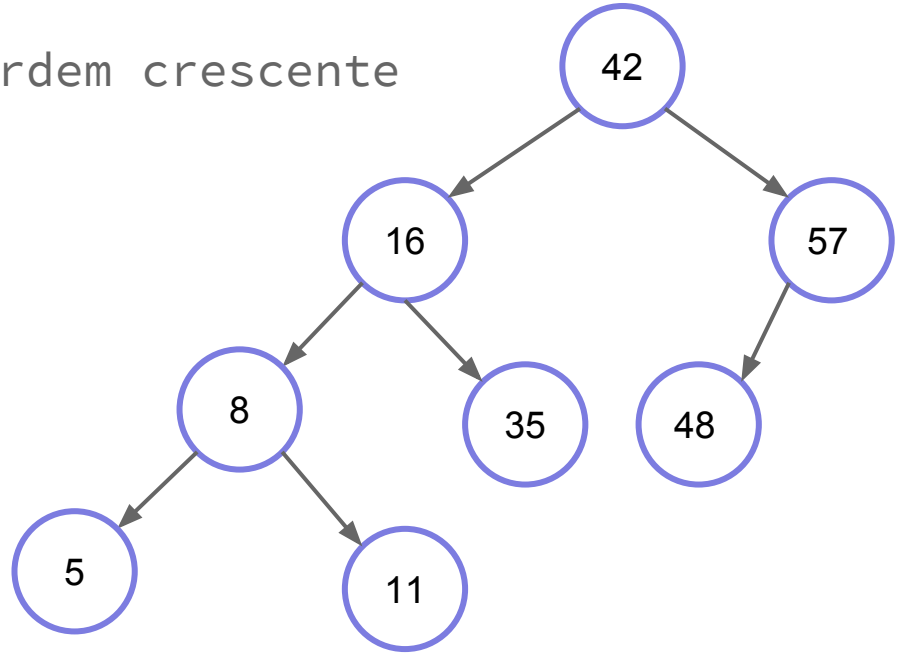
```
5, 8, 11, 16, 35,
42, 48
```



Travessia In-order

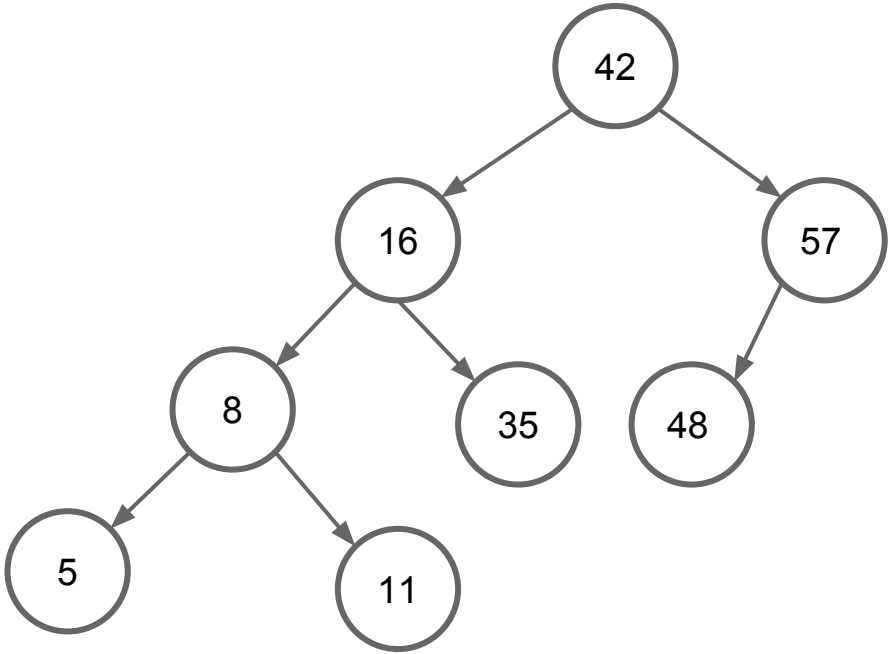
- Percorrer a árvore: ordem crescente

```
5, 8, 11, 16, 35,
42, 48, 57
```



