Árvore Binária de Busca

Estrutura de Dados — QXD0110

Prof. Atílio Gomes Luiz
gomes.atilio@ufc.br

Universidade Federal do Ceará
Campus Quixadá

2º semestre/2025

Introdução

Tempo para Inserção, Remoção e Busca

Usando Listas Duplamente Encadeadas:

- Podemos inserir e remover em $O(1)$
- Mas buscar demora $O(n)$

Tempo para Inserção, Remoção e Busca

Usando Listas Duplamente Encadeadas:

- Podemos inserir e remover em $O(1)$
- Mas buscar demora $O(n)$

Se usarmos vetores não-ordenados:

- Podemos inserir e remover em $O(1)$

Tempo para Inserção, Remoção e Busca

Usando Listas Duplamente Encadeadas:

- Podemos inserir e remover em $O(1)$
- Mas buscar demora $O(n)$

Se usarmos vetores não-ordenados:

- Podemos inserir e remover em $O(1)$
  - insira no final
  - para remover, troque com o último e remova o último

Tempo para Inserção, Remoção e Busca

Usando Listas Duplamente Encadeadas:

- Podemos inserir e remover em $O(1)$
- Mas buscar demora $O(n)$

Se usarmos vetores não-ordenados:

- Podemos inserir e remover em $O(1)$
  - insira no final
  - para remover, troque com o último e remova o último
- Mas buscar demora $O(n)$

Tempo para Inserção, Remoção e Busca

Usando Listas Duplamente Encadeadas:

- Podemos inserir e remover em $O(1)$
- Mas buscar demora $O(n)$

Se usarmos vetores não-ordenados:

- Podemos inserir e remover em $O(1)$
  - insira no final
  - para remover, troque com o último e remova o último
- Mas buscar demora $O(n)$

Se usarmos vetores ordenados:

- Podemos buscar em $O(\lg n)$ usando Busca binária
- Mas inserir e remover leva $O(n)$

Tempo para Inserção, Remoção e Busca

Usando Listas Duplamente Encadeadas:

- Podemos inserir e remover em $O(1)$
- Mas buscar demora $O(n)$

Se usarmos vetores não-ordenados:

- Podemos inserir e remover em $O(1)$
  - insira no final
  - para remover, troque com o último e remova o último
- Mas buscar demora $O(n)$

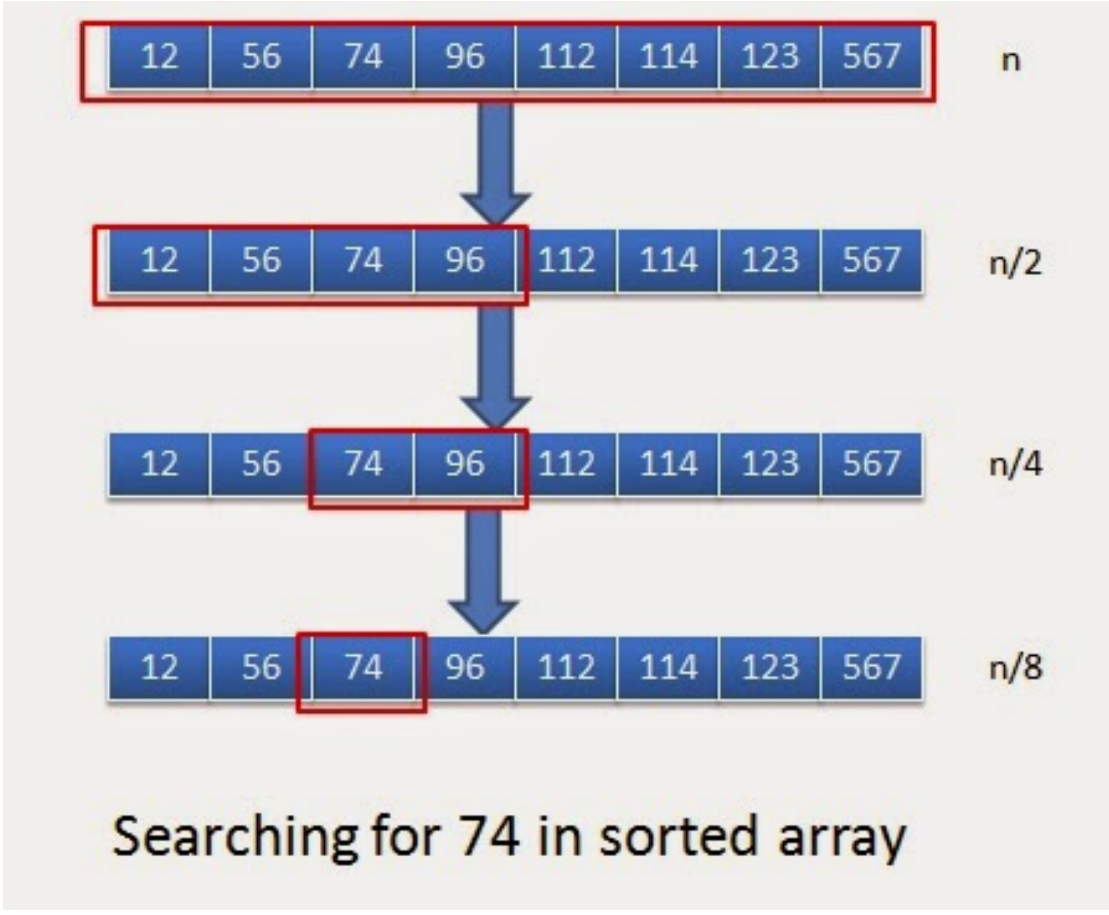
Se usarmos vetores ordenados:

- Podemos buscar em $O(\lg n)$ usando Busca binária
- Mas inserir e remover leva $O(n)$

Veremos árvores binárias de busca

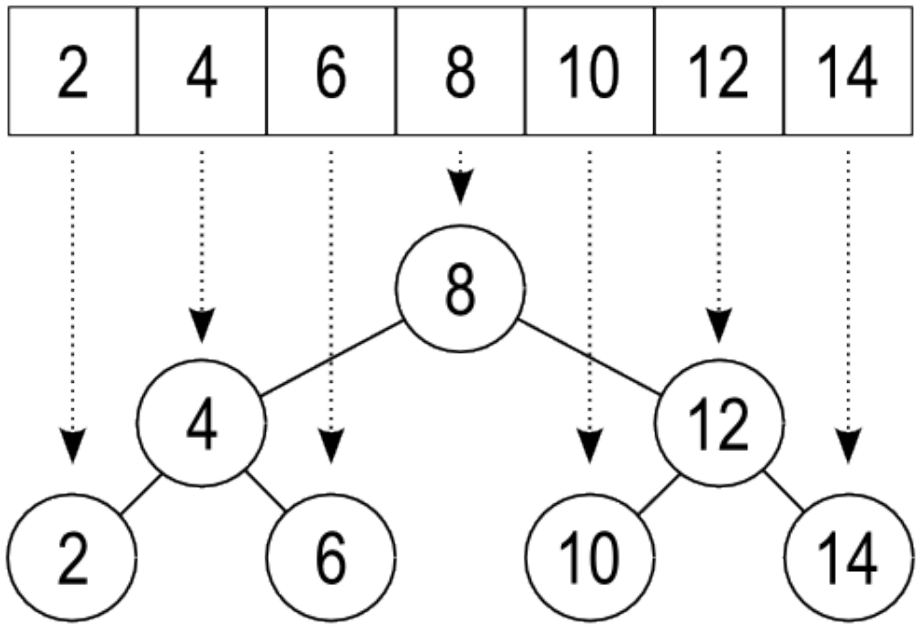
- Inserção, Remoção e Busca levam $O(h)$ onde h é a altura da árvore

Inspiração: Busca binária



Inspiração: Busca binária

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ



Árvore Binária de Busca

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

Definição

Seja $S = \{s_1, \ldots, s_n\}$ um conjunto ordenável.

Uma Árvore Binária de Busca é uma árvore binária rotulada T com as seguintes características:

Árvore Binária de Busca

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

Definição

Seja $S = \{s_1, \ldots, s_n\}$ um conjunto ordenável.

Uma Árvore Binária de Busca é uma árvore binária rotulada T com as seguintes características:

- T possui n nós. Cada nó v de T está rotulado com um elemento $s_j \in S$ e possui como rótulo o valor $v.chave = s_j$.

Árvore Binária de Busca

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

Definição

Seja $S = \{s_1, \ldots, s_n\}$ um conjunto ordenável.

Uma Árvore Binária de Busca é uma árvore binária rotulada T com as seguintes características:

- T possui n nós. Cada nó v de T está rotulado com um elemento $s_j \in S$ e possui como rótulo o valor $v.chave = s_j$.
- Seja v um nó de T. Se w pertence à subárvore esquerda de v, então $w.chave < v.chave$.

Árvore Binária de Busca

Definição

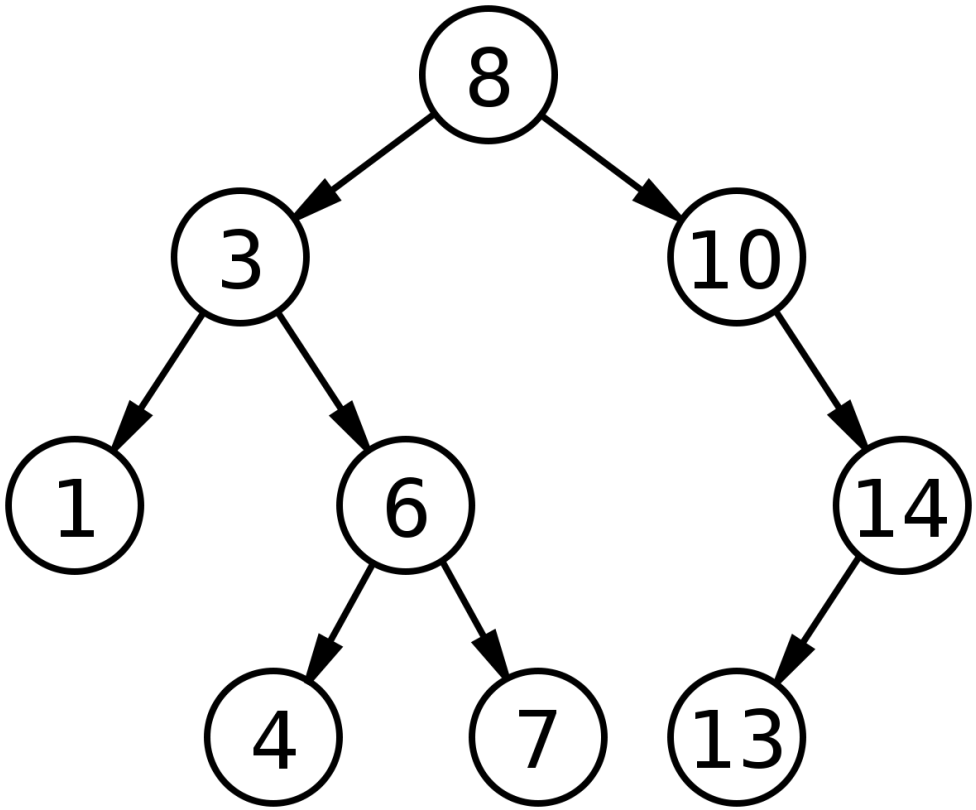
Seja $S = \{s_1, \ldots, s_n\}$ um conjunto ordenável.

Uma Árvore Binária de Busca é uma árvore binária rotulada T com as seguintes características:

- T possui n nós. Cada nó v de T está rotulado com um elemento $s_j \in S$ e possui como rótulo o valor $v.chave = s_j$.
- Seja v um nó de T. Se w pertence à subárvore esquerda de v, então $w.chave < v.chave$.

  Analogamente, se w pertence à subárvore direita de v, então $w.chave > v.chave$.

Exemplo



Árvore Binária de Busca

Implementação em C

Implementação

- Usaremos recursão na implementação das operações de árvore.

Implementação

- Usaremos recursão na implementação das operações de árvore.
- O nó da árvore é implementado como um struct com três campos:
  - uma chave inteira e dois ponteiros, um para o filho esquerdo e outro para o filho direito.

```
1 struct no_arv {
2     int chave;
3     struct no_arv *esq;
4     struct no_arv *dir;
5 };
6
7 typedef struct no_arv NoArv;
```

Busca por um valor

A ideia é semelhante àquela da busca binária:

Busca por um valor

A ideia é semelhante àquela da busca binária:

- Ou o valor a ser buscado está na raiz da árvore

Busca por um valor

A ideia é semelhante àquela da busca binária:

- Ou o valor a ser buscado está na raiz da árvore
- Ou é menor do que o valor da raiz

Busca por um valor

A ideia é semelhante àquela da busca binária:

- Ou o valor a ser buscado está na raiz da árvore
- Ou é menor do que o valor da raiz
  - Se estiver na árvore, está na subárvore esquerda

Busca por um valor

A ideia é semelhante àquela da busca binária:

- Ou o valor a ser buscado está na raiz da árvore
- Ou é menor do que o valor da raiz
  - Se estiver na árvore, está na subárvore esquerda
- Ou é maior do que o valor da raiz

Busca por um valor

A ideia é semelhante àquela da busca binária:

- Ou o valor a ser buscado está na raiz da árvore
- Ou é menor do que o valor da raiz
  - Se estiver na árvore, está na subárvore esquerda
- Ou é maior do que o valor da raiz
  - Se estiver na árvore, está na subárvore direita

Busca por um valor

A ideia é semelhante àquela da busca binária:

- Ou o valor a ser buscado está na raiz da árvore
- Ou é menor do que o valor da raiz
  - Se estiver na árvore, está na subárvore esquerda
- Ou é maior do que o valor da raiz
  - Se estiver na árvore, está na subárvore direita

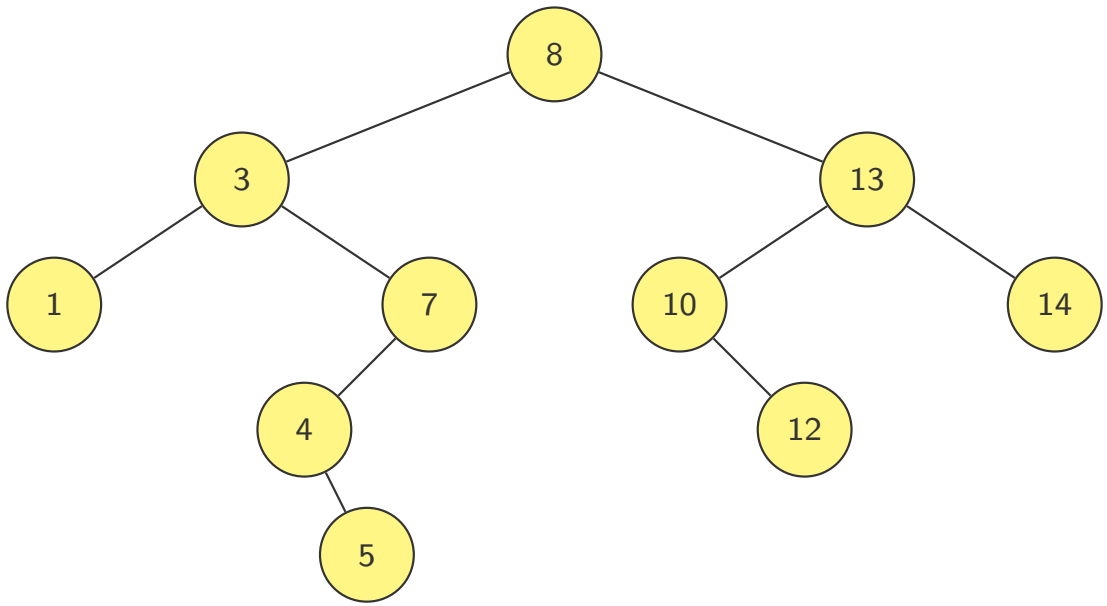
Ex: Buscando por 4

Busca por um valor

A ideia é semelhante àquela da busca binária:

- Ou o valor a ser buscado está na raiz da árvore
- Ou é menor do que o valor da raiz
  - Se estiver na árvore, está na subárvore esquerda
- Ou é maior do que o valor da raiz
  - Se estiver na árvore, está na subárvore direita

Ex: Buscando por 4

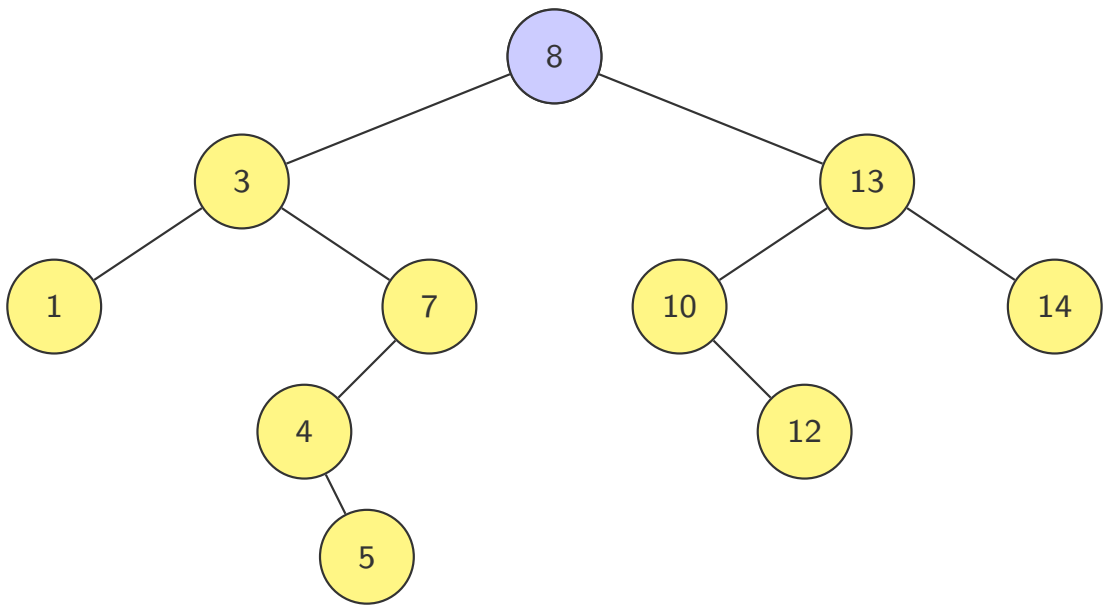


Busca por um valor

A ideia é semelhante àquela da busca binária:

- Ou o valor a ser buscado está na raiz da árvore
- Ou é menor do que o valor da raiz
  - Se estiver na árvore, está na subárvore esquerda
- Ou é maior do que o valor da raiz
  - Se estiver na árvore, está na subárvore direita

Ex: Buscando por 4

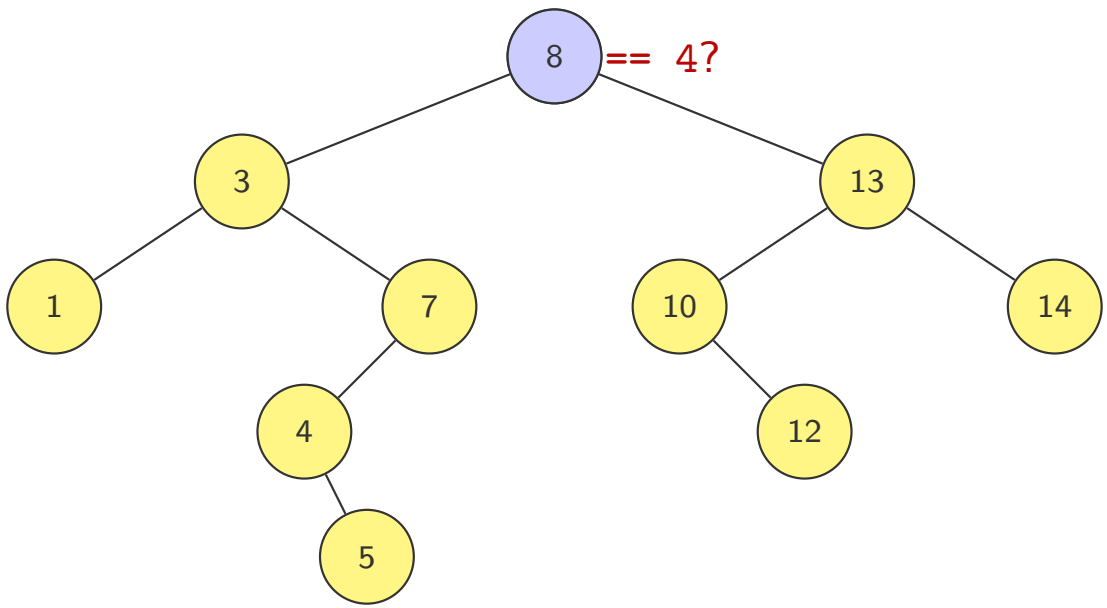


Busca por um valor

A ideia é semelhante àquela da busca binária:

- Ou o valor a ser buscado está na raiz da árvore
- Ou é menor do que o valor da raiz
  - Se estiver na árvore, está na subárvore esquerda
- Ou é maior do que o valor da raiz
  - Se estiver na árvore, está na subárvore direita

Ex: Buscando por 4

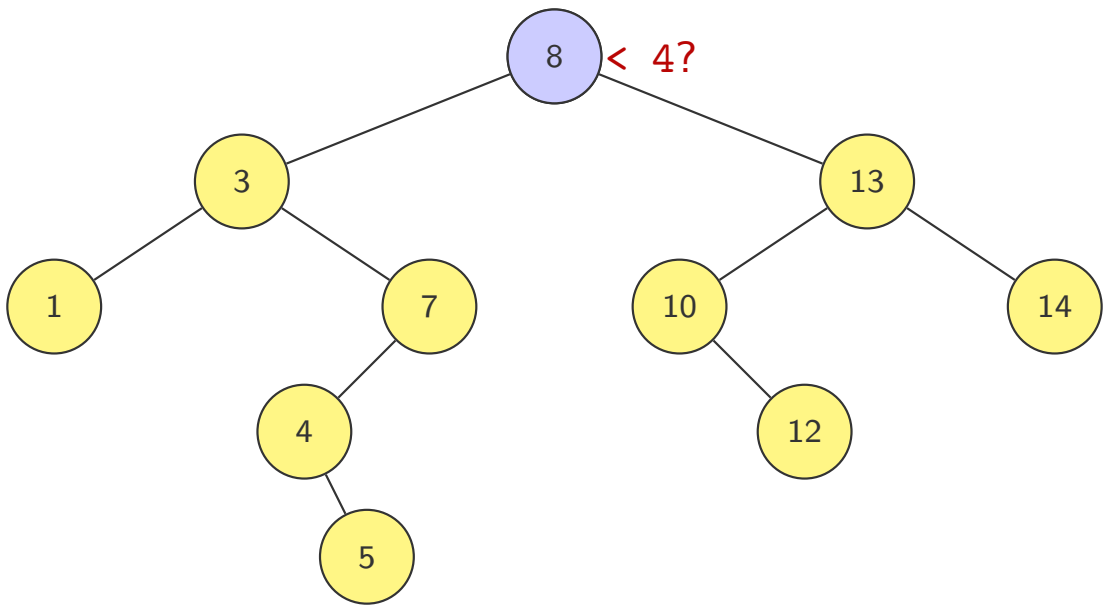


Busca por um valor

A ideia é semelhante àquela da busca binária:

- Ou o valor a ser buscado está na raiz da árvore
- Ou é menor do que o valor da raiz
  - Se estiver na árvore, está na subárvore esquerda
- Ou é maior do que o valor da raiz
  - Se estiver na árvore, está na subárvore direita

Ex: Buscando por 4

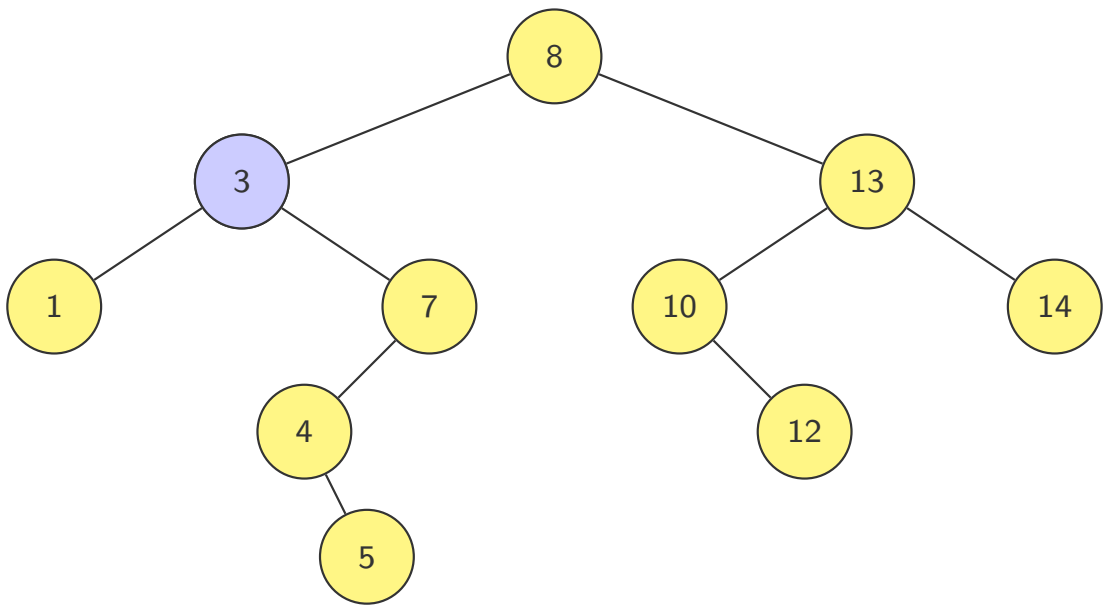


Busca por um valor

A ideia é semelhante àquela da busca binária:

- Ou o valor a ser buscado está na raiz da árvore
- Ou é menor do que o valor da raiz
  - Se estiver na árvore, está na subárvore esquerda
- Ou é maior do que o valor da raiz
  - Se estiver na árvore, está na subárvore direita

Ex: Buscando por 4

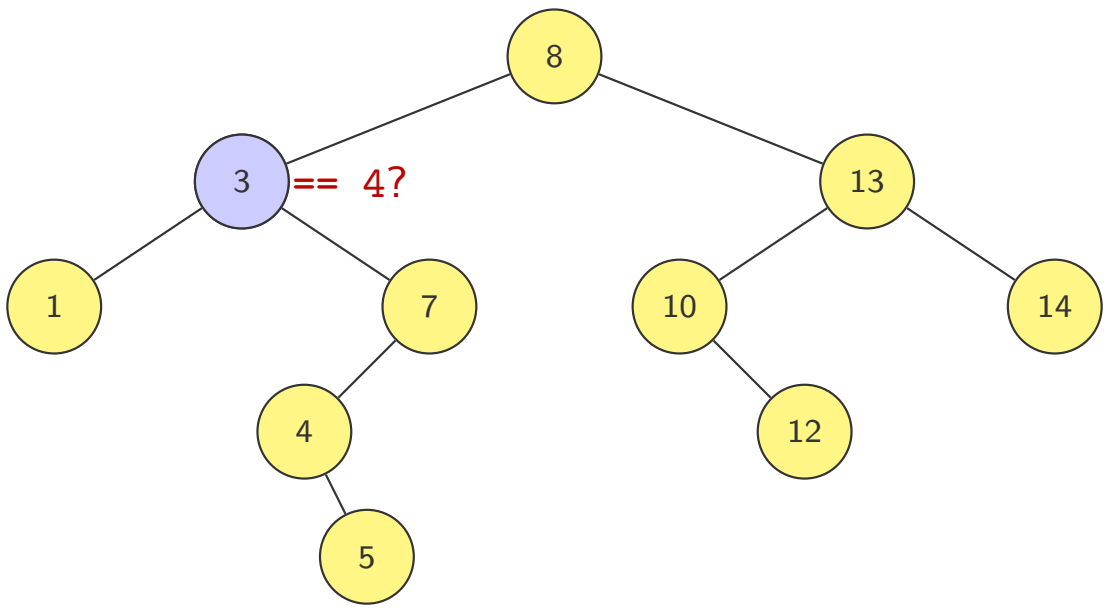


Busca por um valor

A ideia é semelhante àquela da busca binária:

- Ou o valor a ser buscado está na raiz da árvore
- Ou é menor do que o valor da raiz
  - Se estiver na árvore, está na subárvore esquerda
- Ou é maior do que o valor da raiz
  - Se estiver na árvore, está na subárvore direita

Ex: Buscando por 4

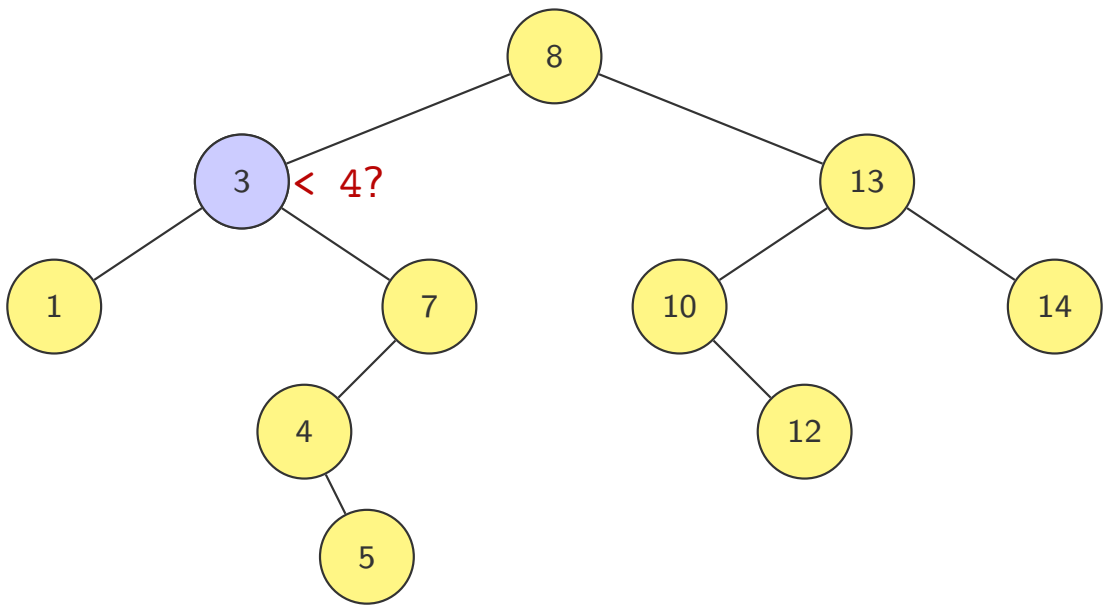


Busca por um valor

A ideia é semelhante àquela da busca binária:

- Ou o valor a ser buscado está na raiz da árvore
- Ou é menor do que o valor da raiz
  - Se estiver na árvore, está na subárvore esquerda
- Ou é maior do que o valor da raiz
  - Se estiver na árvore, está na subárvore direita

Ex: Buscando por 4

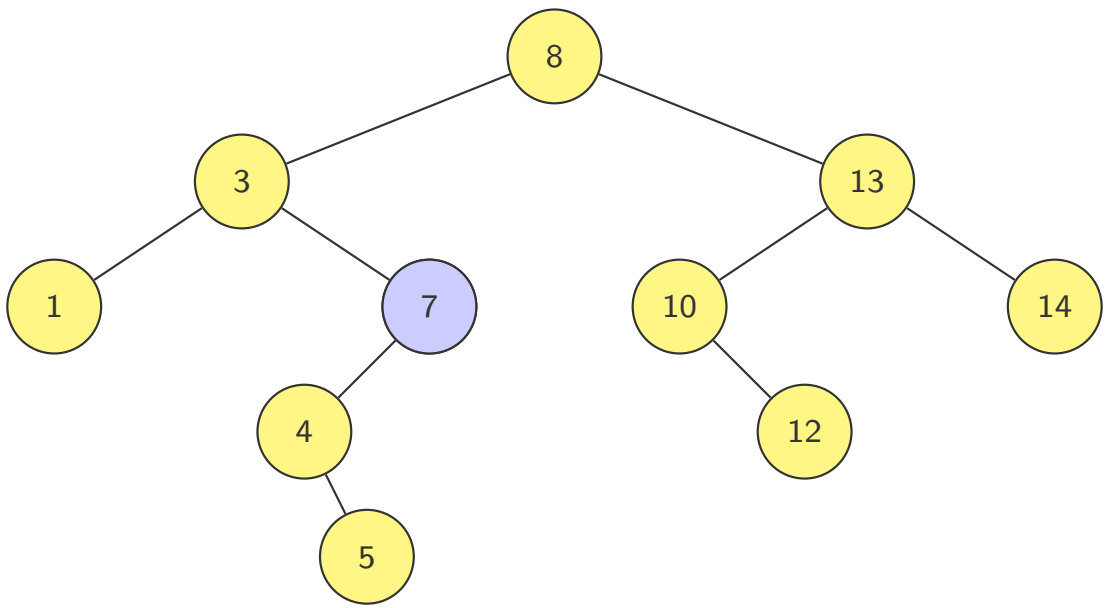


Busca por um valor

A ideia é semelhante àquela da busca binária:

- Ou o valor a ser buscado está na raiz da árvore
- Ou é menor do que o valor da raiz
  - Se estiver na árvore, está na subárvore esquerda
- Ou é maior do que o valor da raiz
  - Se estiver na árvore, está na subárvore direita