Travessia Pre-order

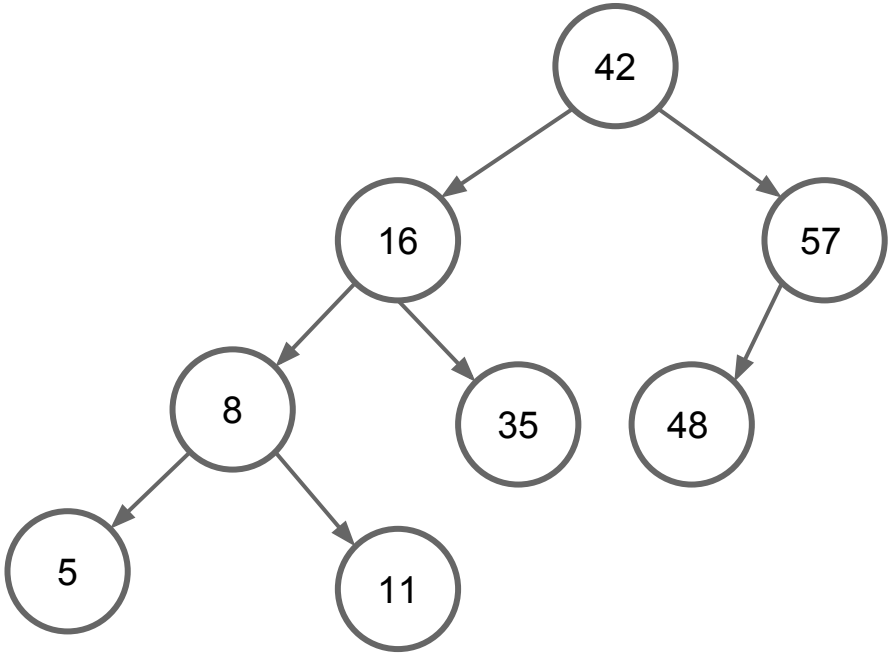
- Percorrer a árvore



Pre
order

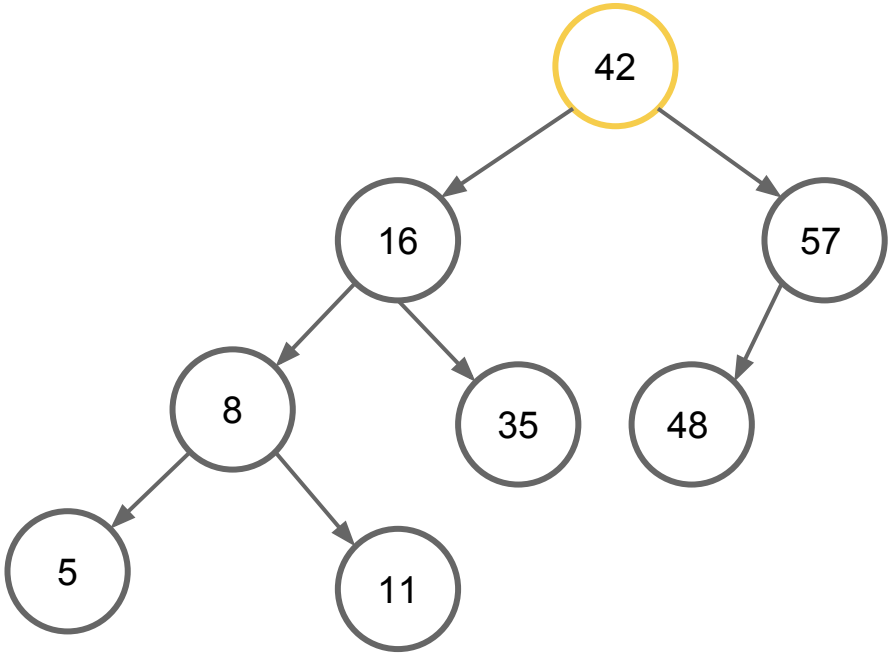
Travessia Pre-order

- Percorrer a árvore



Travessia Pre-order

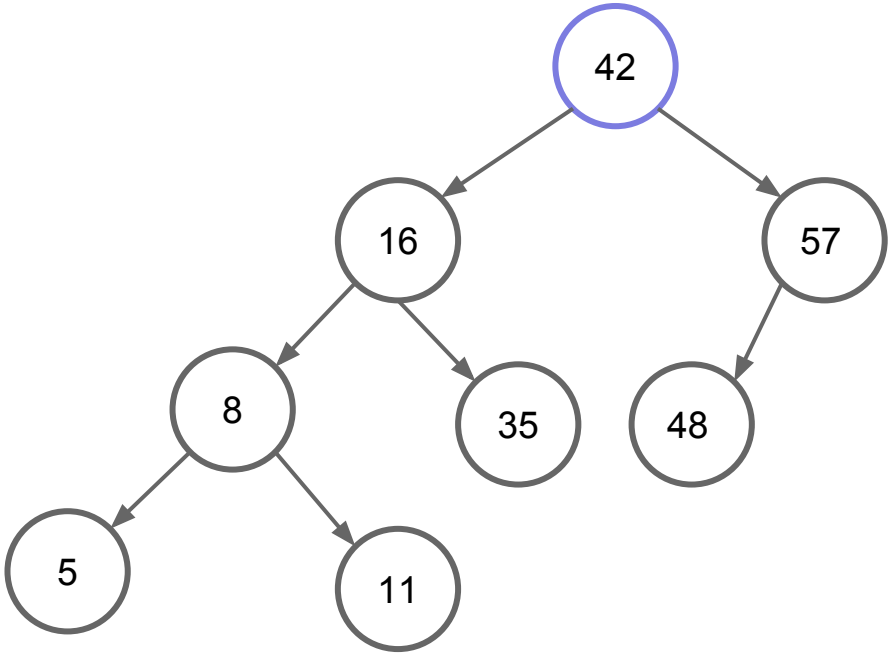
- Percorrer a árvore



Travessia Pre-order

- Percorrer a árvore

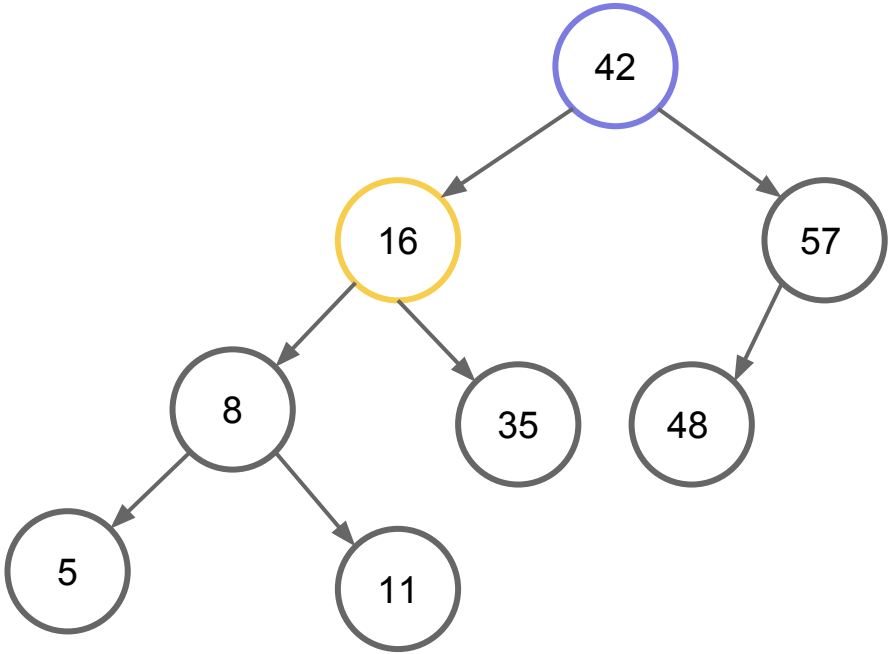
```
42,
```



Travessia Pre-order

- Percorrer a árvore

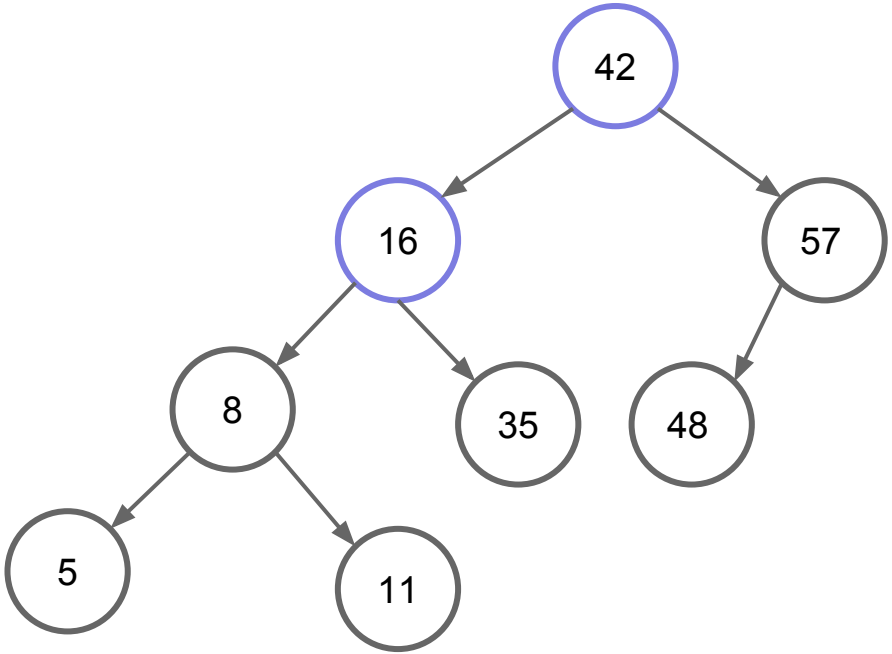
```
42,
```



Travessia Pre-order

- Percorrer a árvore

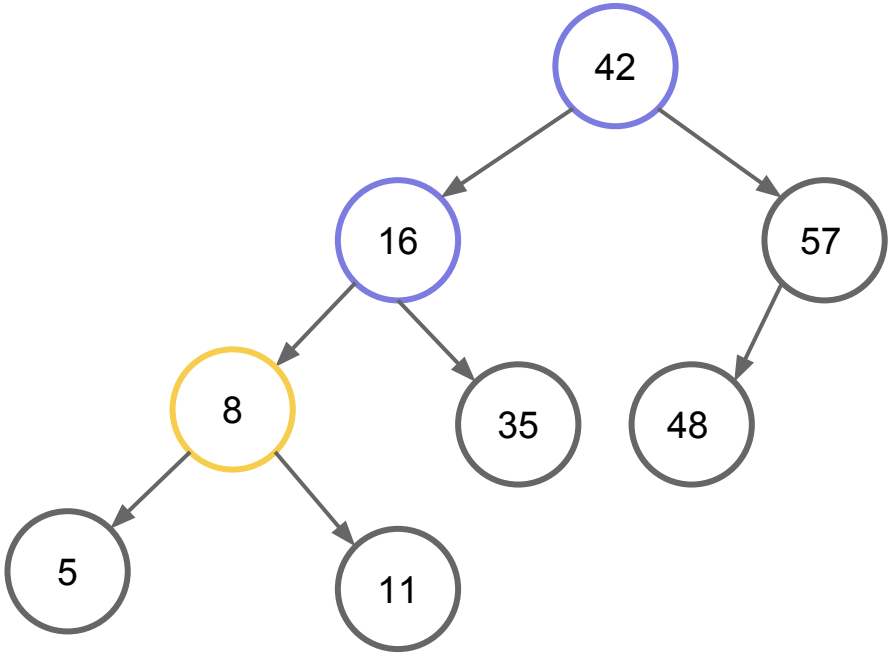
```
42, 16
```



Travessia Pre-order

- Percorrer a árvore

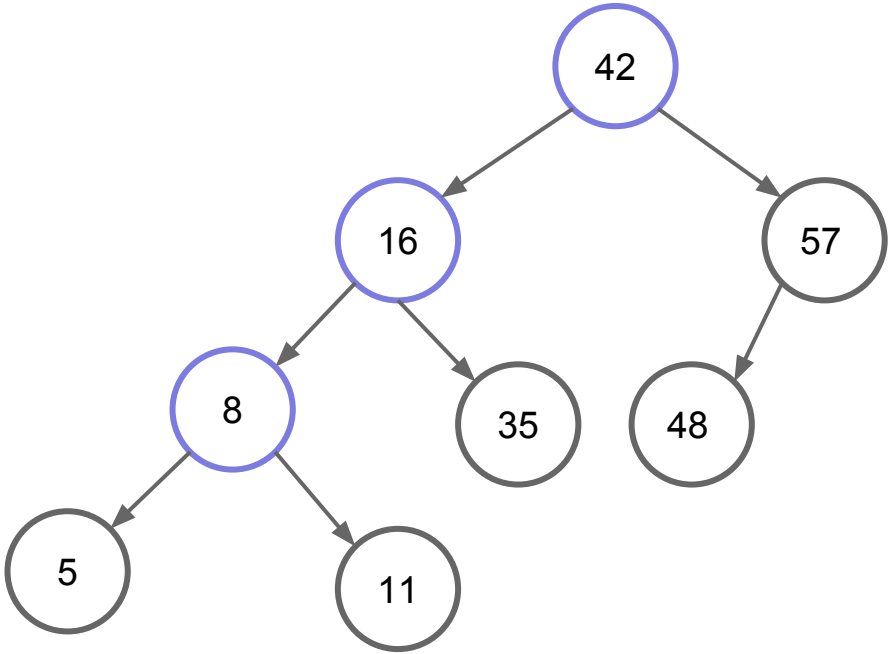