Ex: Buscando por 4

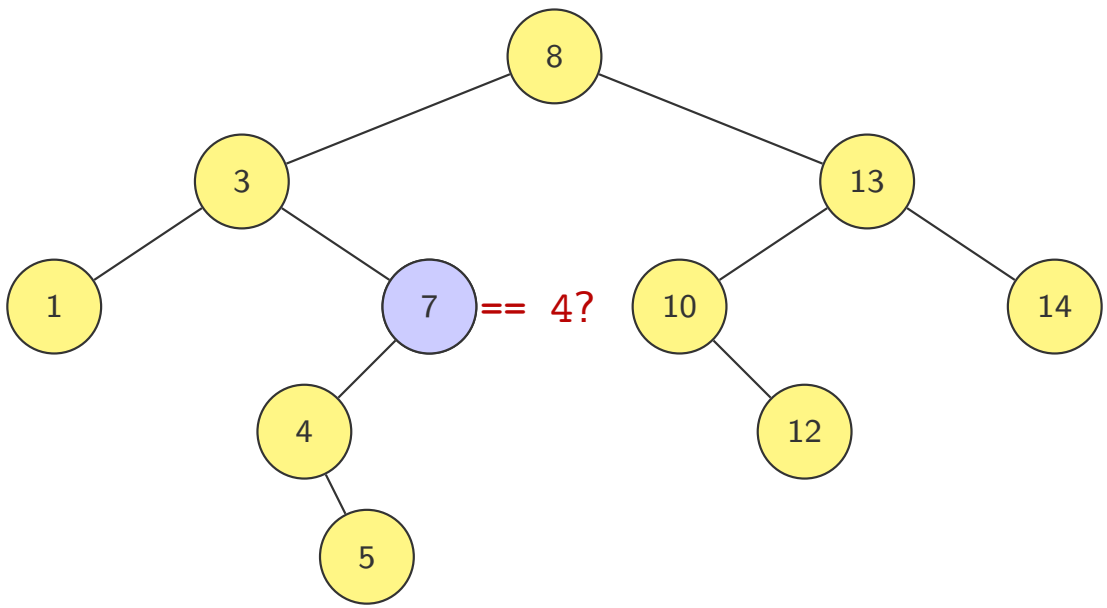


Busca por um valor

A ideia é semelhante àquela da busca binária:

- Ou o valor a ser buscado está na raiz da árvore
- Ou é menor do que o valor da raiz
  - Se estiver na árvore, está na subárvore esquerda
- Ou é maior do que o valor da raiz
  - Se estiver na árvore, está na subárvore direita

Ex: Buscando por 4

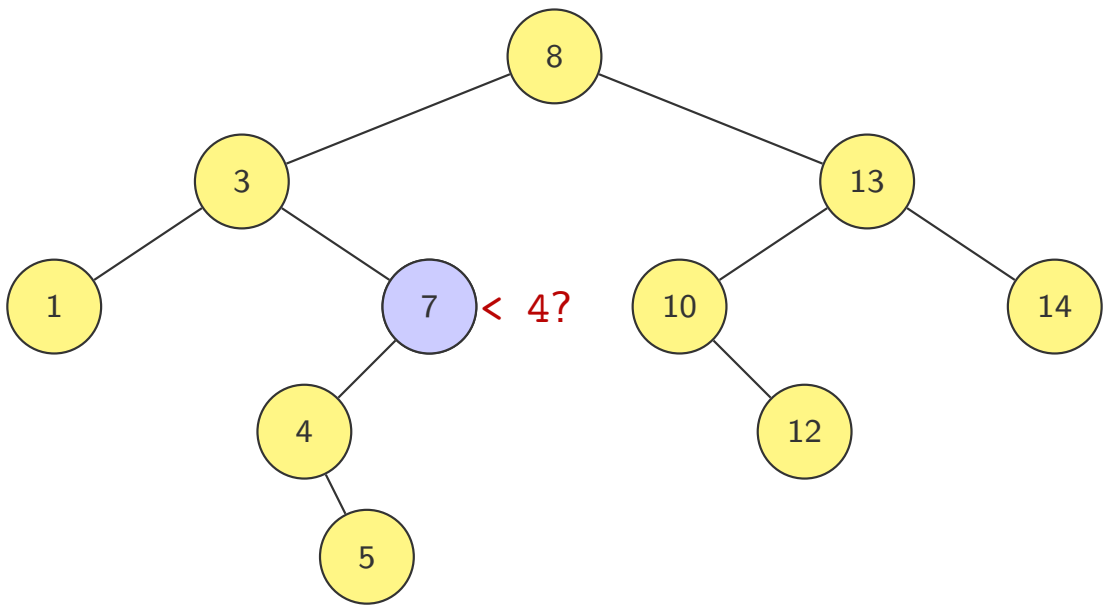


Busca por um valor

A ideia é semelhante àquela da busca binária:

- Ou o valor a ser buscado está na raiz da árvore
- Ou é menor do que o valor da raiz
  - Se estiver na árvore, está na subárvore esquerda
- Ou é maior do que o valor da raiz
  - Se estiver na árvore, está na subárvore direita

Ex: Buscando por 4

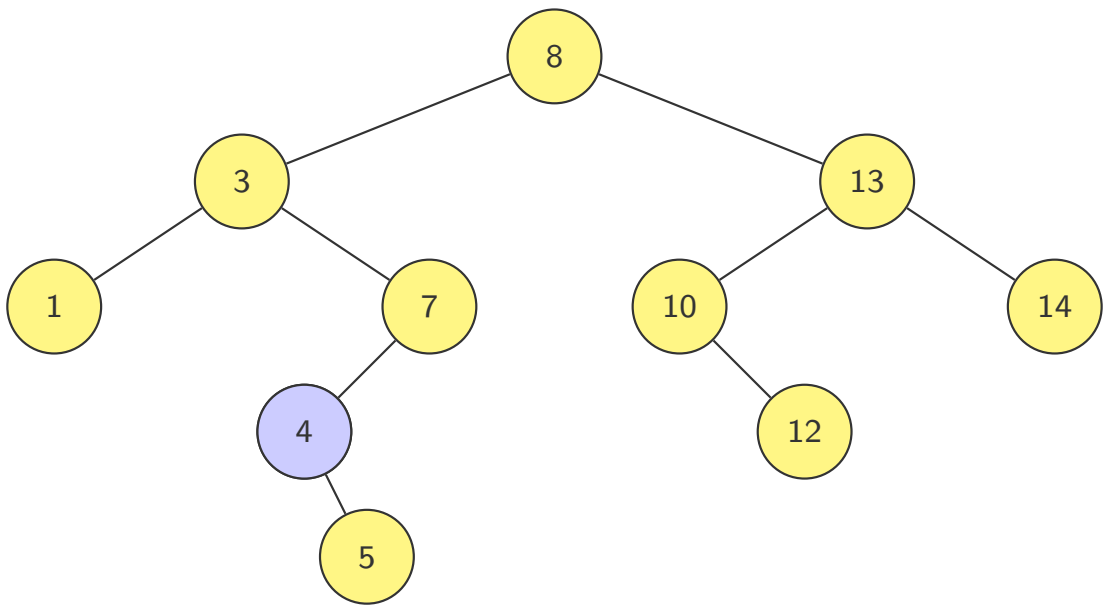


Busca por um valor

A ideia é semelhante àquela da busca binária:

- Ou o valor a ser buscado está na raiz da árvore
- Ou é menor do que o valor da raiz
  - Se estiver na árvore, está na subárvore esquerda
- Ou é maior do que o valor da raiz
  - Se estiver na árvore, está na subárvore direita

Ex: Buscando por 4

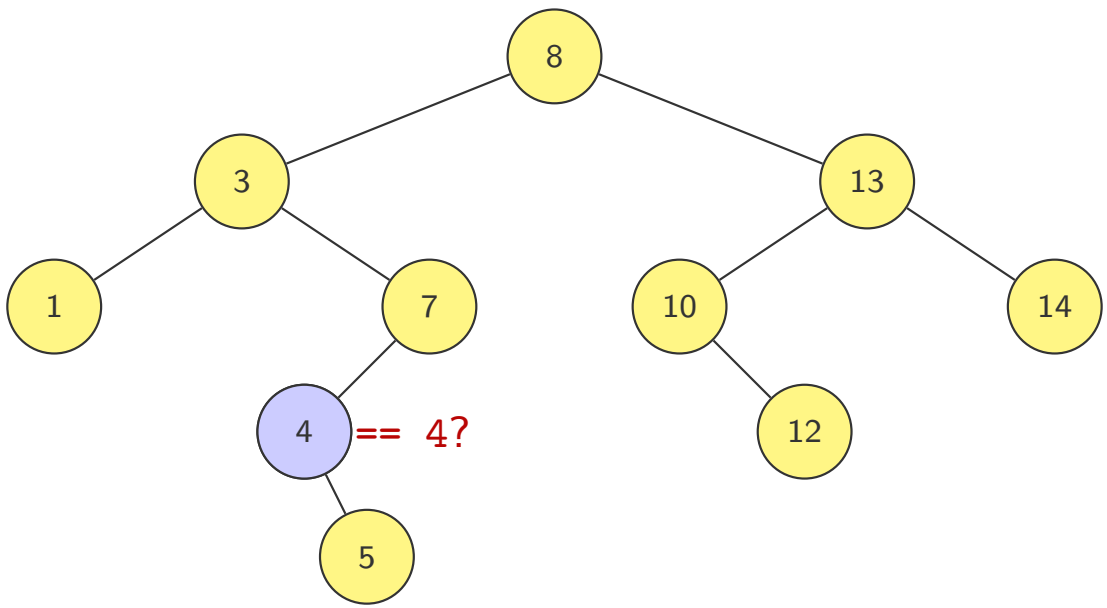


Busca por um valor

A ideia é semelhante àquela da busca binária:

- Ou o valor a ser buscado está na raiz da árvore
- Ou é menor do que o valor da raiz
  - Se estiver na árvore, está na subárvore esquerda
- Ou é maior do que o valor da raiz
  - Se estiver na árvore, está na subárvore direita

Ex: Buscando por 4

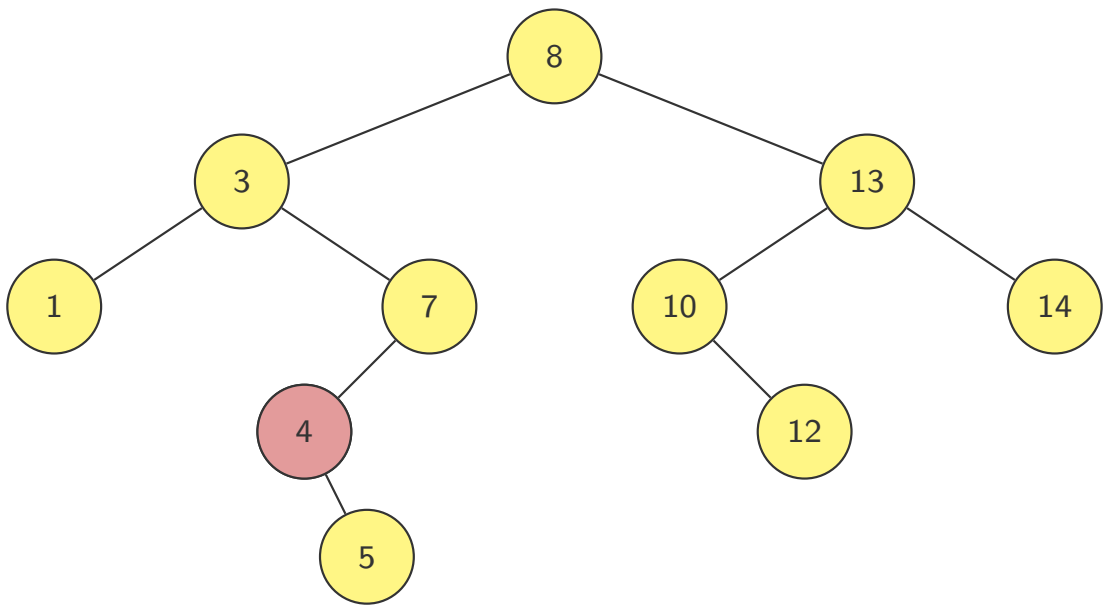


Busca por um valor

A ideia é semelhante àquela da busca binária:

- Ou o valor a ser buscado está na raiz da árvore
- Ou é menor do que o valor da raiz
  - Se estiver na árvore, está na subárvore esquerda
- Ou é maior do que o valor da raiz
  - Se estiver na árvore, está na subárvore direita

Ex: Buscando por 4

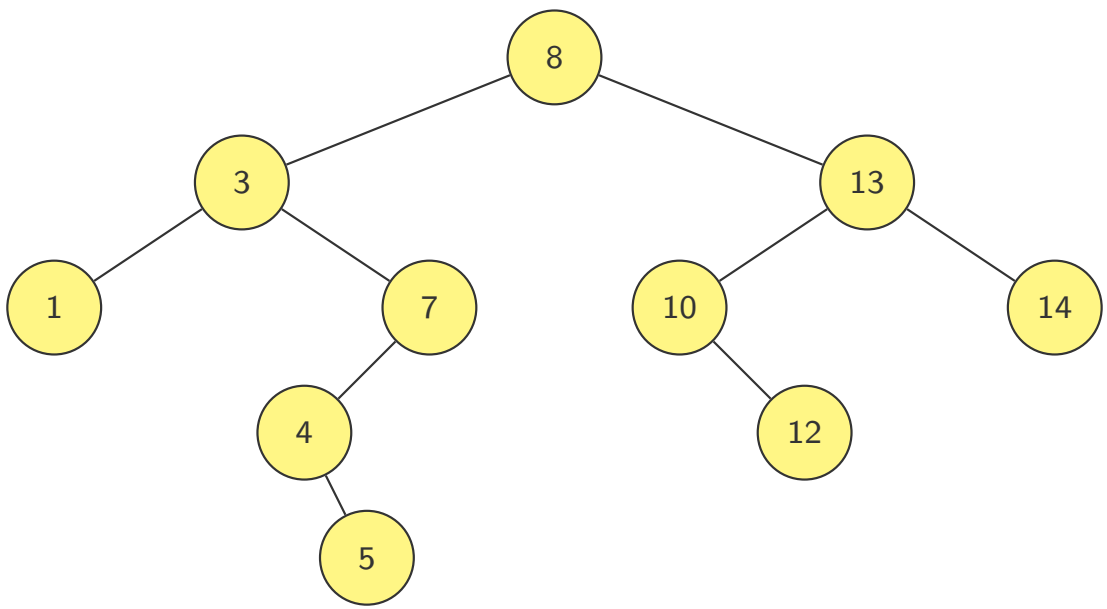


Busca por um valor

A ideia é semelhante a da busca binária:

- Ou o valor a ser buscado está na raiz da árvore
- Ou é menor do que o valor da raiz
  - Se estiver na árvore, está na subárvore esquerda
- Ou é maior do que o valor da raiz
  - Se estiver na árvore, está na subárvore direita

Ex: Buscando por 11

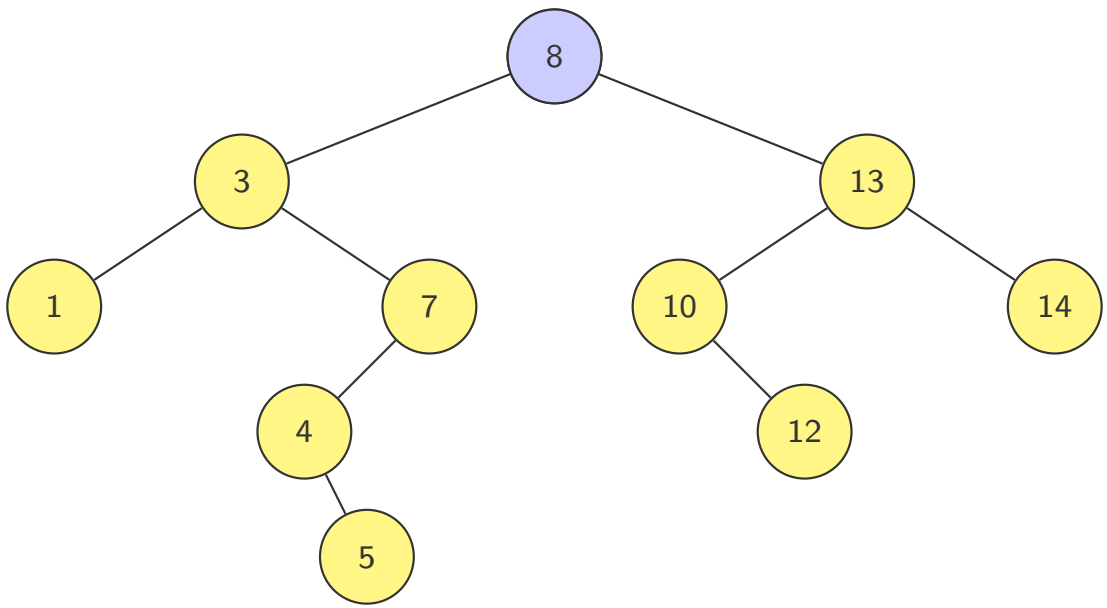


Busca por um valor

A ideia é semelhante a da busca binária:

- Ou o valor a ser buscado está na raiz da árvore
- Ou é menor do que o valor da raiz
  - Se estiver na árvore, está na subárvore esquerda
- Ou é maior do que o valor da raiz
  - Se estiver na árvore, está na subárvore direita

Ex: Buscando por 11

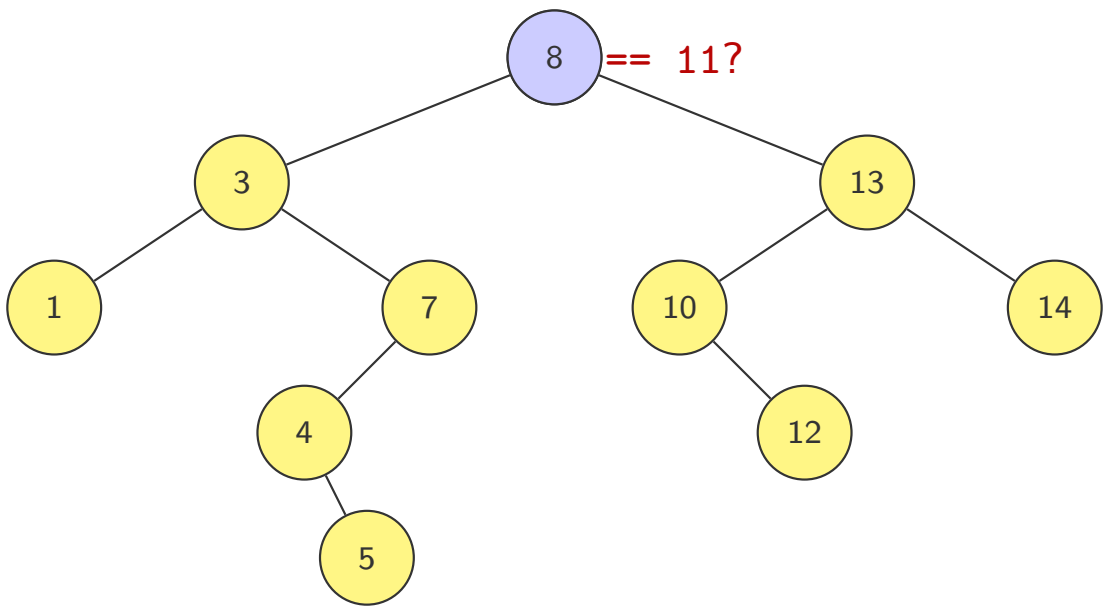


Busca por um valor

A ideia é semelhante a da busca binária:

- Ou o valor a ser buscado está na raiz da árvore
- Ou é menor do que o valor da raiz
  - Se estiver na árvore, está na subárvore esquerda
- Ou é maior do que o valor da raiz
  - Se estiver na árvore, está na subárvore direita

Ex: Buscando por 11

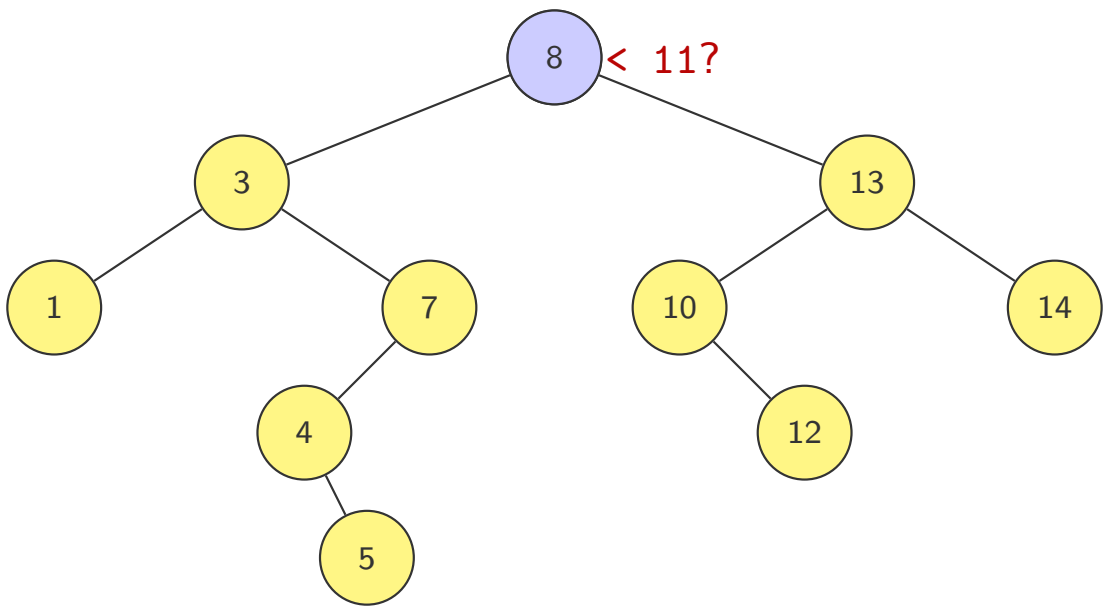


Busca por um valor

A ideia é semelhante a da busca binária:

- Ou o valor a ser buscado está na raiz da árvore
- Ou é menor do que o valor da raiz
  - Se estiver na árvore, está na subárvore esquerda
- Ou é maior do que o valor da raiz
  - Se estiver na árvore, está na subárvore direita

Ex: Buscando por 11

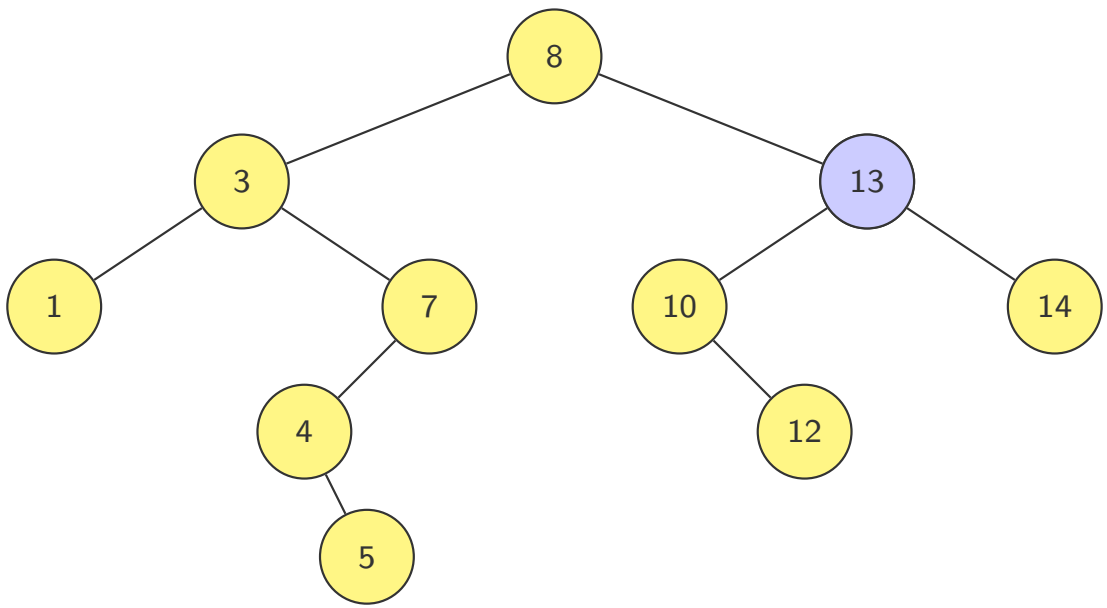


Busca por um valor

A ideia é semelhante a da busca binária:

- Ou o valor a ser buscado está na raiz da árvore
- Ou é menor do que o valor da raiz
  - Se estiver na árvore, está na subárvore esquerda
- Ou é maior do que o valor da raiz
  - Se estiver na árvore, está na subárvore direita

Ex: Buscando por 11

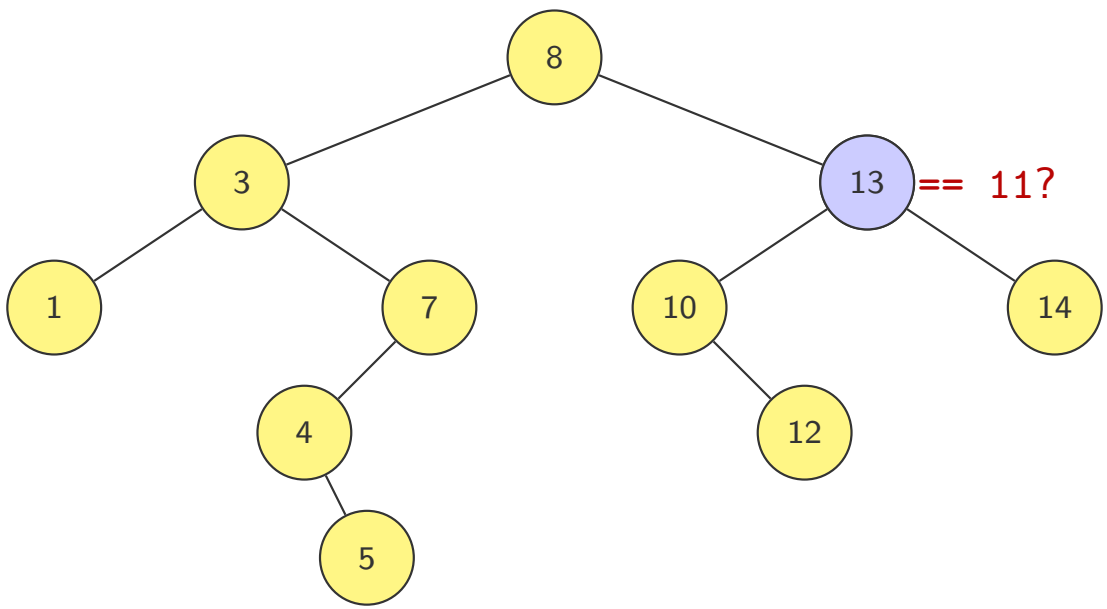


Busca por um valor

A ideia é semelhante a da busca binária:

- Ou o valor a ser buscado está na raiz da árvore
- Ou é menor do que o valor da raiz
  - Se estiver na árvore, está na subárvore esquerda
- Ou é maior do que o valor da raiz
  - Se estiver na árvore, está na subárvore direita

Ex: Buscando por 11

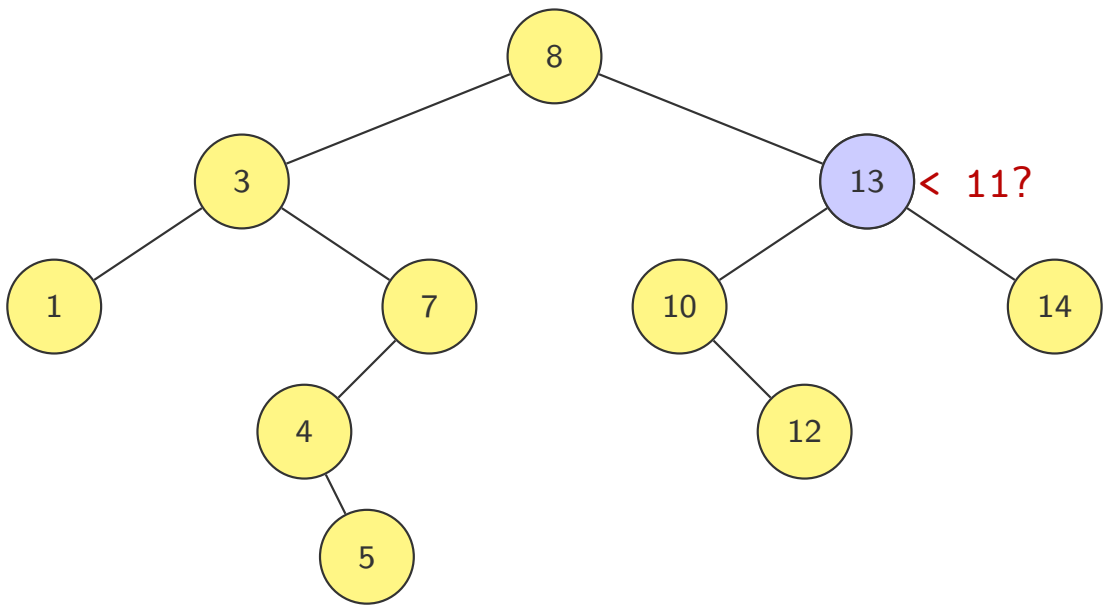


Busca por um valor

A ideia é semelhante a da busca binária:

- Ou o valor a ser buscado está na raiz da árvore
- Ou é menor do que o valor da raiz
  - Se estiver na árvore, está na subárvore esquerda
- Ou é maior do que o valor da raiz
  - Se estiver na árvore, está na subárvore direita

Ex: Buscando por 11

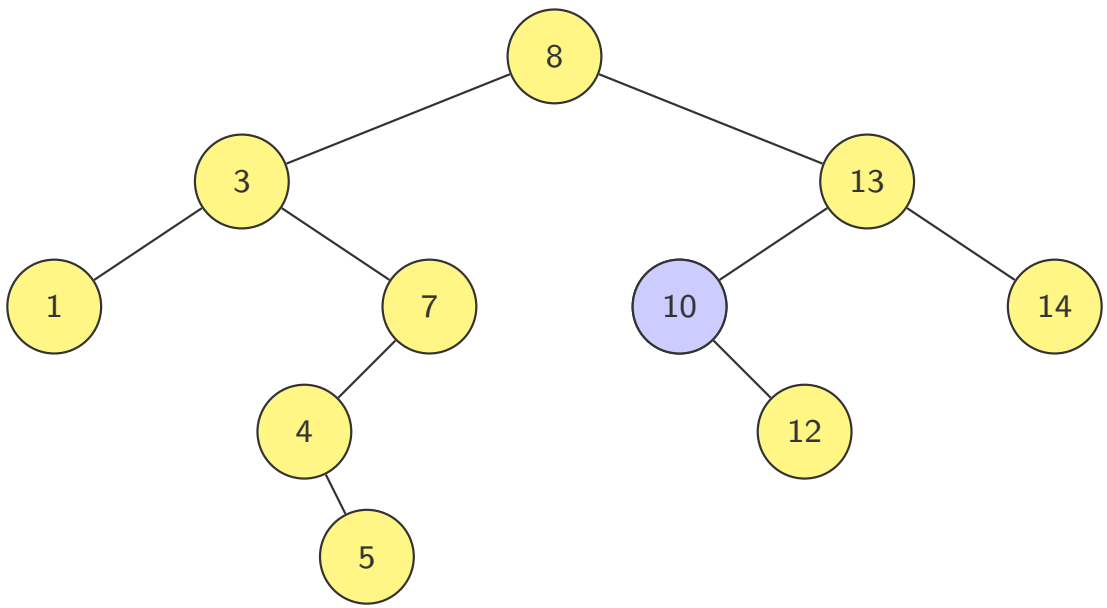


Busca por um valor

A ideia é semelhante a da busca binária:

- Ou o valor a ser buscado está na raiz da árvore
- Ou é menor do que o valor da raiz
  - Se estiver na árvore, está na subárvore esquerda
- Ou é maior do que o valor da raiz
  - Se estiver na árvore, está na subárvore direita

Ex: Buscando por 11

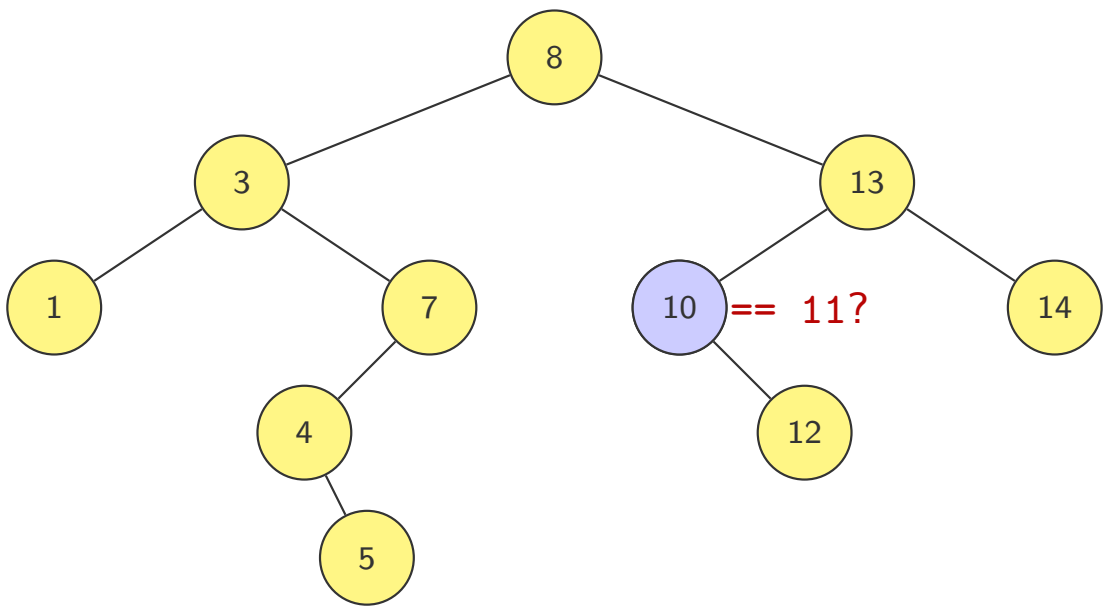


Busca por um valor

A ideia é semelhante a da busca binária:

- Ou o valor a ser buscado está na raiz da árvore
- Ou é menor do que o valor da raiz
  - Se estiver na árvore, está na subárvore esquerda
- Ou é maior do que o valor da raiz
  - Se estiver na árvore, está na subárvore direita

Ex: Buscando por 11

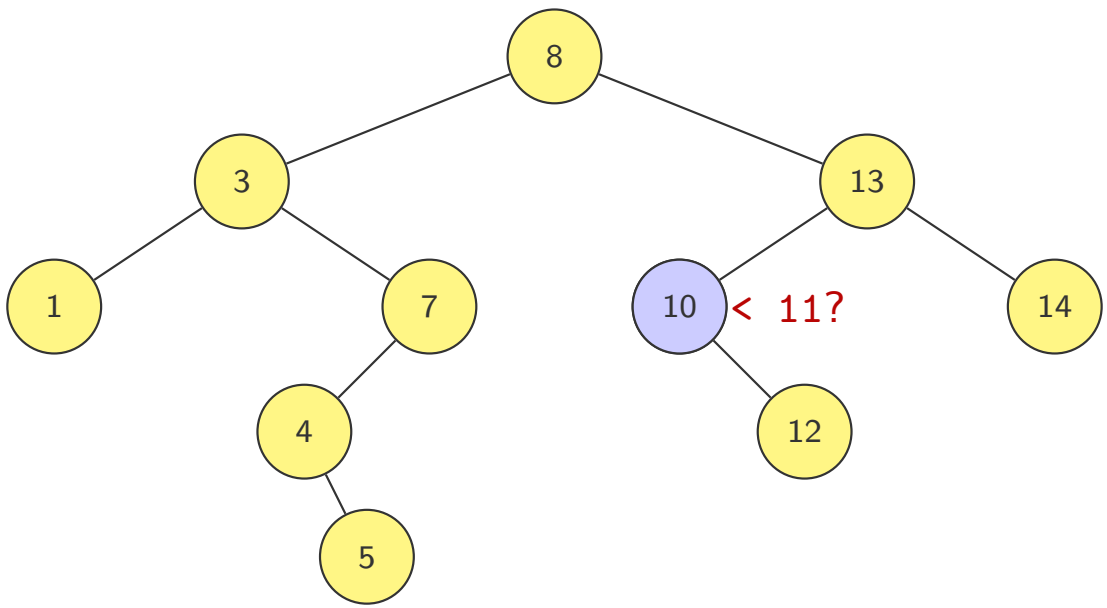


Busca por um valor

A ideia é semelhante a da busca binária:

- Ou o valor a ser buscado está na raiz da árvore
- Ou é menor do que o valor da raiz
  - Se estiver na árvore, está na subárvore esquerda
- Ou é maior do que o valor da raiz
  - Se estiver na árvore, está na subárvore direita

Ex: Buscando por 11

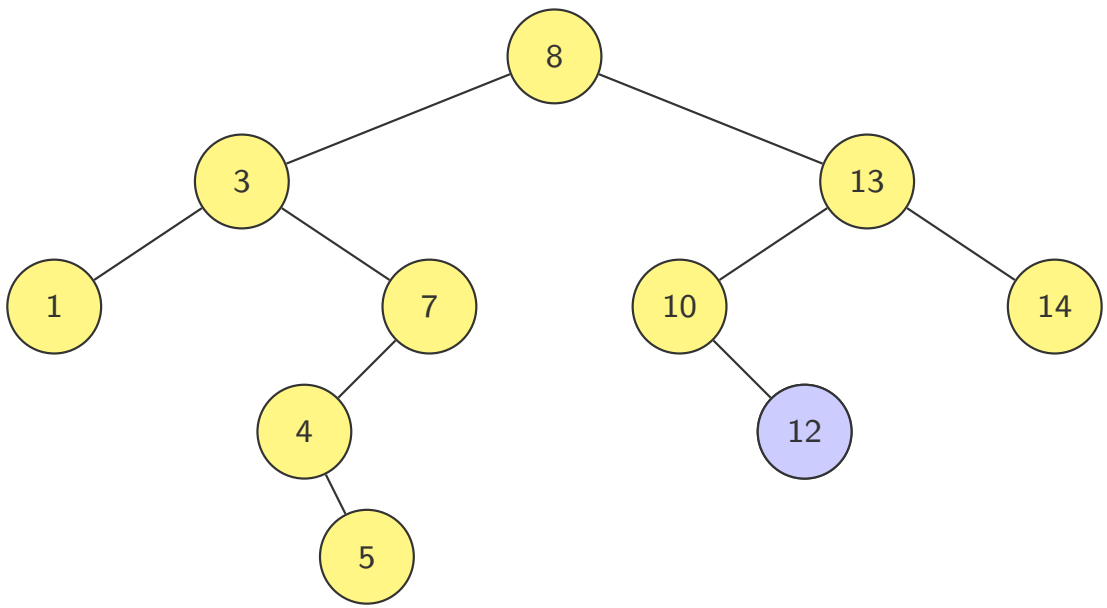


Busca por um valor

A ideia é semelhante a da busca binária:

- Ou o valor a ser buscado está na raiz da árvore
- Ou é menor do que o valor da raiz
  - Se estiver na árvore, está na subárvore esquerda
- Ou é maior do que o valor da raiz
  - Se estiver na árvore, está na subárvore direita

Ex: Buscando por 11

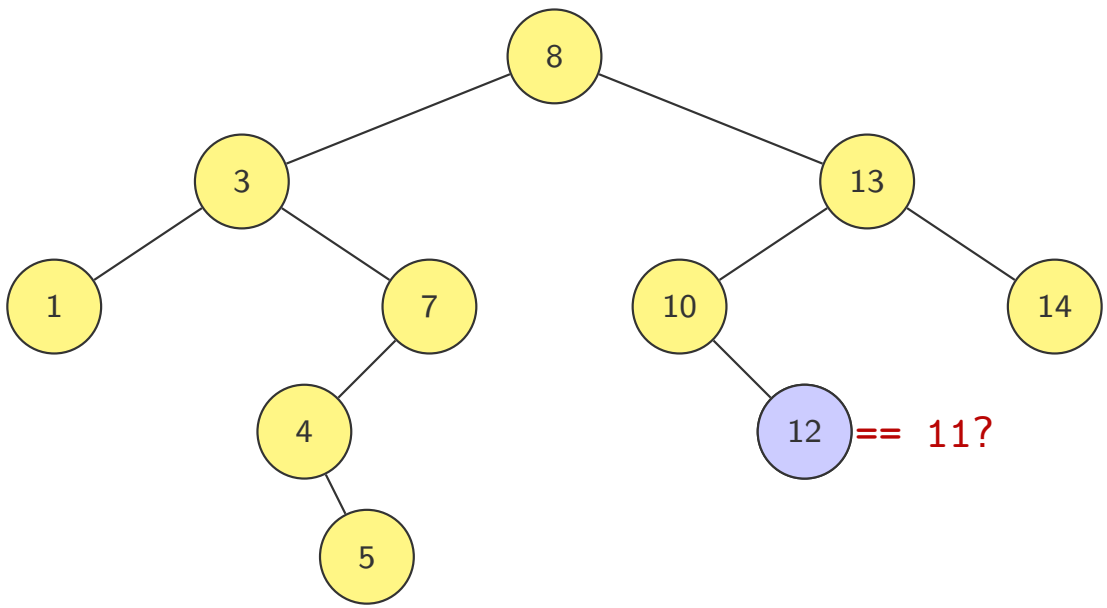


Busca por um valor

A ideia é semelhante a da busca binária:

- Ou o valor a ser buscado está na raiz da árvore
- Ou é menor do que o valor da raiz
  - Se estiver na árvore, está na subárvore esquerda
- Ou é maior do que o valor da raiz
  - Se estiver na árvore, está na subárvore direita

Ex: Buscando por 11

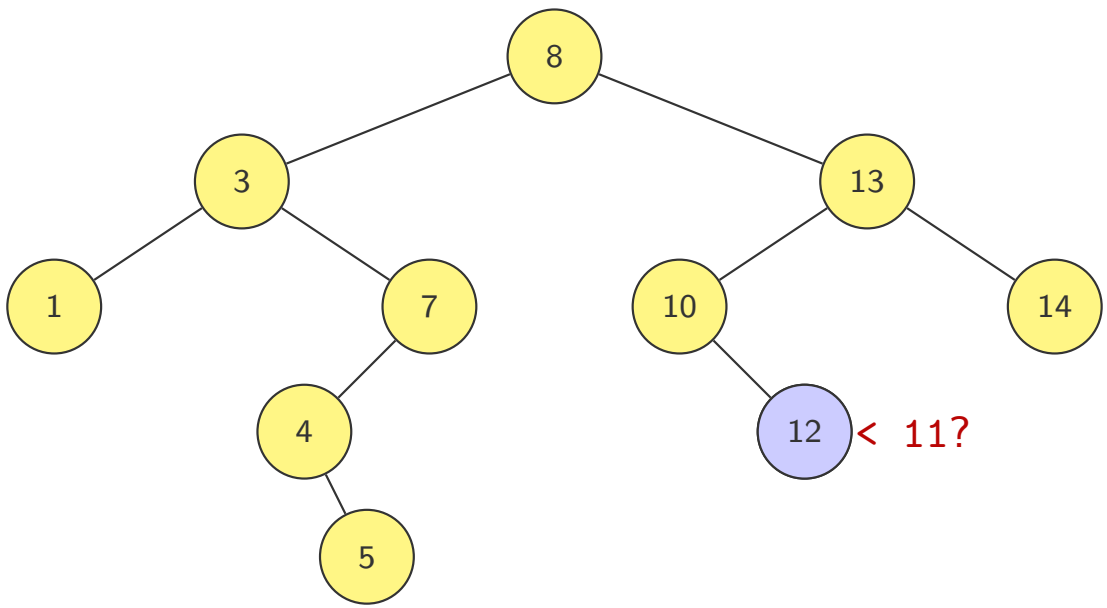


Busca por um valor

A ideia é semelhante a da busca binária:

- Ou o valor a ser buscado está na raiz da árvore
- Ou é menor do que o valor da raiz
  - Se estiver na árvore, está na subárvore esquerda
- Ou é maior do que o valor da raiz
  - Se estiver na árvore, está na subárvore direita

Ex: Buscando por 11

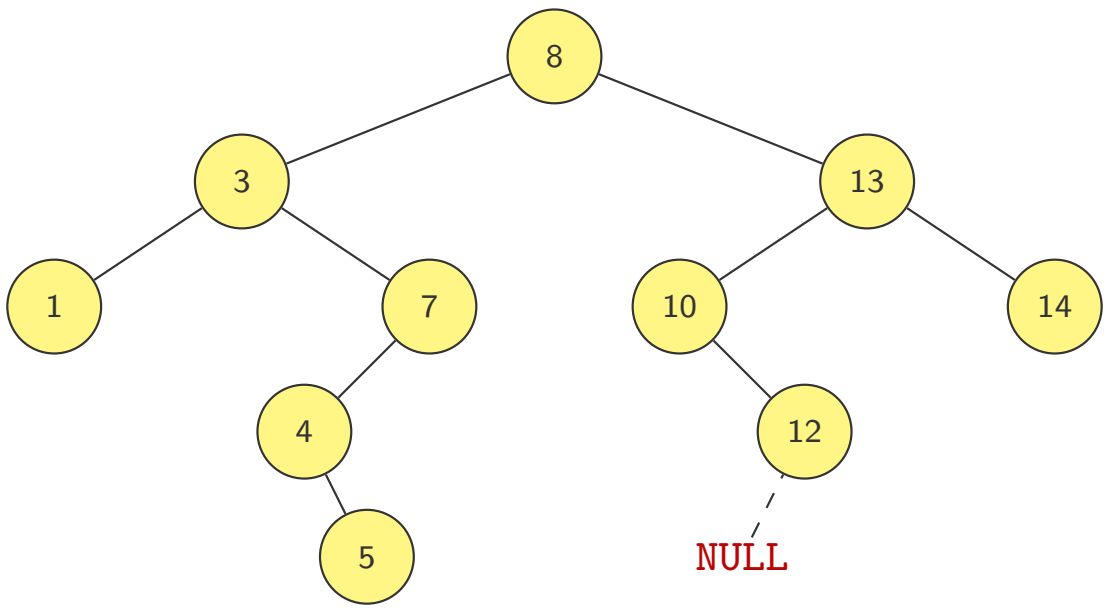


Busca por um valor

A ideia é semelhante a da busca binária:

- Ou o valor a ser buscado está na raiz da árvore
- Ou é menor do que o valor da raiz
  - Se estiver na árvore, está na subárvore esquerda
- Ou é maior do que o valor da raiz
  - Se estiver na árvore, está na subárvore direita

Ex: Buscando por 11



Implementação - Busca

Implementação - Busca


```
1  NoArv* abb_busca(NoArv *node, int valor) {
2      if(node == NULL)
3          return NULL;
4      else if(valor < node->chave)
5          return abb_busca(node->esq, valor);
6      else if(valor > node->chave)
7          return abb_busca(node->dir, valor);
8      else
9          return node;
10 }
```

Eficiência da busca

Qual é o tempo da busca?

Eficiência da busca

Qual é o tempo da busca?

- depende da forma da árvore...

Eficiência da busca

Qual é o tempo da busca?

- depende da forma da árvore...

Ex: 31 nós

Eficiência da busca

Qual é o tempo da busca?

- depende da forma da árvore...

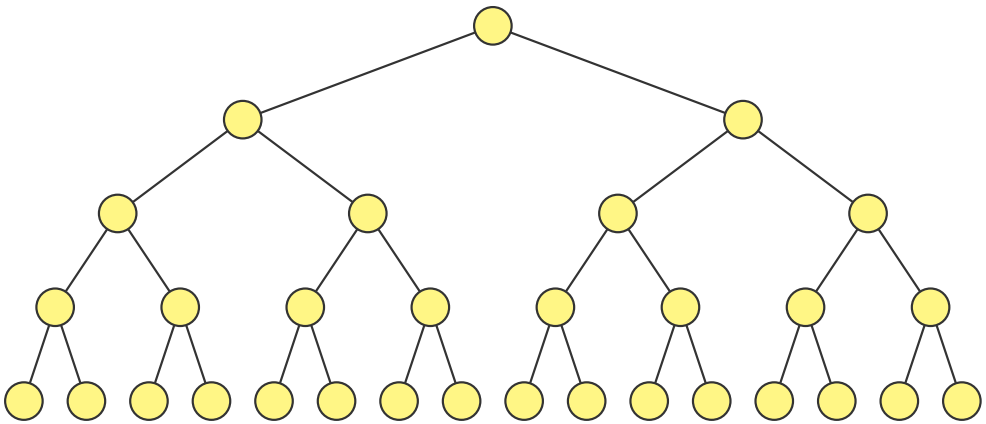
Ex: 31 nós

Eficiência da busca

Qual é o tempo da busca?

- depende da forma da árvore...

Ex: 31 nós



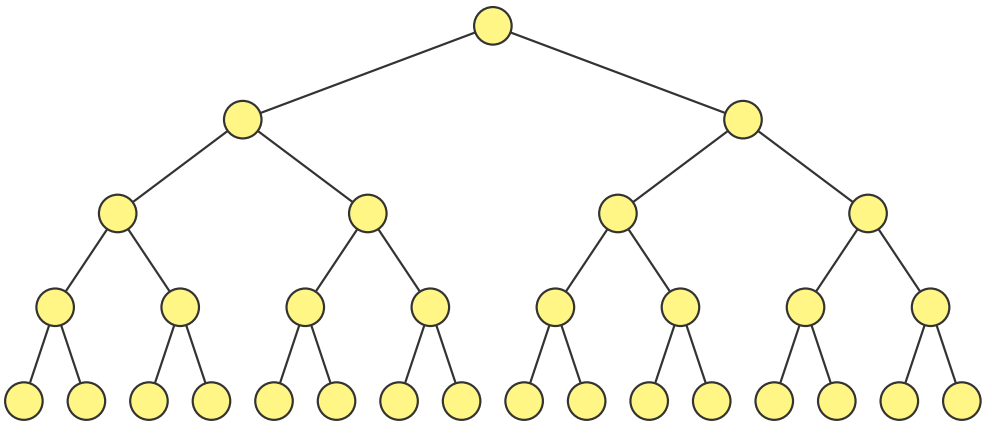
Melhor árvore: $O(\lg n)$

Eficiência da busca

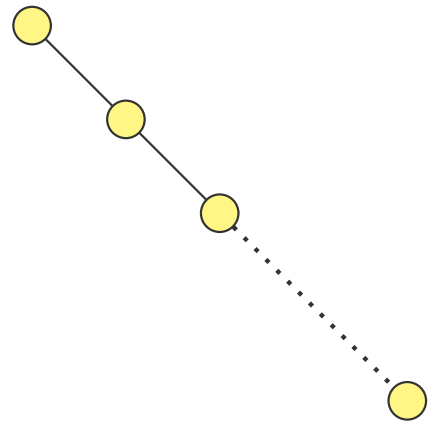
Qual é o tempo da busca?