```
42, 16
```



Travessia Pre-order

- Percorrer a árvore

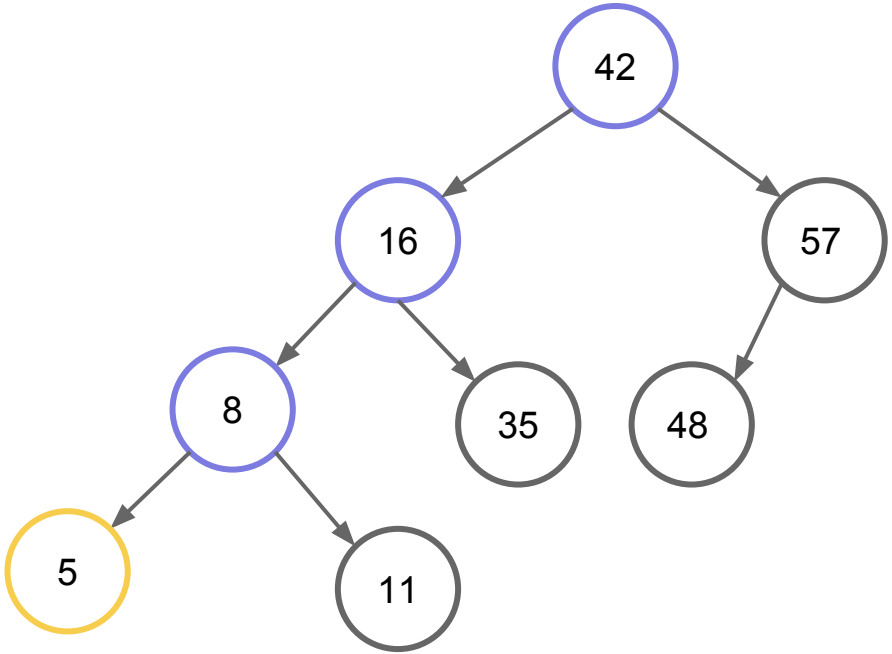
```
42, 16, 8
```



Travessia Pre-order

- Percorrer a árvore

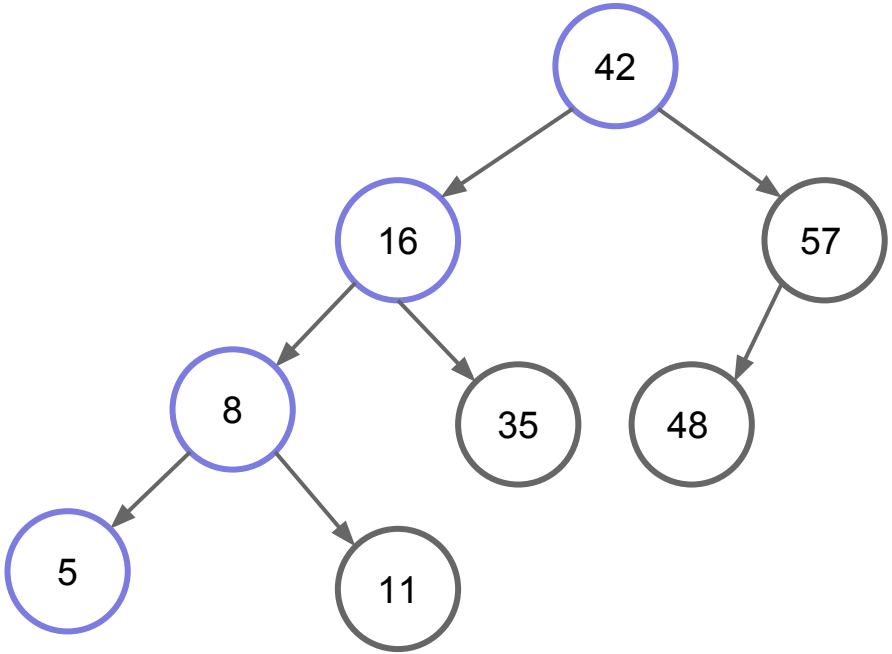
```
42, 16, 8
```



Travessia Pre-order

- Percorrer a árvore

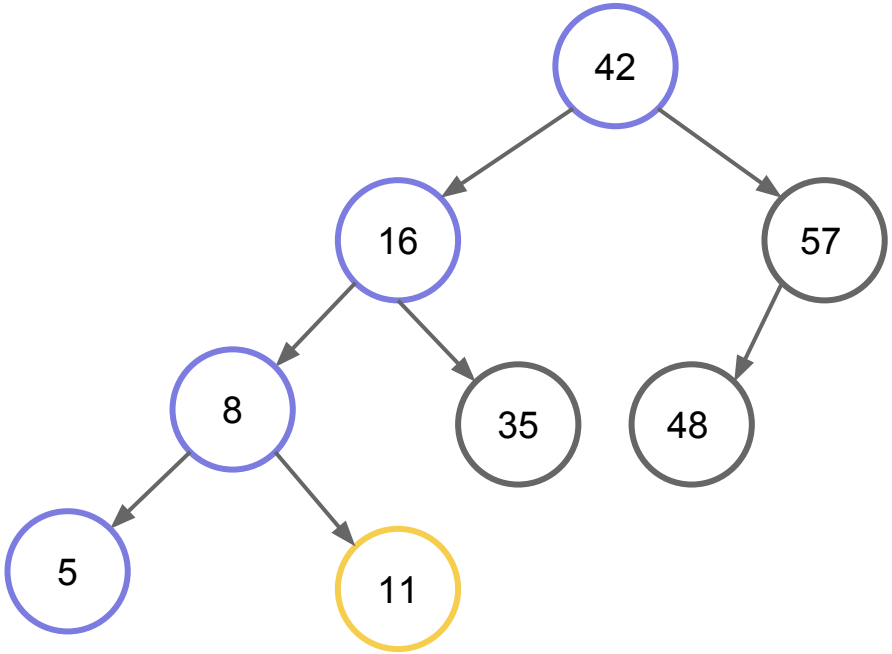
```
42, 16, 8, 5
```



Travessia Pre-order

- Percorrer a árvore

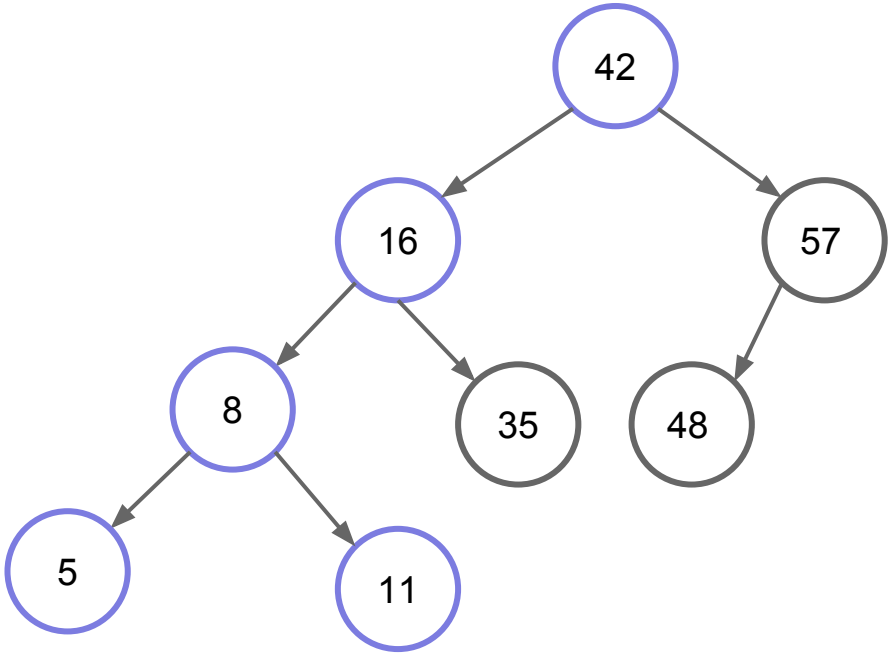
```
42, 16, 8, 5
```



Travessia Pre-order

- Percorrer a árvore

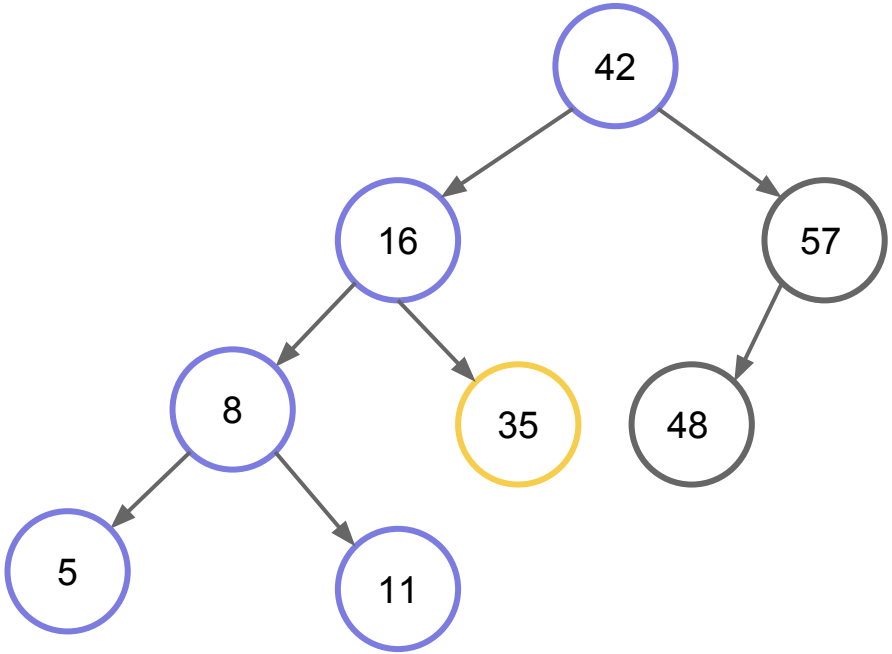
```
42, 16, 8, 5, 11
```



Travessia Pre-order

- Percorrer a árvore

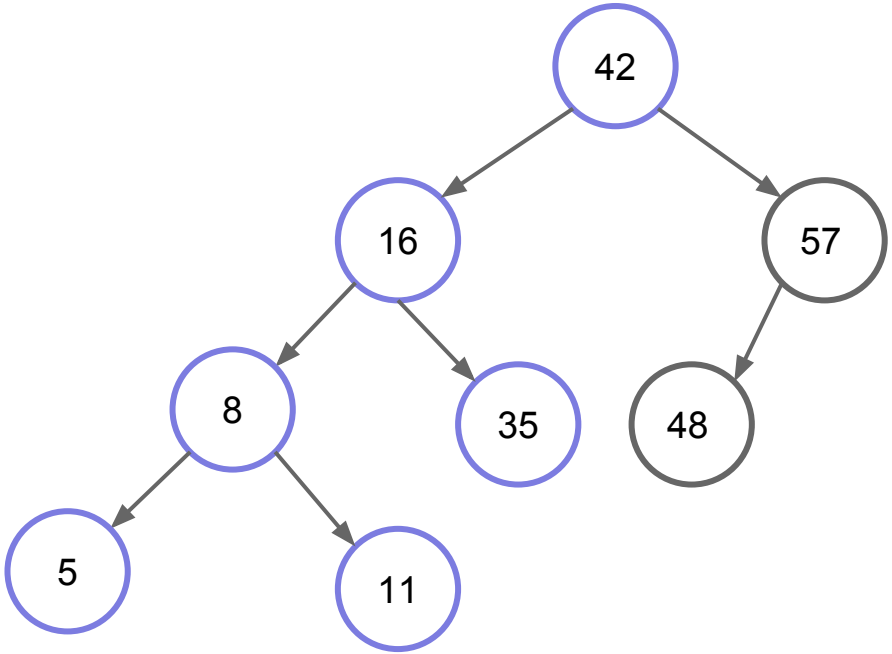
```
42, 16, 8, 5, 11
```



Travessia Pre-order

- Percorrer a árvore

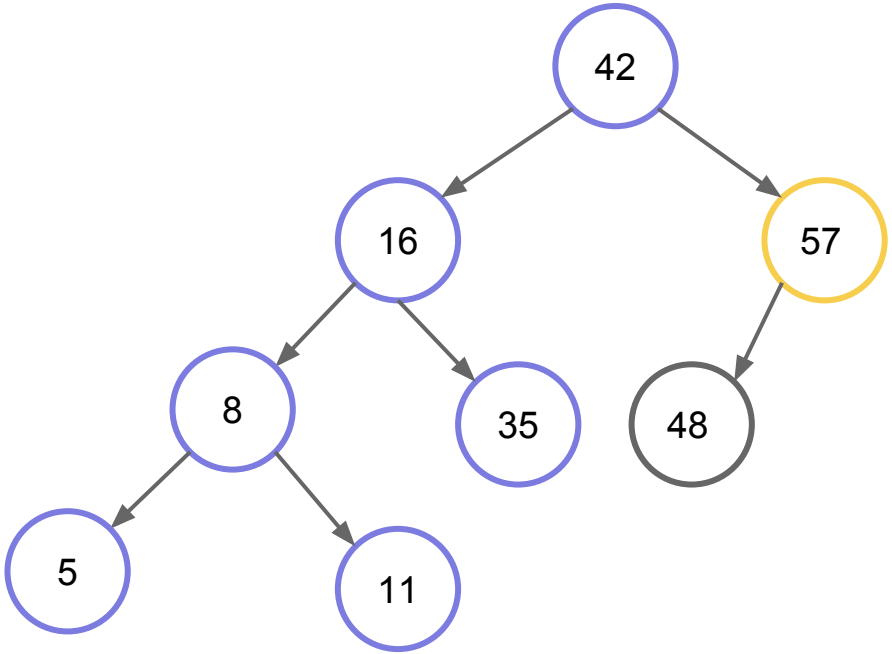
```
42, 16, 8, 5, 11,
35
```



Travessia Pre-order

- Percorrer a árvore

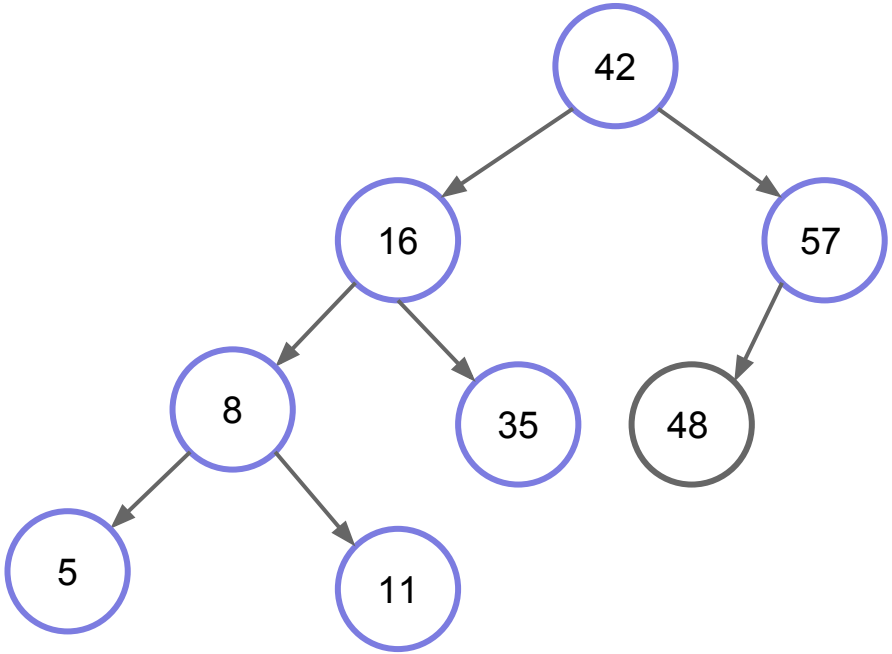
```
42, 16, 8, 5, 11,
35
```



Travessia Pre-order

- Percorrer a árvore

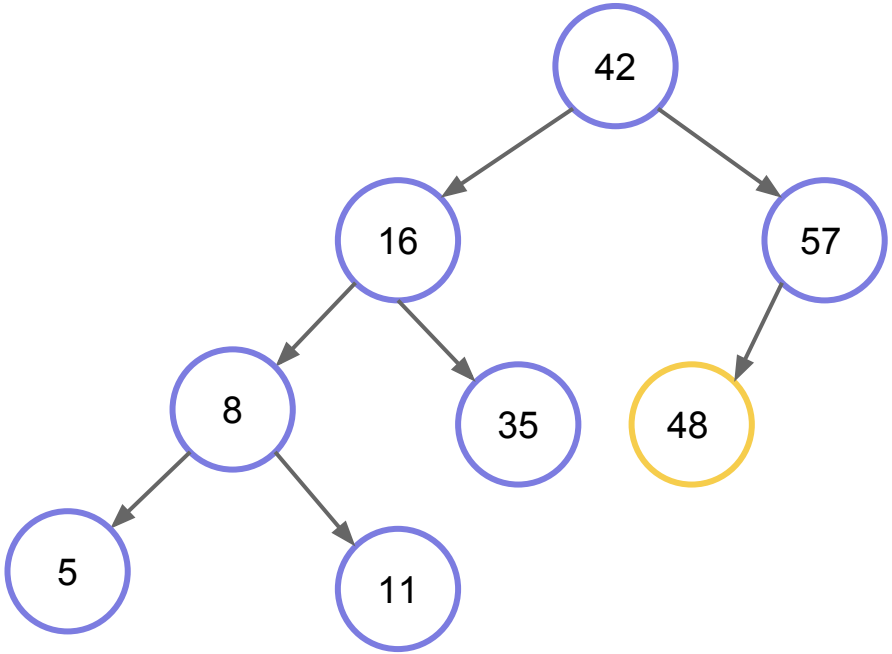
```
42, 16, 8, 5, 11,
35, 57
```



Travessia Pre-order

- Percorrer a árvore

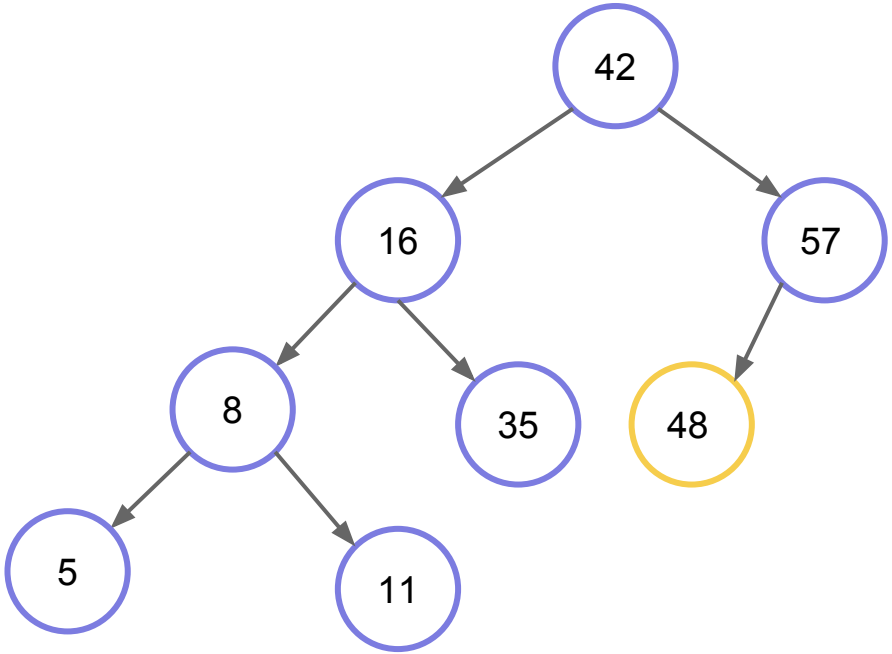
```
42, 16, 8, 5, 11,
35, 57
```



Travessia Pre-order

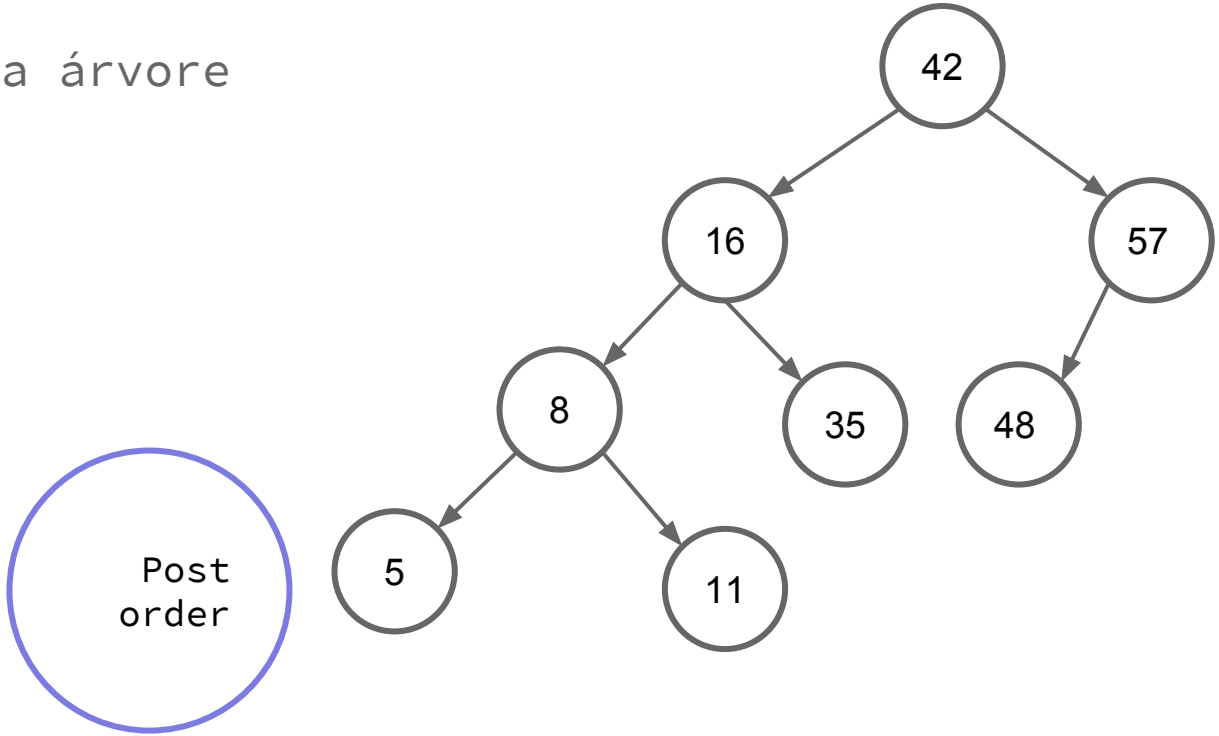
- Percorrer a árvore

```
42, 16, 8, 5, 11,
35, 57, 48
```



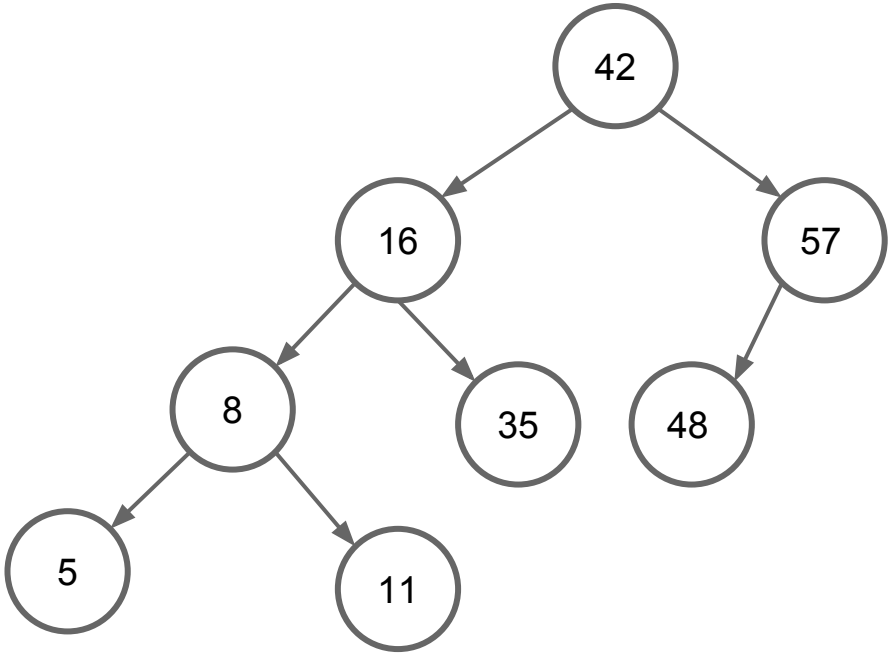
Travessia

- Percorrer a árvore



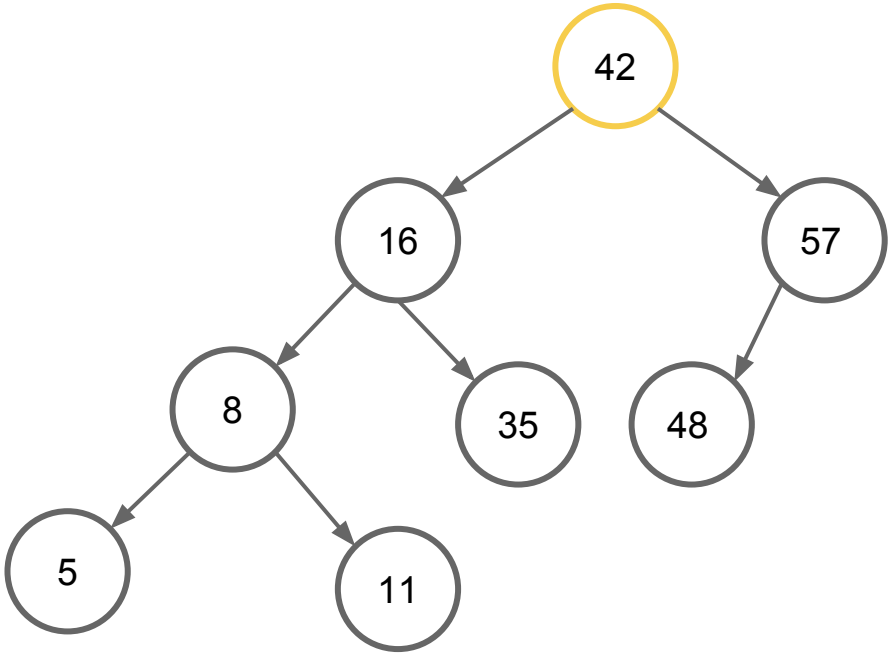
Travessia Post-order

- Percorrer a árvore



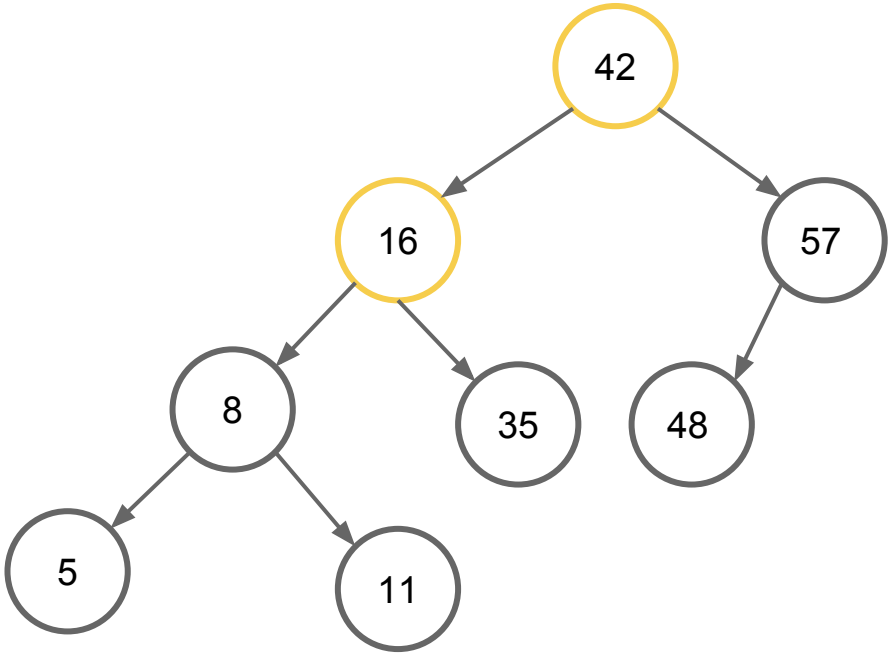
Travessia Post-order

- Percorrer a árvore



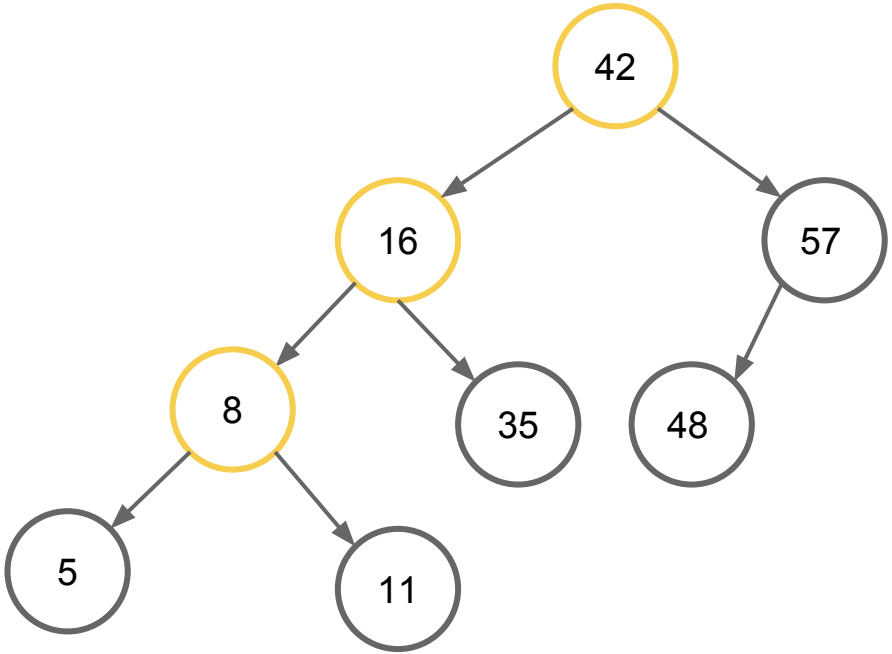
Travessia Post-order

- Percorrer a árvore



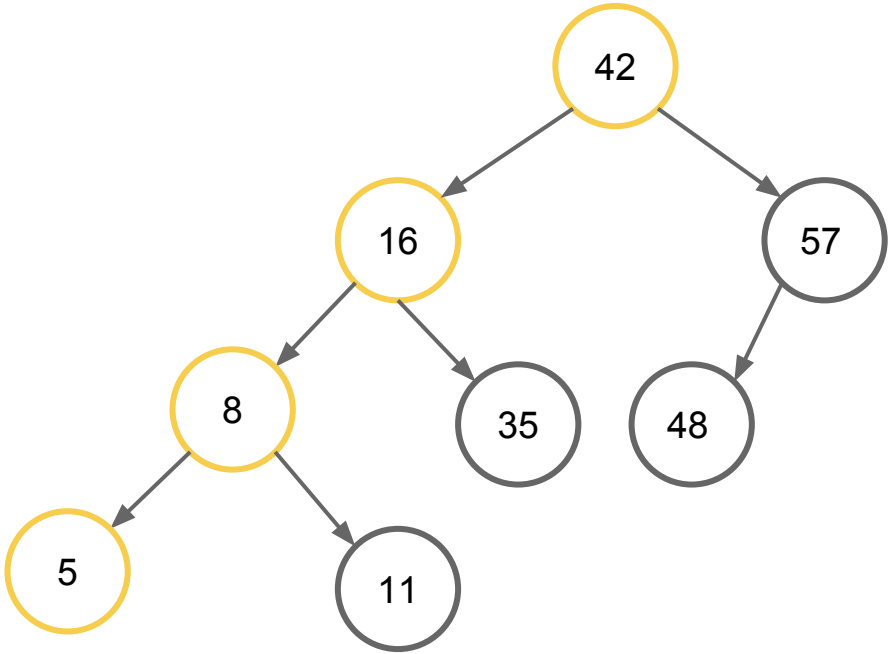
Travessia Post-order

- Percorrer a árvore



Travessia Post-order

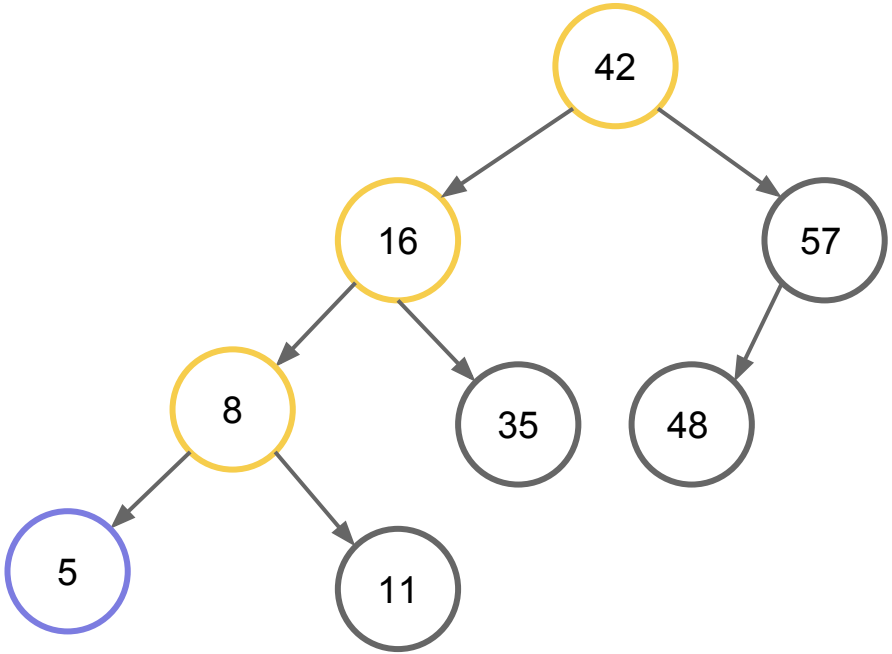
- Percorrer a árvore



Travessia Post-order

- Percorrer a árvore

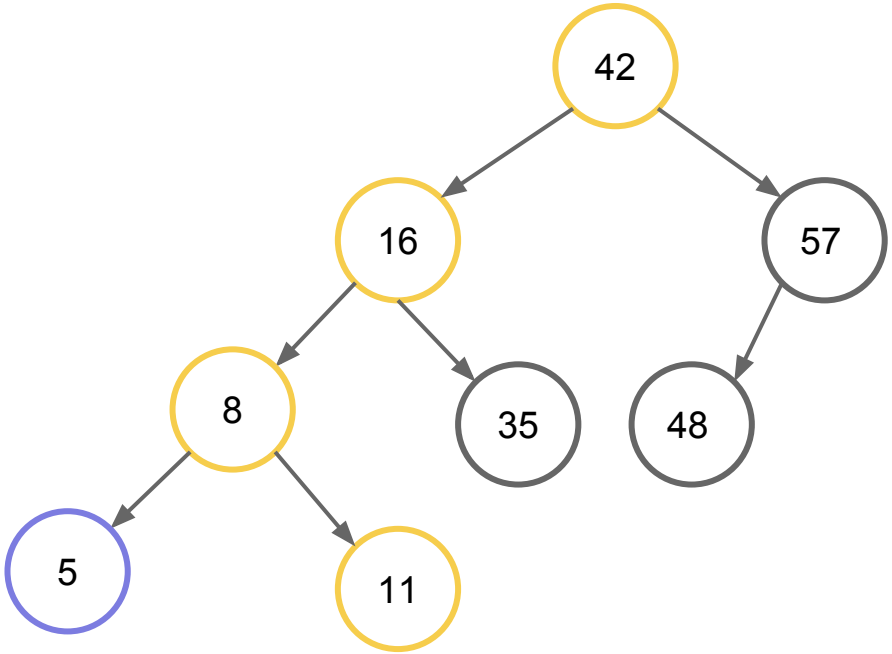
```
5,
```



Travessia Post-order

- Percorrer a árvore

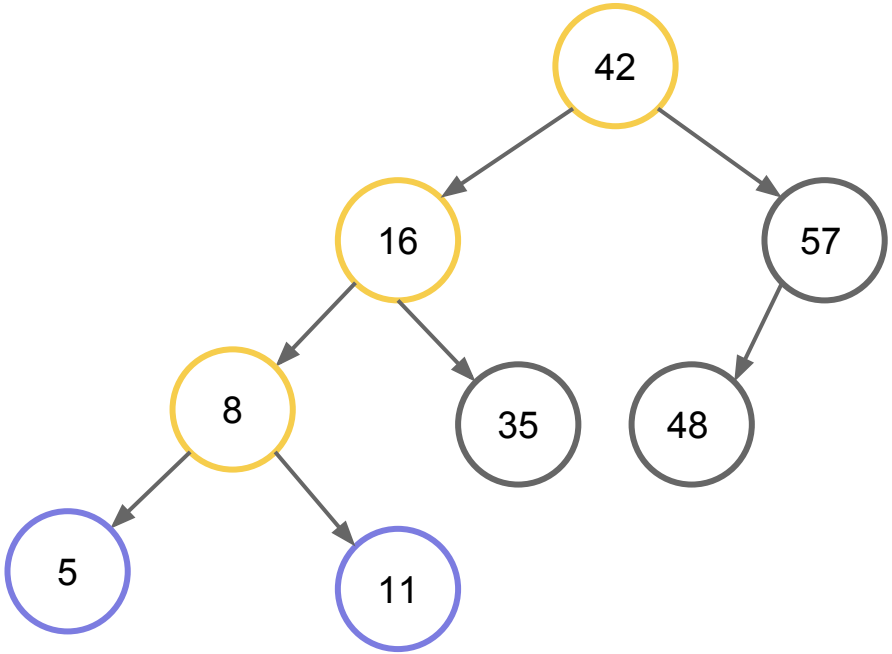
```
5,
```



Travessia Post-order

- Percorrer a árvore

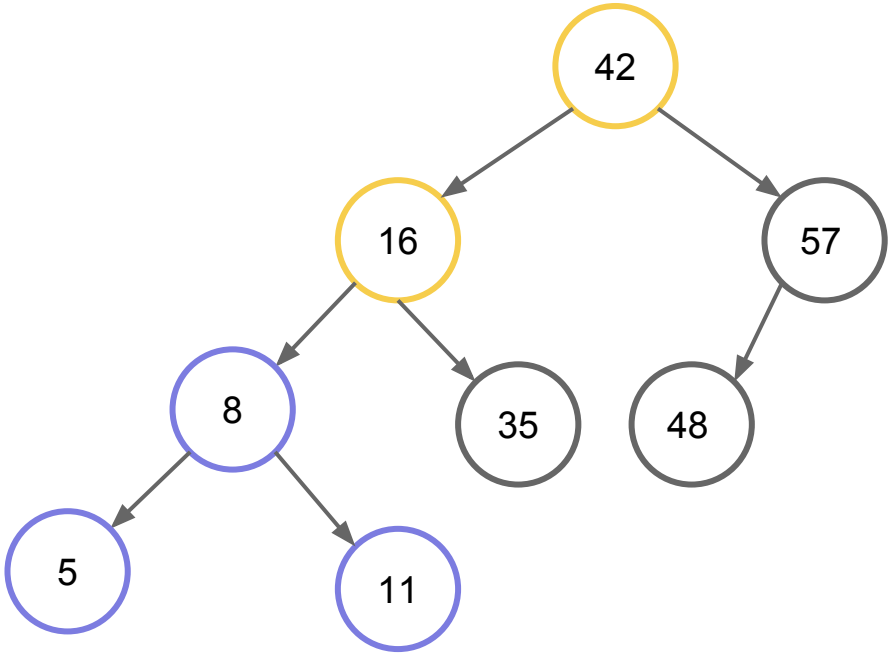
```
5, 11,
```



Travessia Post-order

- Percorrer a árvore

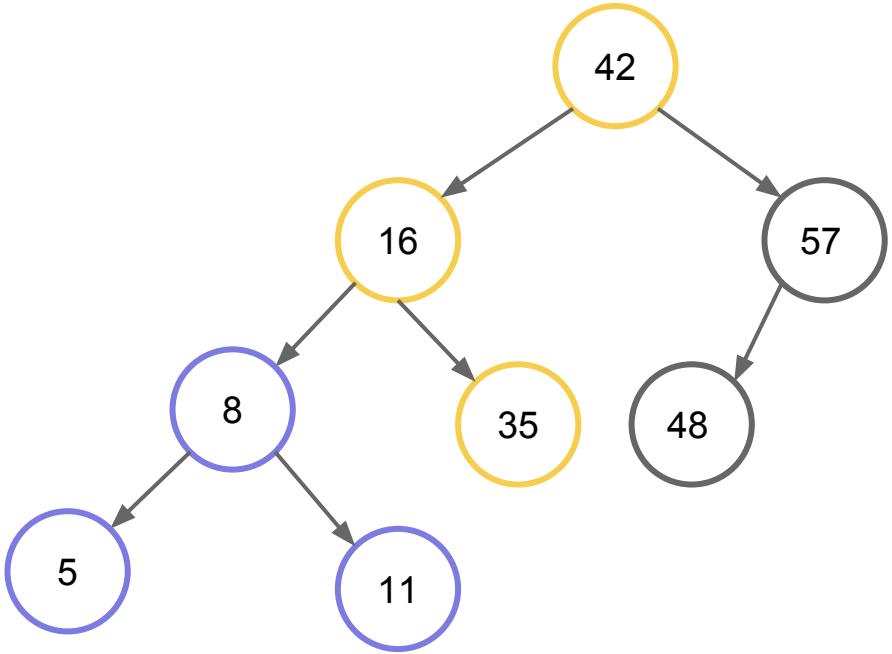
```
5, 11, 8,
```



Travessia Post-order

- Percorrer a árvore

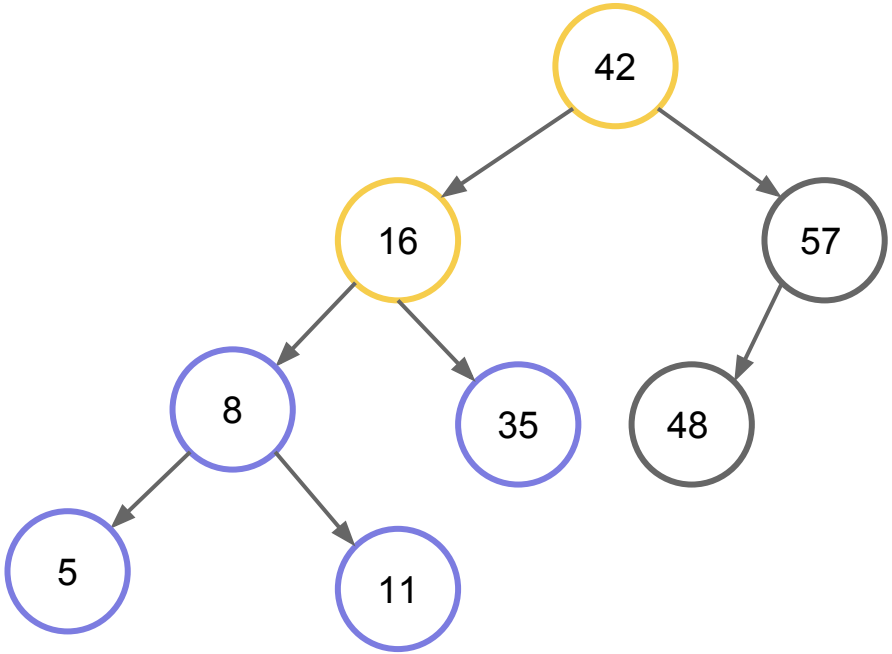
```
5, 11, 8,
```



Travessia Post-order

- Percorrer a árvore

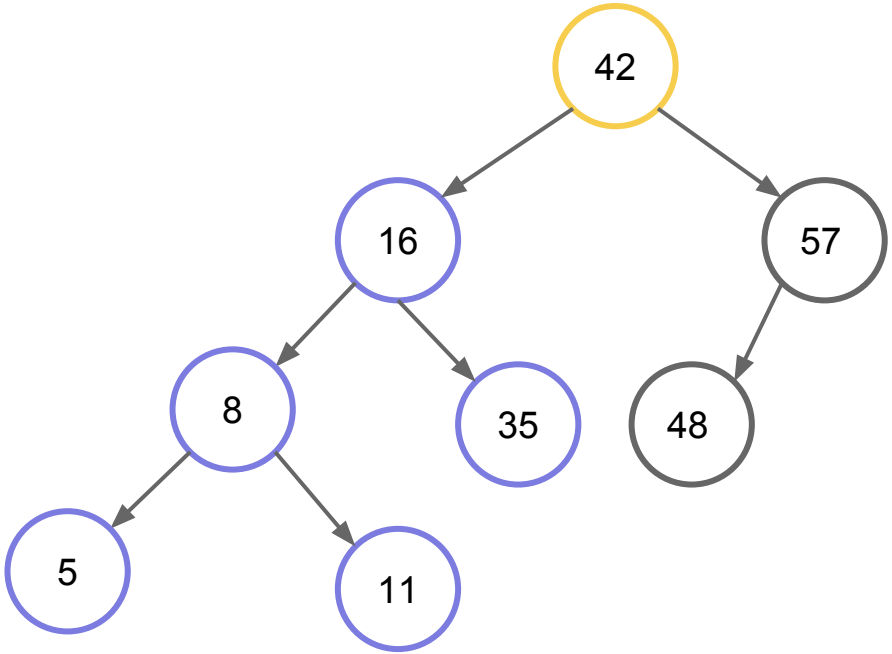
```
5, 11, 8, 35
```



Travessia Post-order

- Percorrer a árvore

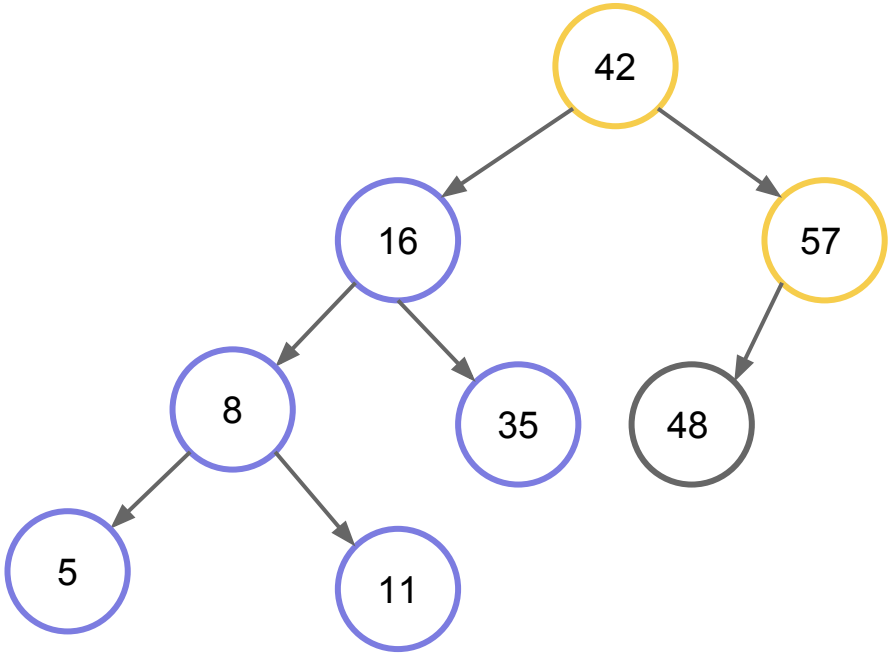
```
5, 11, 8, 35, 16
```



Travessia Post-order

- Percorrer a árvore

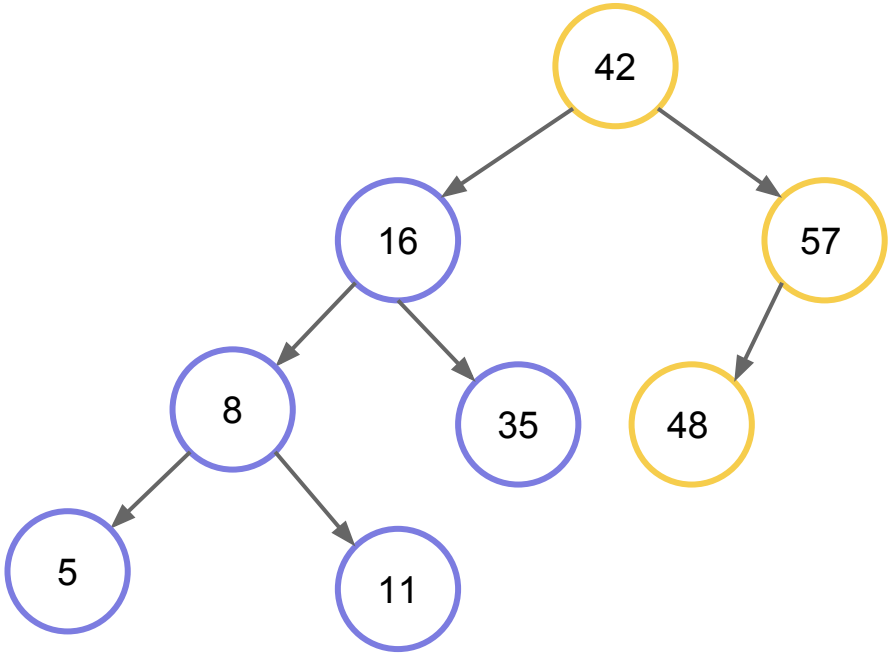
```
5, 11, 8, 35, 16
```



Travessia Post-order

- Percorrer a árvore

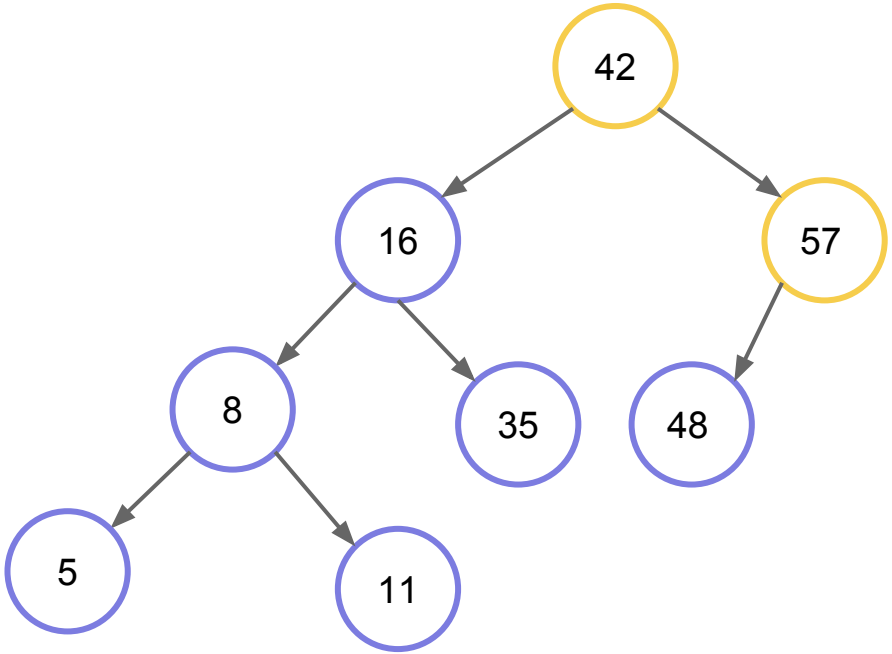
```
5, 11, 8, 35, 16
```



Travessia Post-order

- Percorrer a árvore

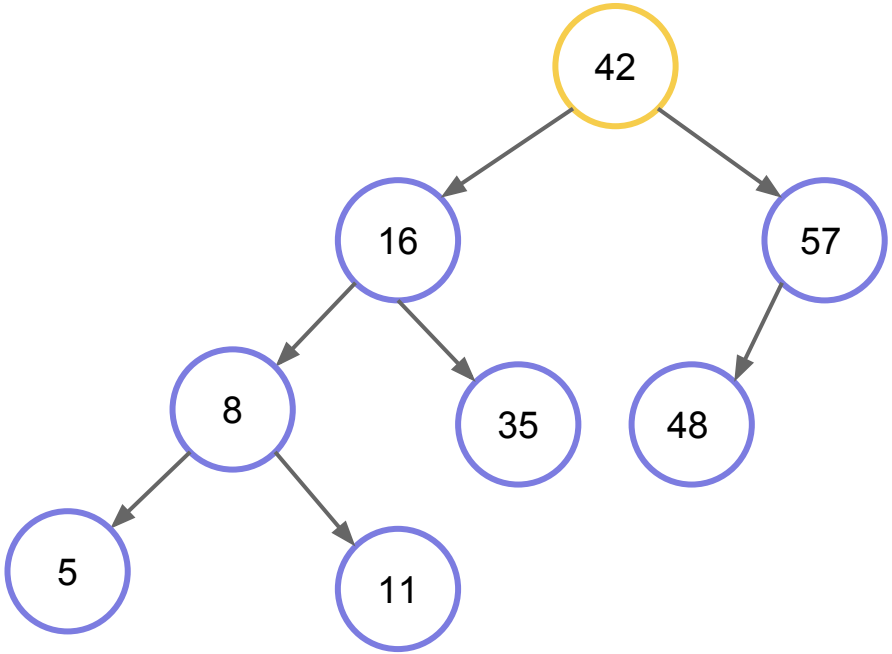
```
5, 11, 8, 35, 16, 48
```



Travessia Post-order

- Percorrer a árvore

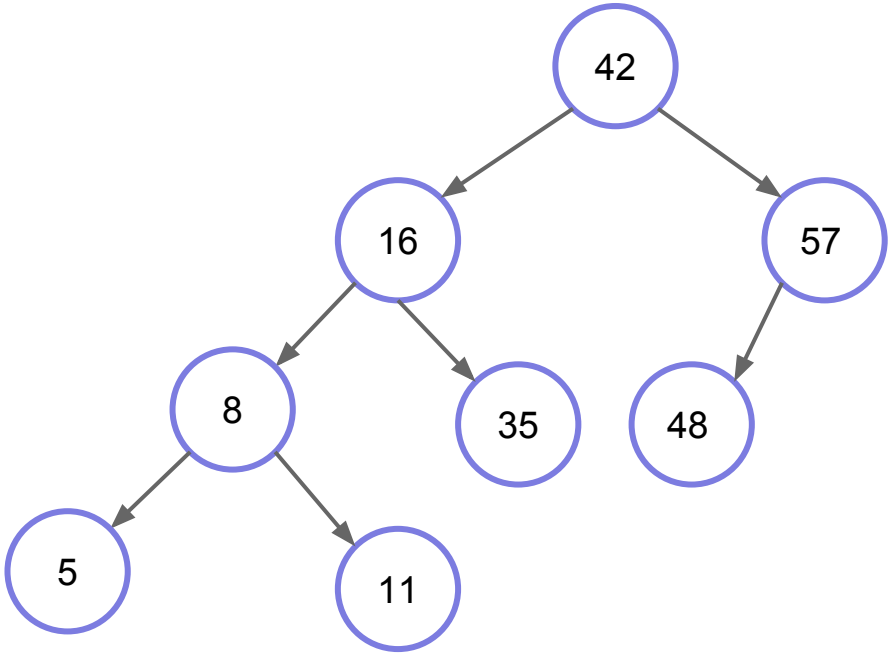
```
5, 11, 8, 35, 16,
48, 57
```



Travessia Post-order

- Percorrer a árvore

```
5, 11, 8, 35, 16,
48, 57, 42
```



Operações

Inserção 

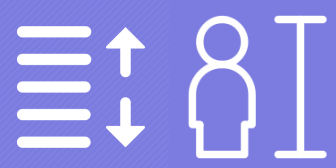
Busca 

Remoção 

Travessia 

visualgo

Características

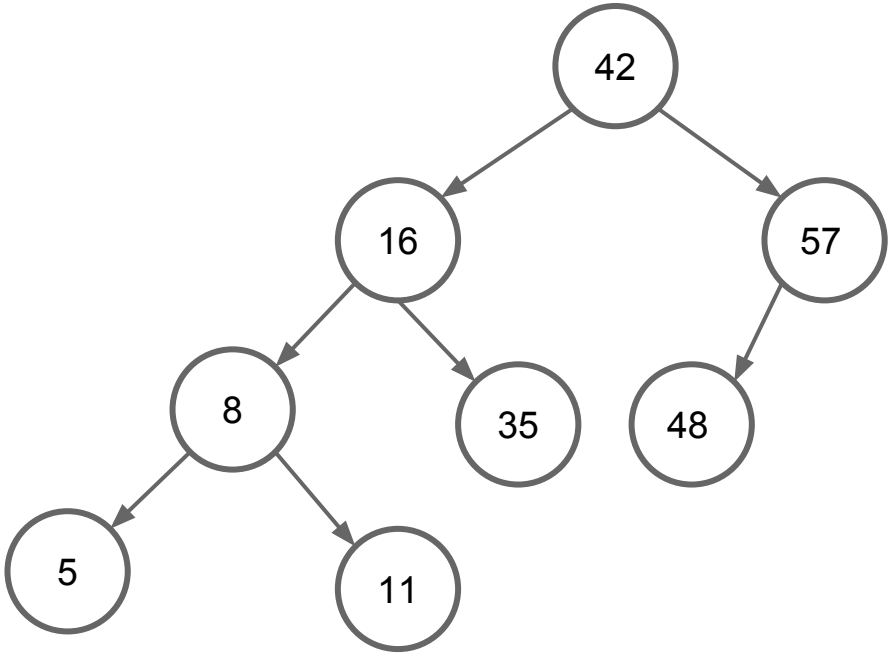


Nível do nó

Caminho até o nó

Achar o nó

Calcular passos

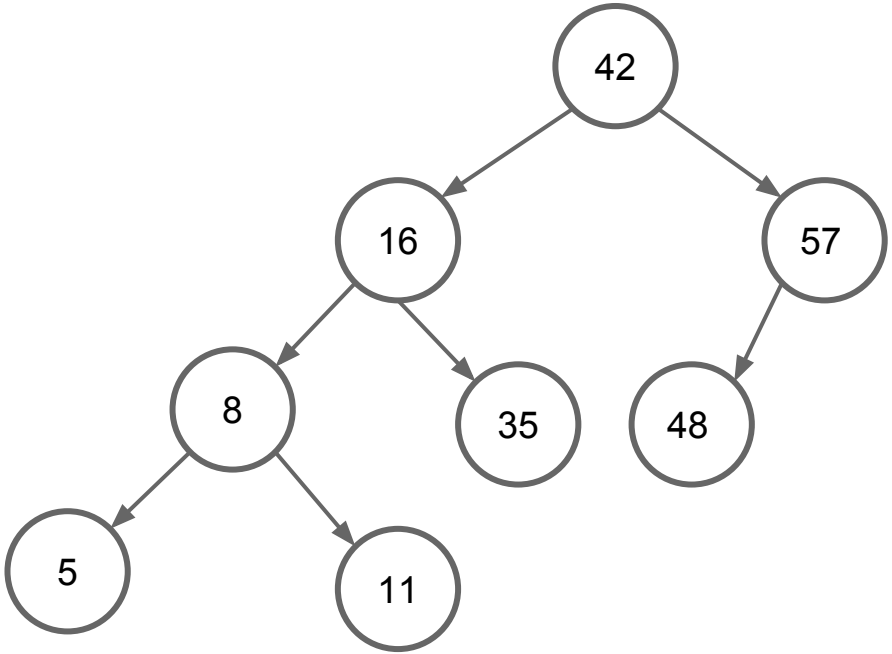


Nível do nó

Caminho até o nó

Achar o nó

Calcular passos

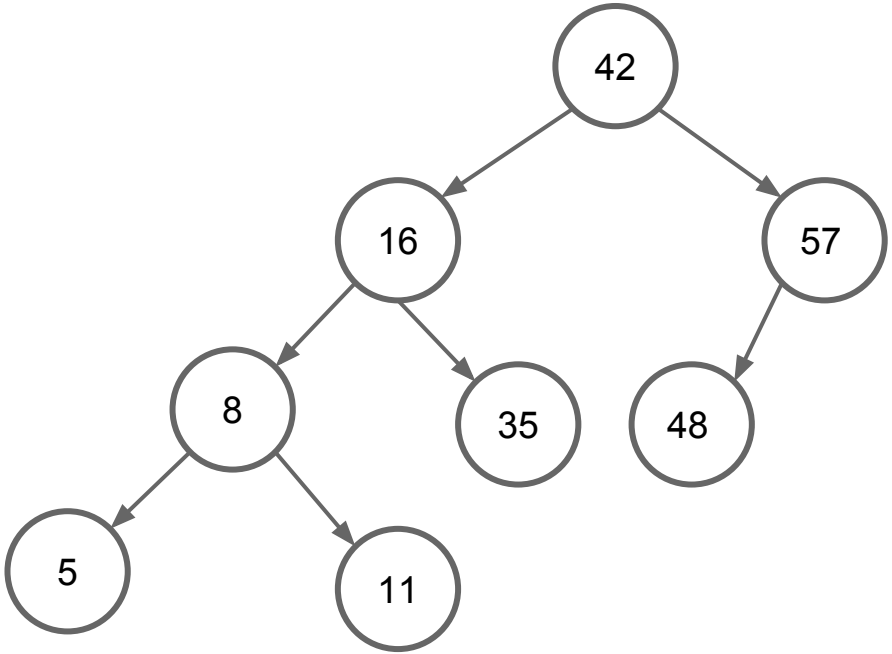


Nível do nó

Caminho até o nó

Achar o nó

Calcular passos

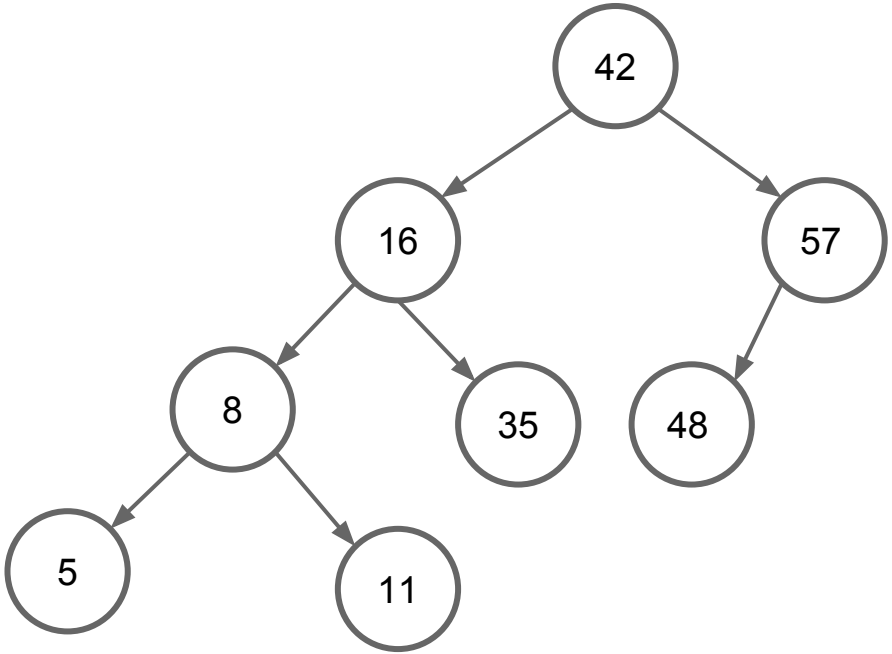


Nível do nó

Caminho até o nó

Achar o nó

Calcular passos



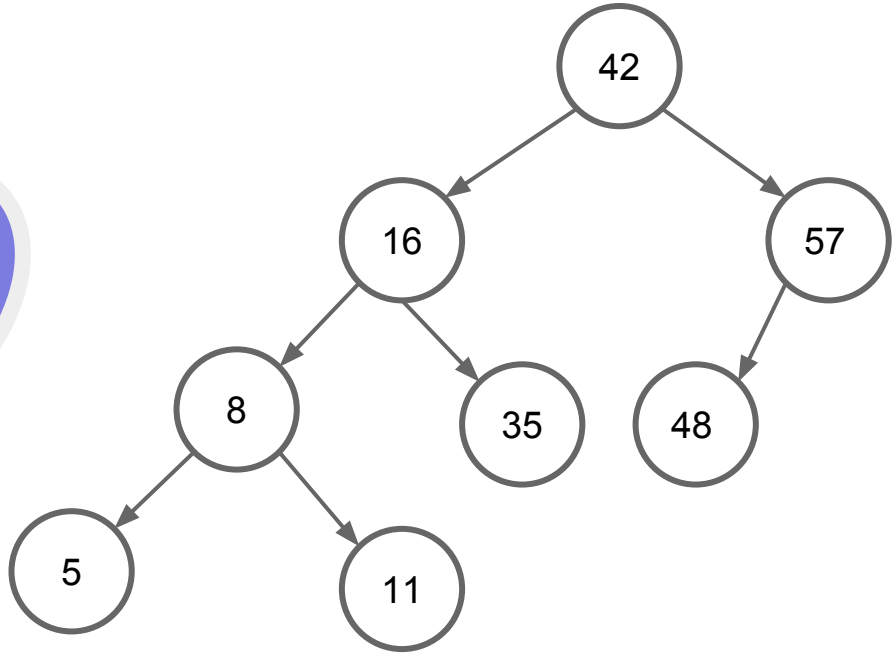
Nível do nó

Caminho até o nó

Achar o nó

Calcular passos

modificar busca

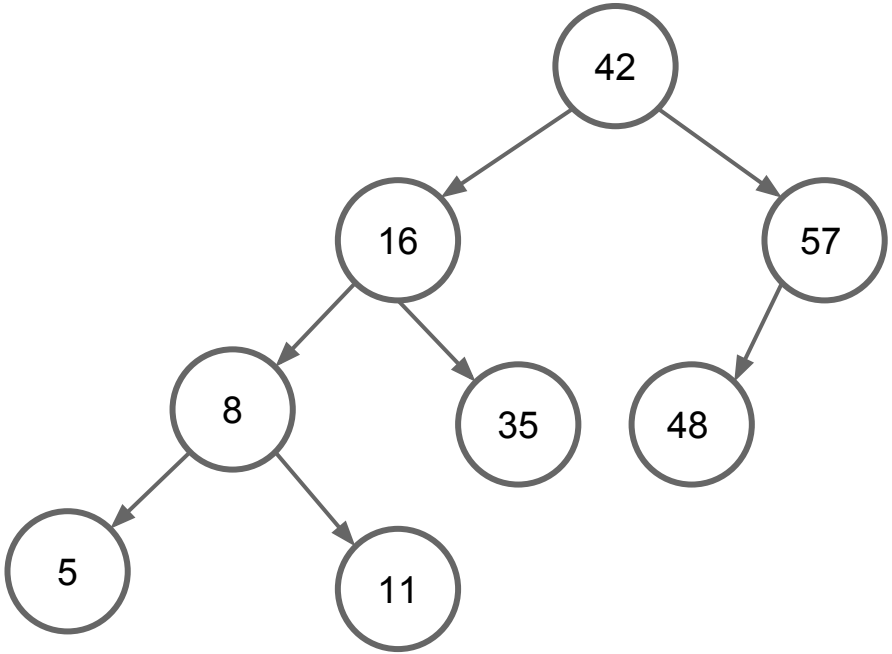


Altura da Árvore

Maior nível

Máximo entre subárvores

Atravessar a árvore

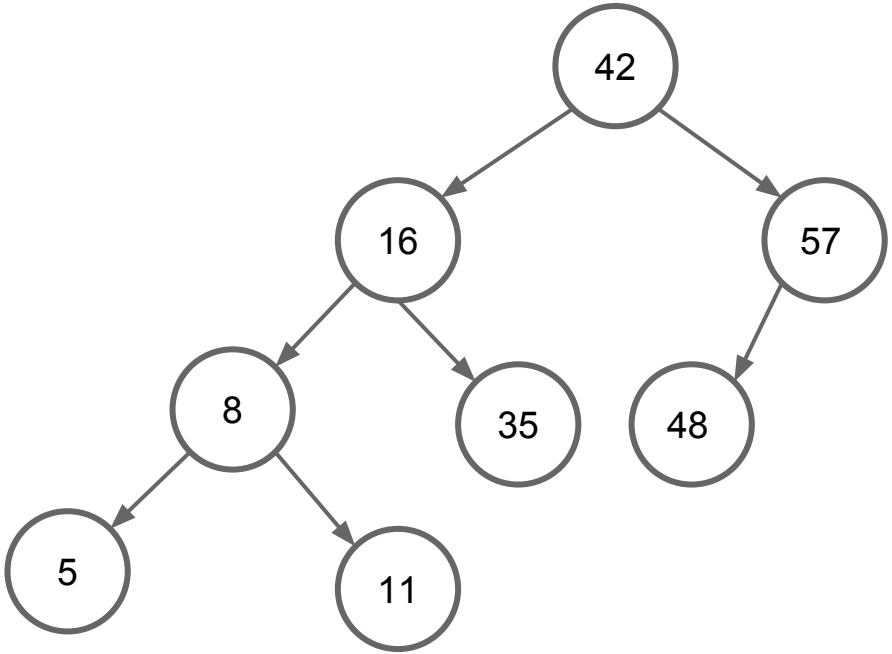


Altura da Árvore

Maior nível

Máximo entre subárvores

Atravessar a árvore

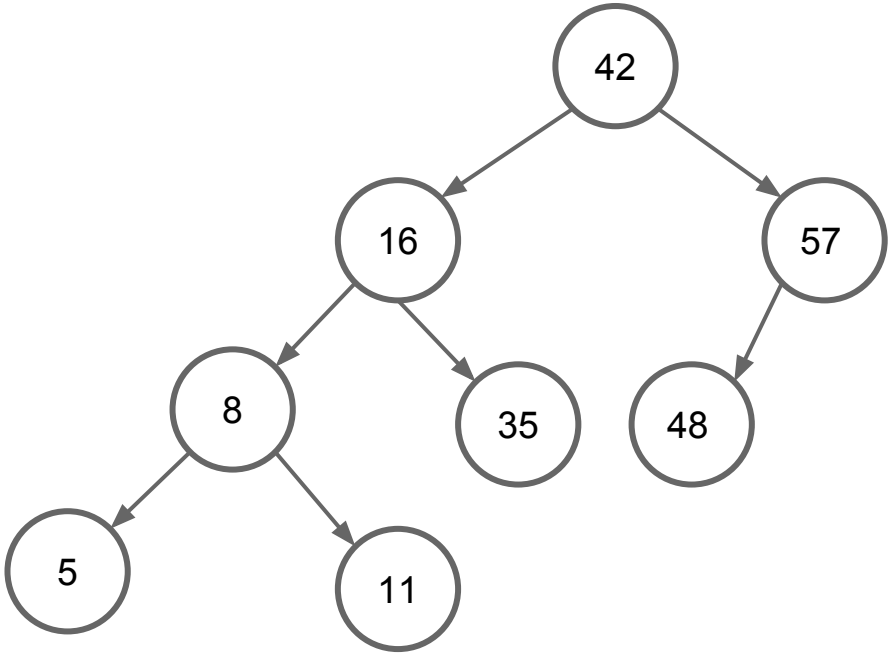


Altura da Árvore

Maior nível

Máximo entre subárvores

Atravessar a árvore

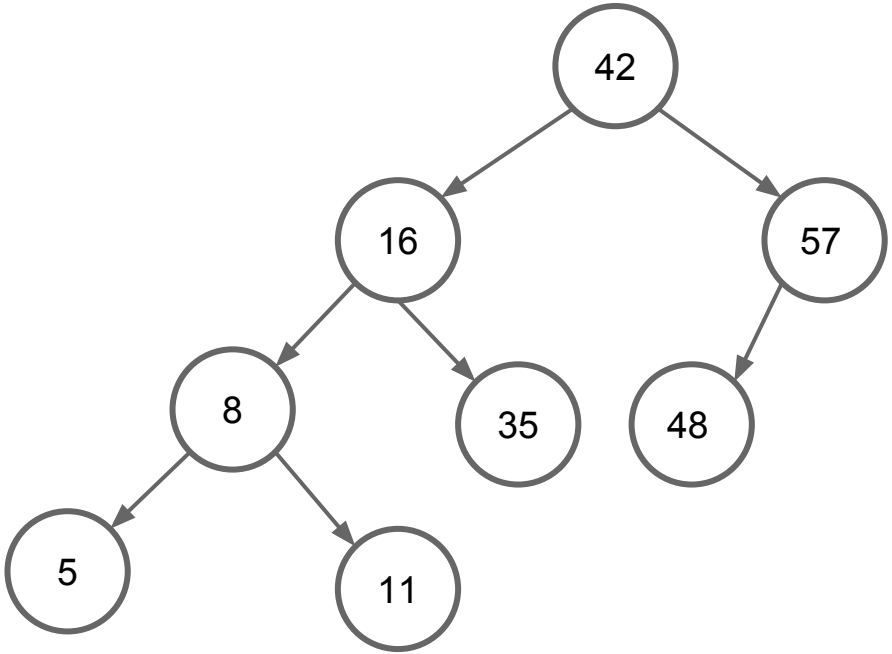


Altura da Árvore

Maior nível

Máximo entre subárvores

Atravessar a árvore



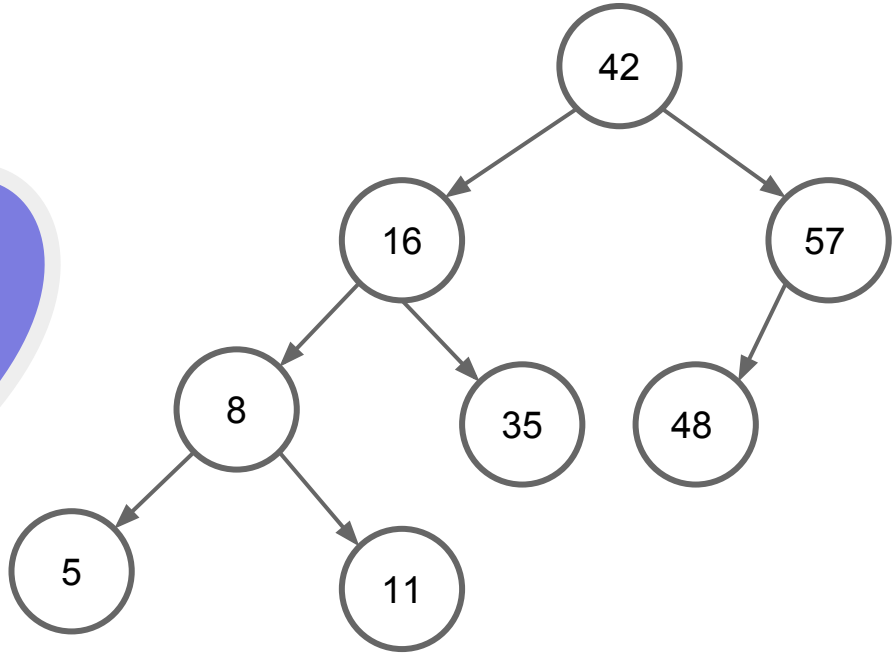
Altura da Árvore

Maior nível

Máximo entre subárvores

Atravessar a árvore

atravessar com nível



Árvores no C++

Maps

```
#include <map>

map<string, int> estados;

estados["Mato Grosso"] = 66;

estados["Bahia"] = 75;

estados.erase("Bahia");
```

Remoção

Maps

```
#include <map>

    for(auto chave_valor: estados)
        cout << chave_valor.first << " => "
                << chave_valor.second << "\n";
```

Sets

```
#include <set>

set<int > identificadores;

identificadores.insert(68);

identificadores.insert(33);

identificadores.erase(33);
```

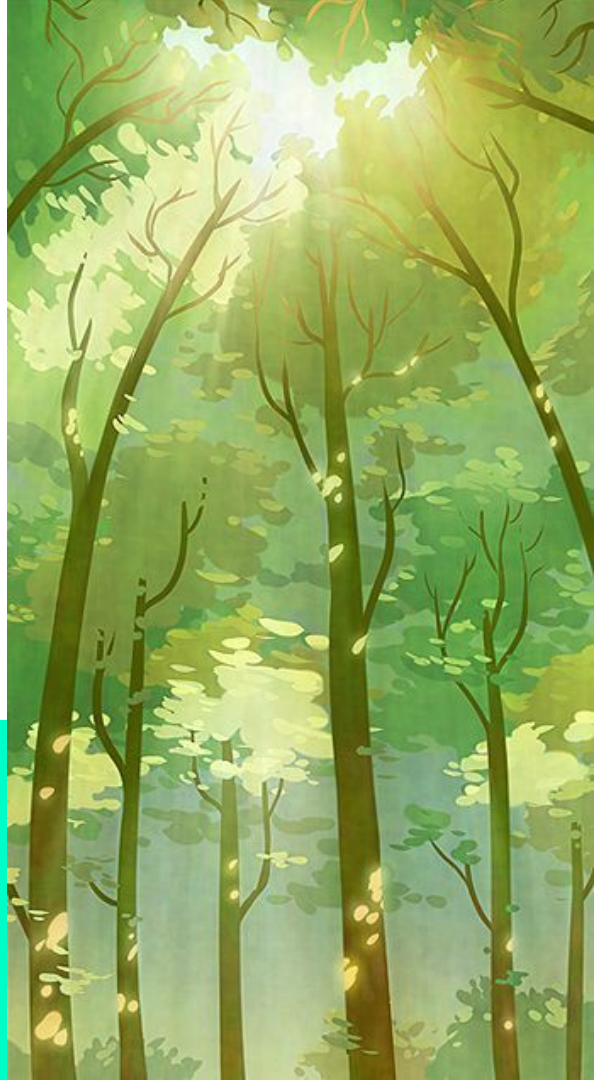
Inserção

Sets

```
#include <set>

for(auto item: identificadores)
    cout << item << "\n";
```

Árvores binárias



Prof: Bruna Moreira
brunamoreira@unb.br