- depende da forma da árvore...

Ex: 31 nós



Melhor árvore: $O(\lg n)$



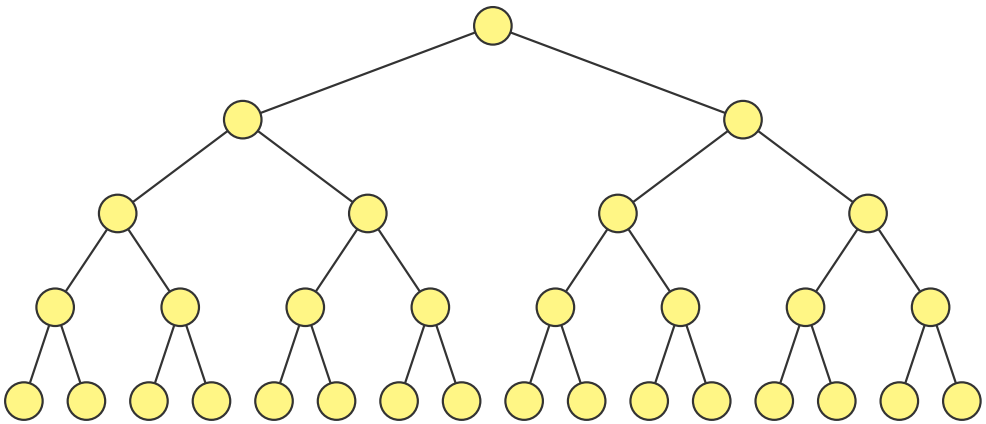
Pior árvore: $O(n)$

Eficiência da busca

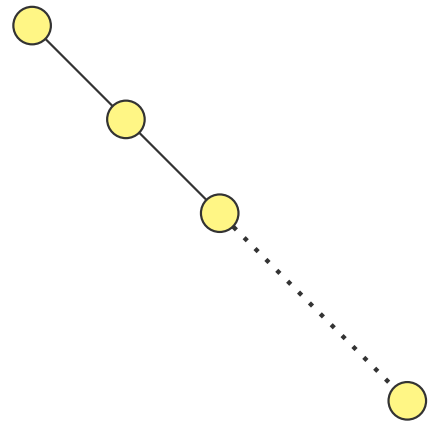
Qual é o tempo da busca?

- depende da forma da árvore...

Ex: 31 nós



Melhor árvore: $O(\lg n)$



Pior árvore: $O(n)$

Para ter a pior árvore basta inserir em ordem crescente...

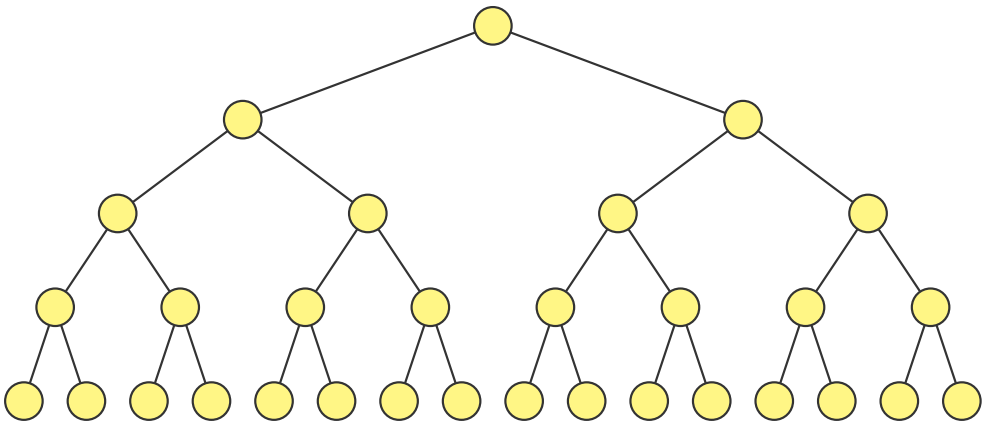
Eficiência da busca

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

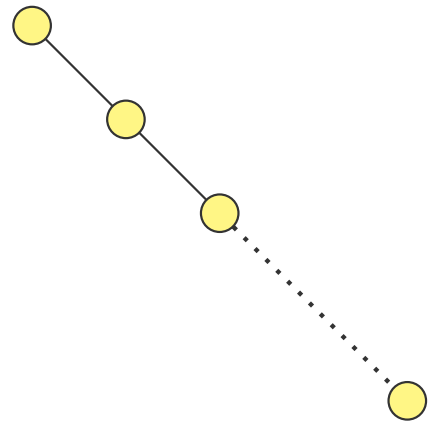
Qual é o tempo da busca?

- depende da forma da árvore...

Ex: 31 nós



Melhor árvore: $O(\lg n)$



Pior árvore: $O(n)$

Para ter a pior árvore basta inserir em ordem crescente...

Caso médio: em uma árvore com n elementos adicionados em ordem aleatória a busca demora (em média) $O(\lg n)$

Inserindo um valor

Precisamos determinar onde inserir o valor:

Inserindo um valor

Precisamos determinar onde inserir o valor:

- fazemos uma busca pelo valor

Inserindo um valor

Precisamos determinar onde inserir o valor:

- fazemos uma busca pelo valor
- e colocamos ele na posição onde deveria estar

Inserindo um valor

Precisamos determinar onde inserir o valor:

- fazemos uma busca pelo valor
- e colocamos ele na posição onde deveria estar

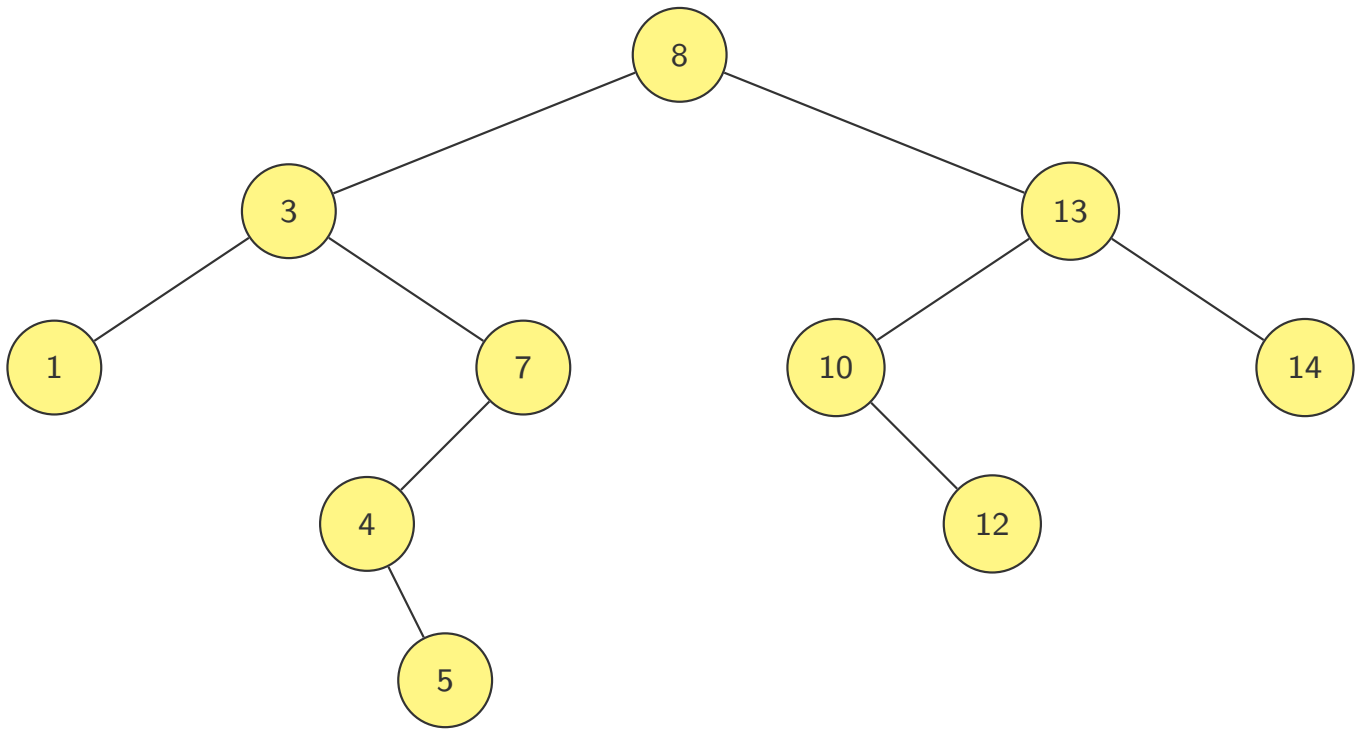
Ex: Inserindo 11

Inserindo um valor

Precisamos determinar onde inserir o valor:

- fazemos uma busca pelo valor
- e colocamos ele na posição onde deveria estar

Ex: Inserindo 11

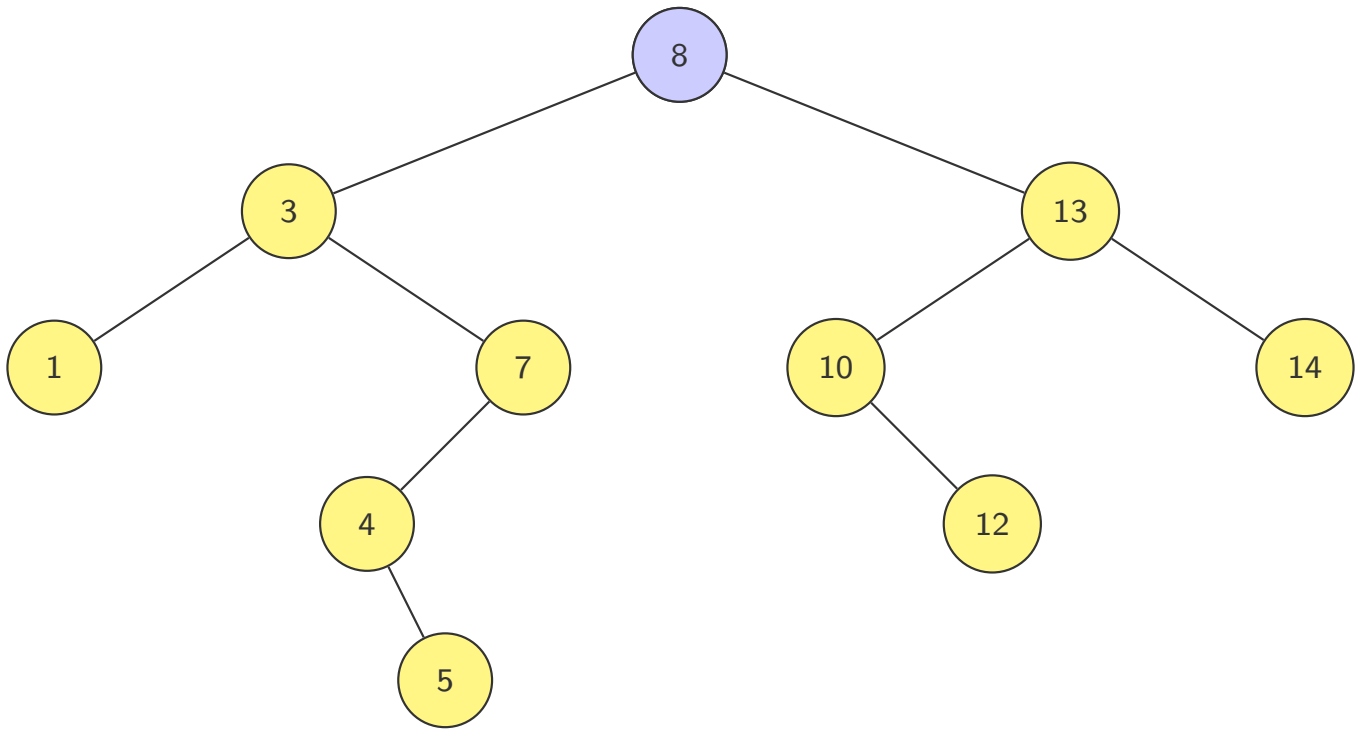


Inserindo um valor

Precisamos determinar onde inserir o valor:

- fazemos uma busca pelo valor
- e colocamos ele na posição onde deveria estar

Ex: Inserindo 11

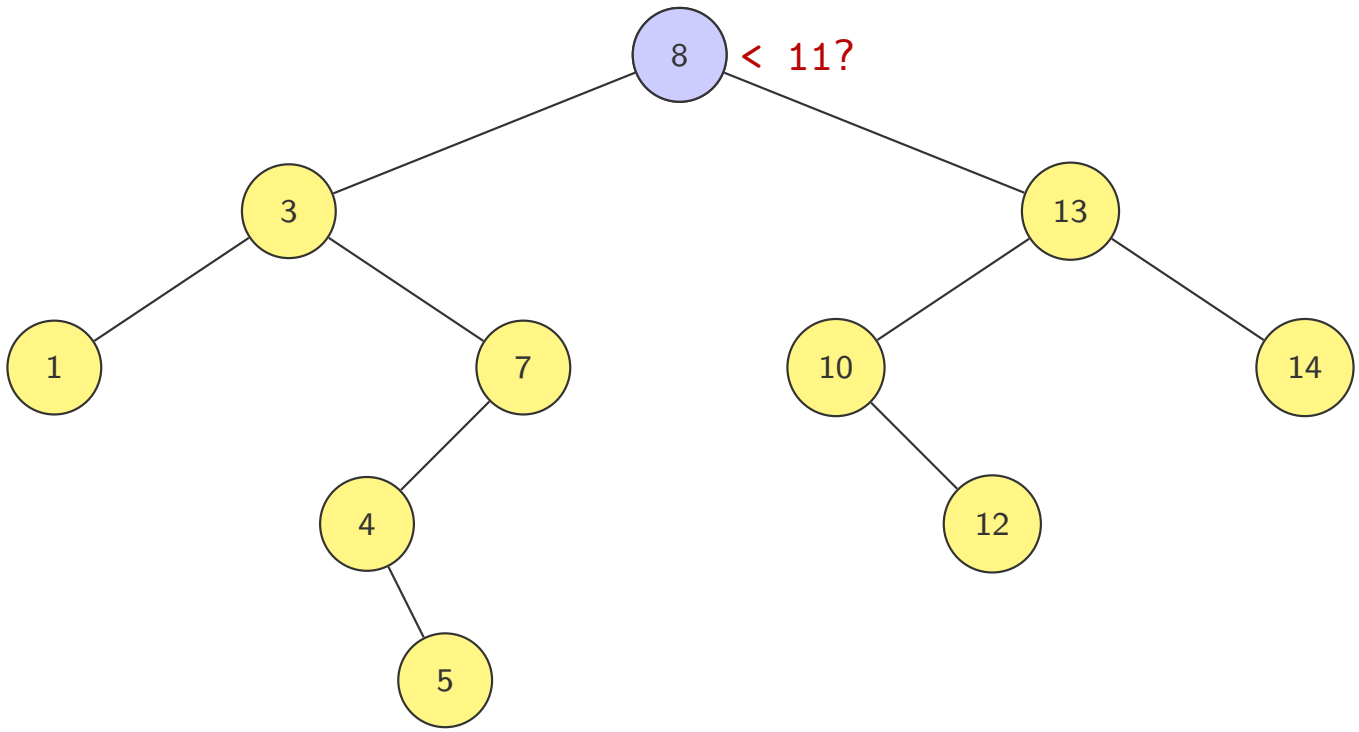


Inserindo um valor

Precisamos determinar onde inserir o valor:

- fazemos uma busca pelo valor
- e colocamos ele na posição onde deveria estar

Ex: Inserindo 11

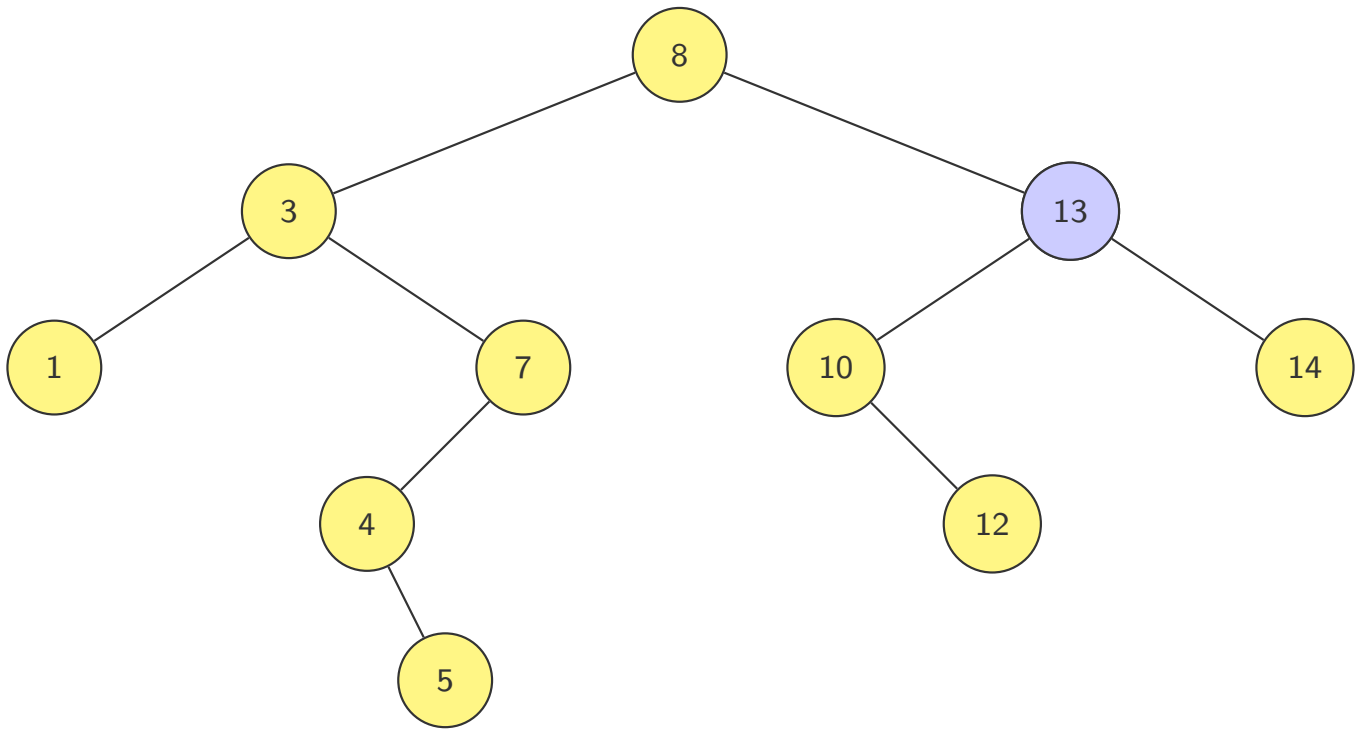


Inserindo um valor

Precisamos determinar onde inserir o valor:

- fazemos uma busca pelo valor
- e colocamos ele na posição onde deveria estar

Ex: Inserindo 11

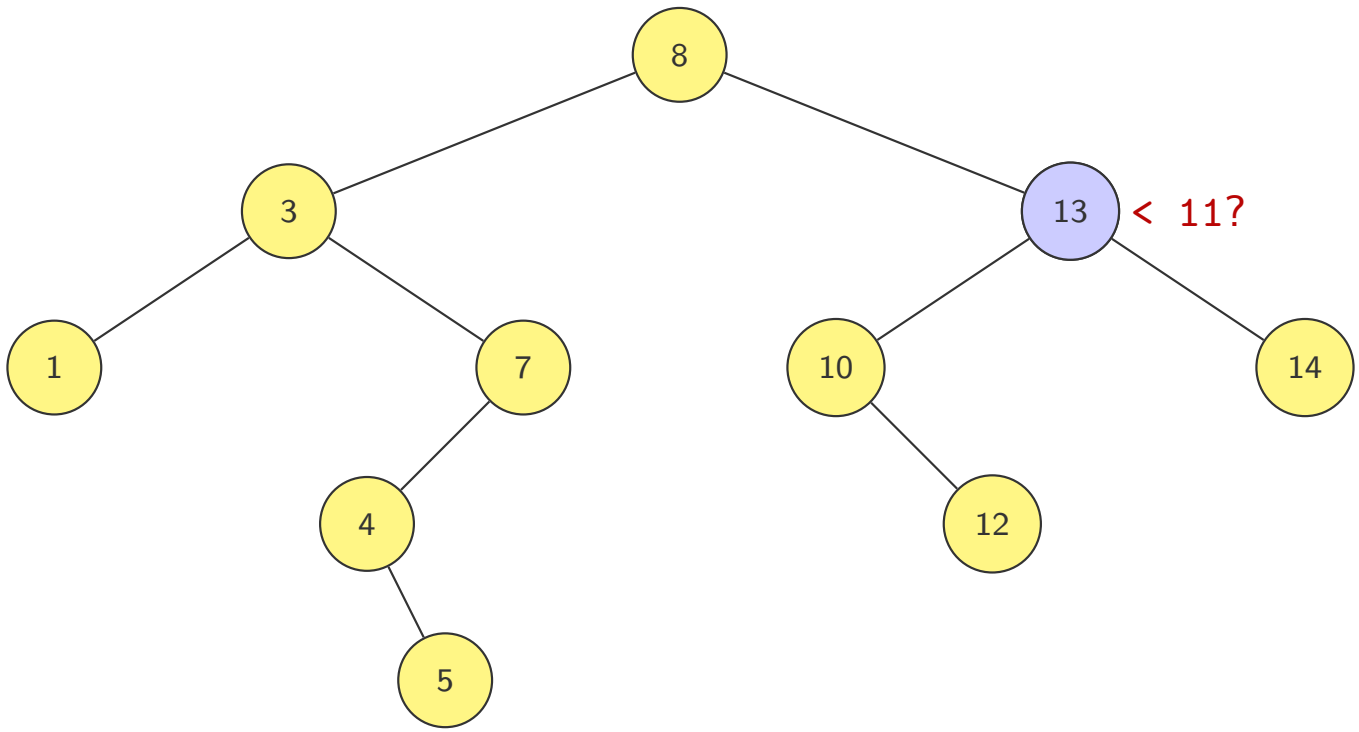


Inserindo um valor

Precisamos determinar onde inserir o valor:

- fazemos uma busca pelo valor
- e colocamos ele na posição onde deveria estar

Ex: Inserindo 11

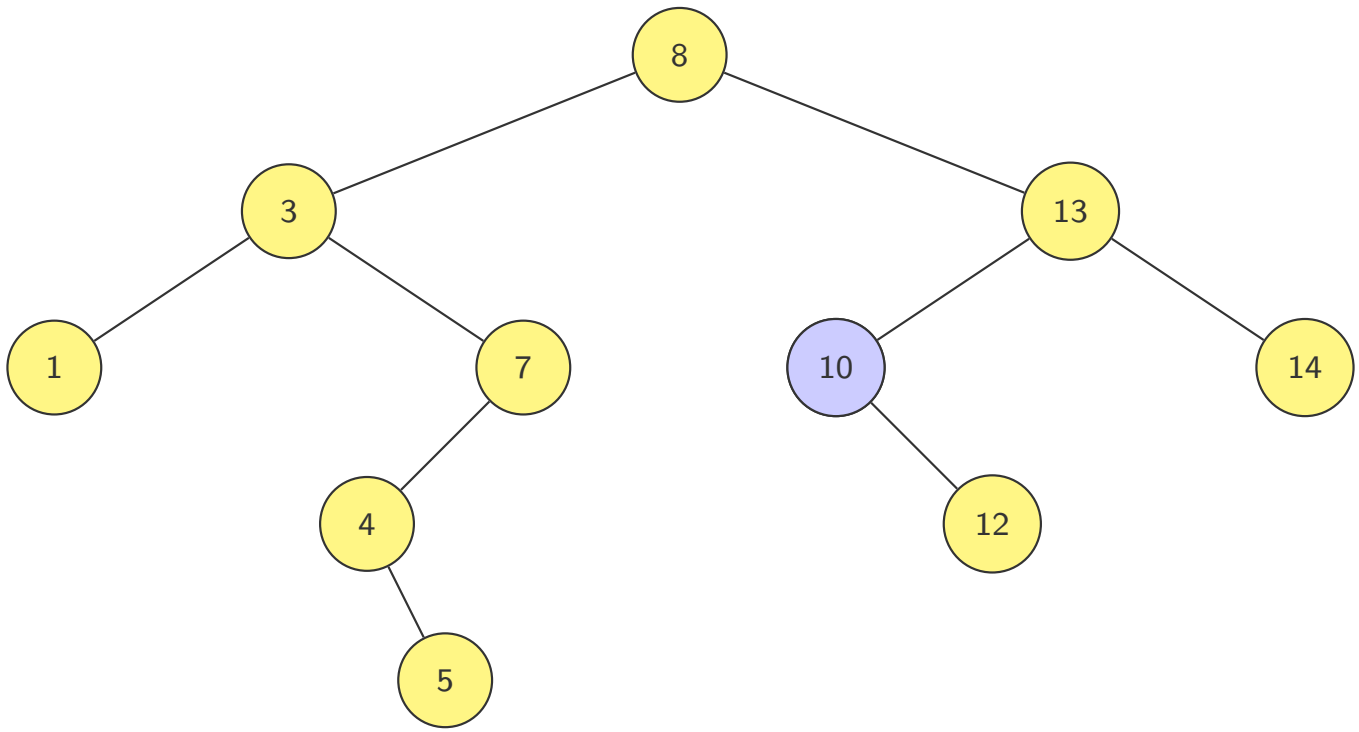


Inserindo um valor

Precisamos determinar onde inserir o valor:

- fazemos uma busca pelo valor
- e colocamos ele na posição onde deveria estar

Ex: Inserindo 11

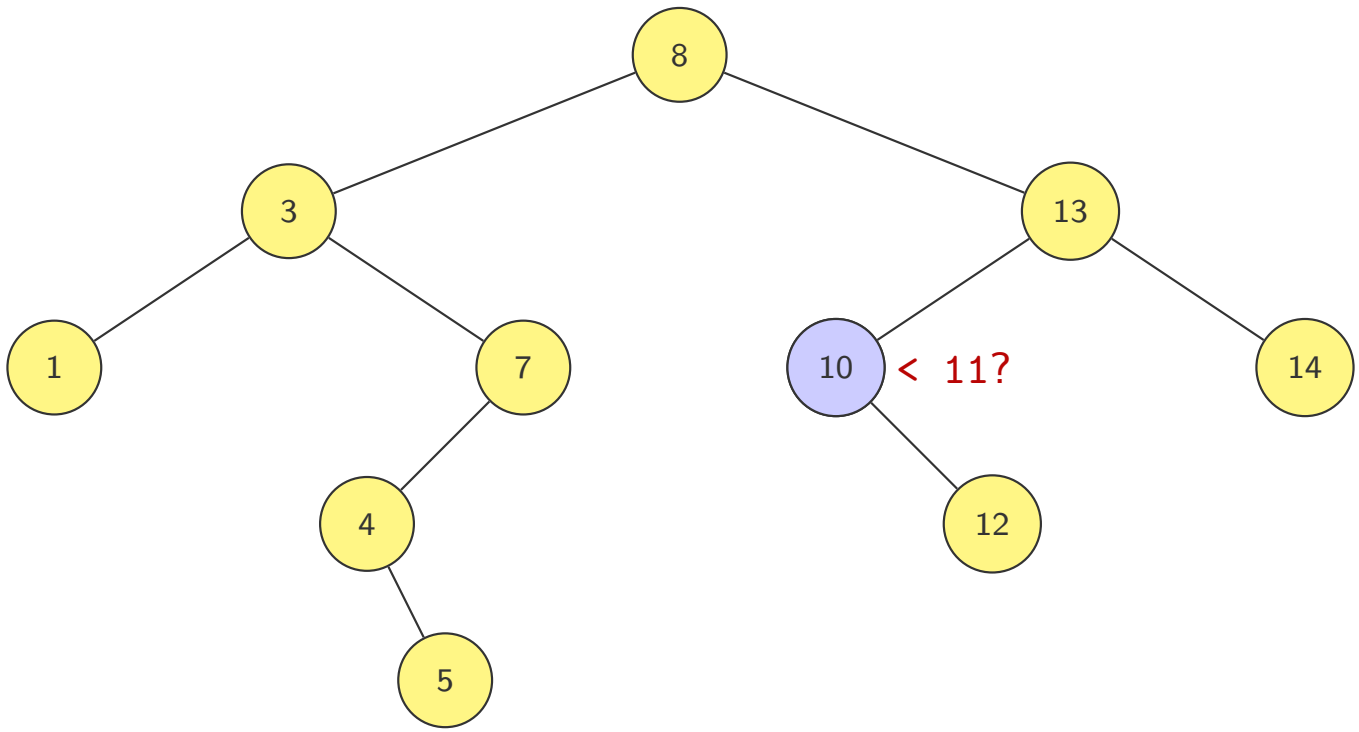


Inserindo um valor

Precisamos determinar onde inserir o valor:

- fazemos uma busca pelo valor
- e colocamos ele na posição onde deveria estar

Ex: Inserindo 11

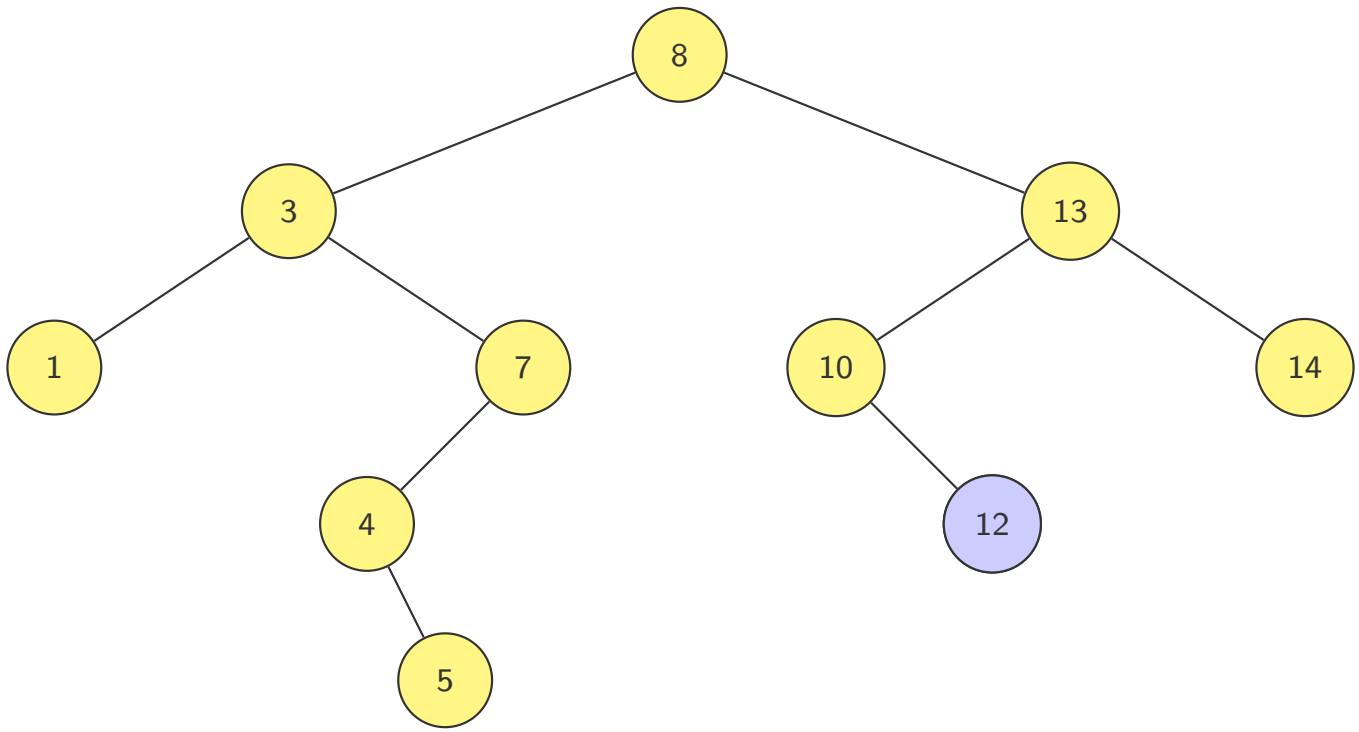


Inserindo um valor

Precisamos determinar onde inserir o valor:

- fazemos uma busca pelo valor
- e colocamos ele na posição onde deveria estar

Ex: Inserindo 11

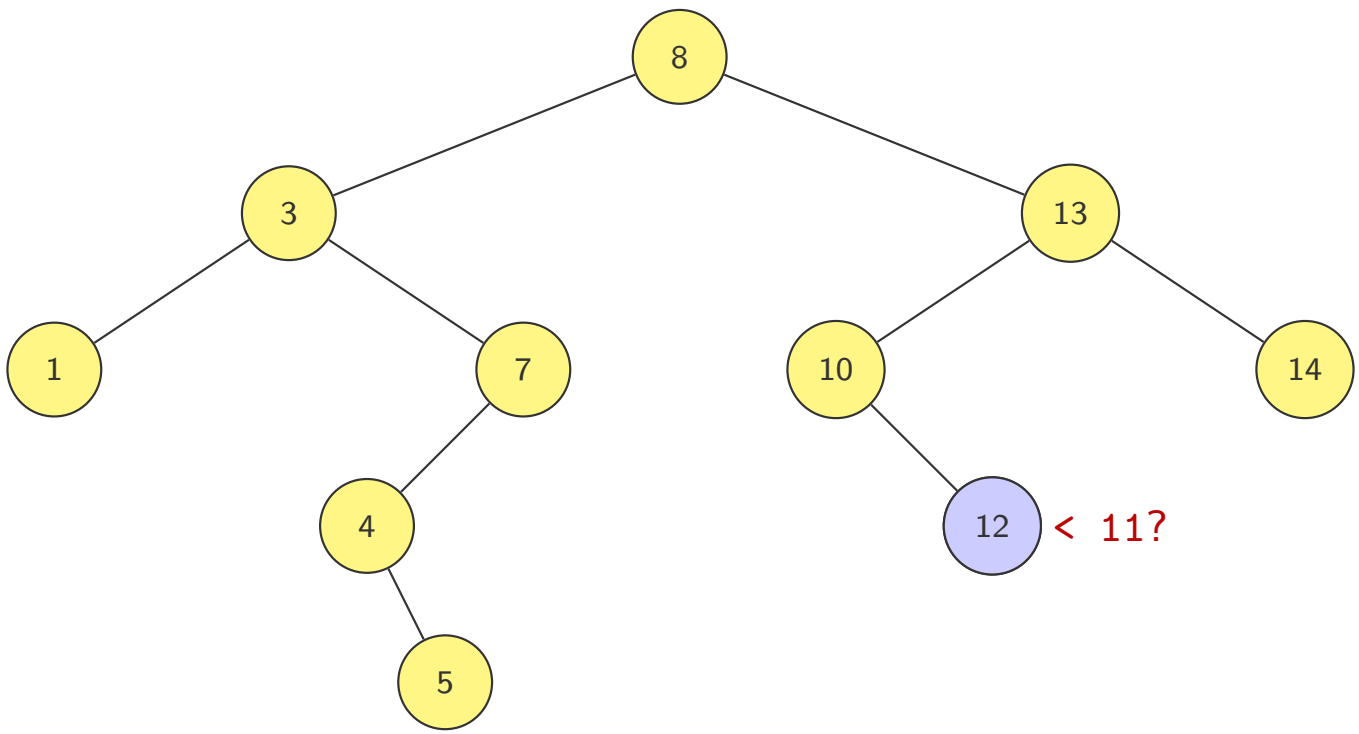


Inserindo um valor

Precisamos determinar onde inserir o valor:

- fazemos uma busca pelo valor
- e colocamos ele na posição onde deveria estar

Ex: Inserindo 11

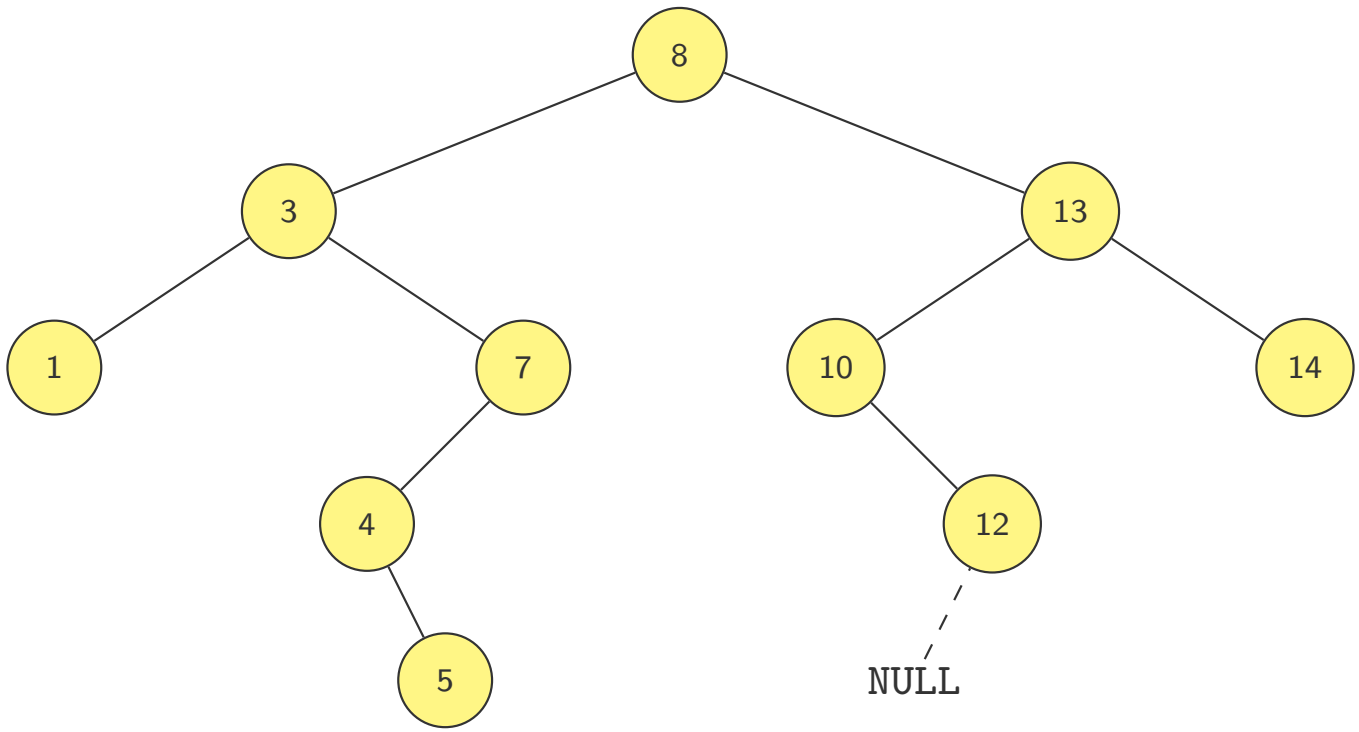


Inserindo um valor

Precisamos determinar onde inserir o valor:

- fazemos uma busca pelo valor
- e colocamos ele na posição onde deveria estar

Ex: Inserindo 11

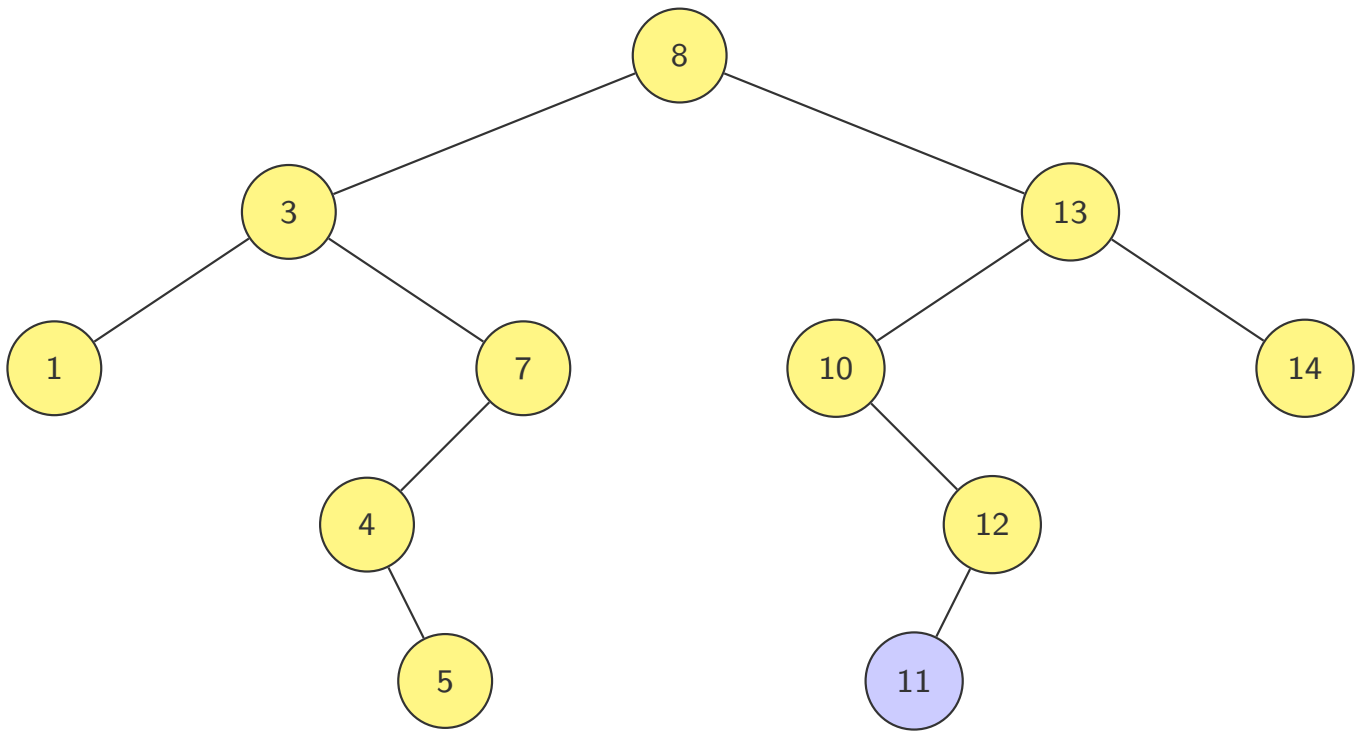


Inserindo um valor

Precisamos determinar onde inserir o valor:

- fazemos uma busca pelo valor
- e colocamos ele na posição onde deveria estar

Ex: Inserindo 11



Inserção — Implementação

Inserção — Implementação

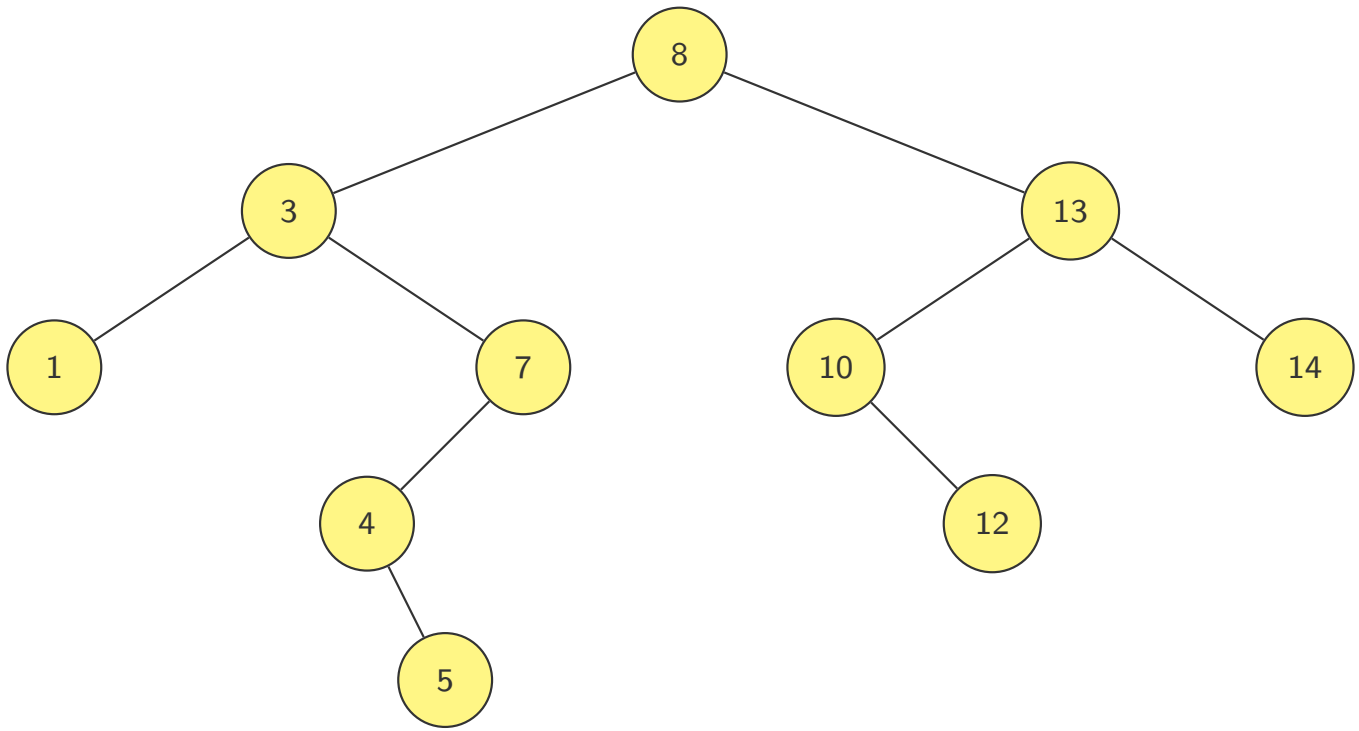

```
1  // Esta funcao insere um novo valor na arvore somente
2  // se a chave nao for repetida. Ela devolve um ponteiro
3  // para a raiz da nova arvore.
4  NoArv* abb_insere(NoArv *node, int valor) {
5      if(node == NULL) {
6          node = (NoArv*) malloc(sizeof(NoArv));
7          node->chave = valor;
8          node->esq = node->dir = NULL;
9      } else if(valor < node->chave) {
10         node->esq = abb_insere(node->esq, valor);
11     } else if(valor > node->chave) {
12         node->dir = abb_insere(node->dir, valor);
13     }
14     return node;
15 }
```

Mínimo da Árvore Binária de Busca

Onde está o nó com a menor chave de uma árvore?

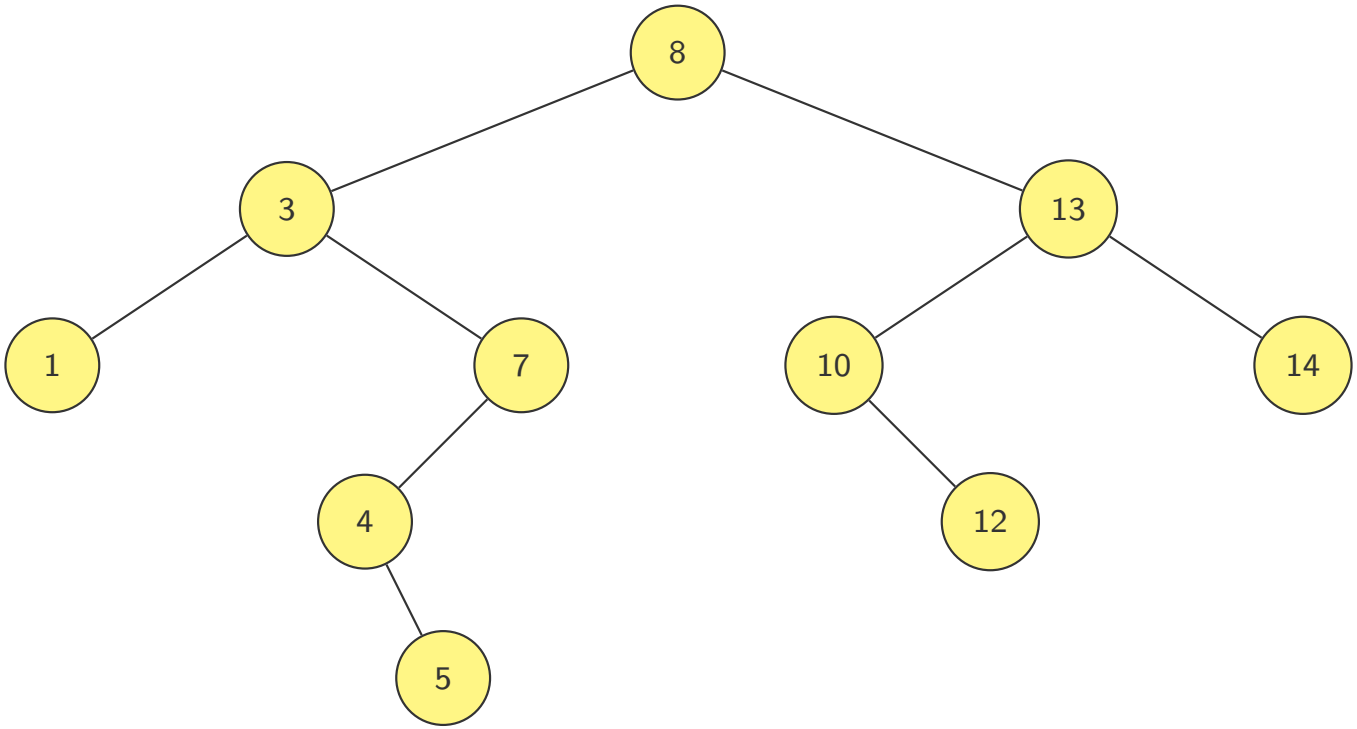
Mínimo da Árvore Binária de Busca

Onde está o nó com a menor chave de uma árvore?



Mínimo da Árvore Binária de Busca

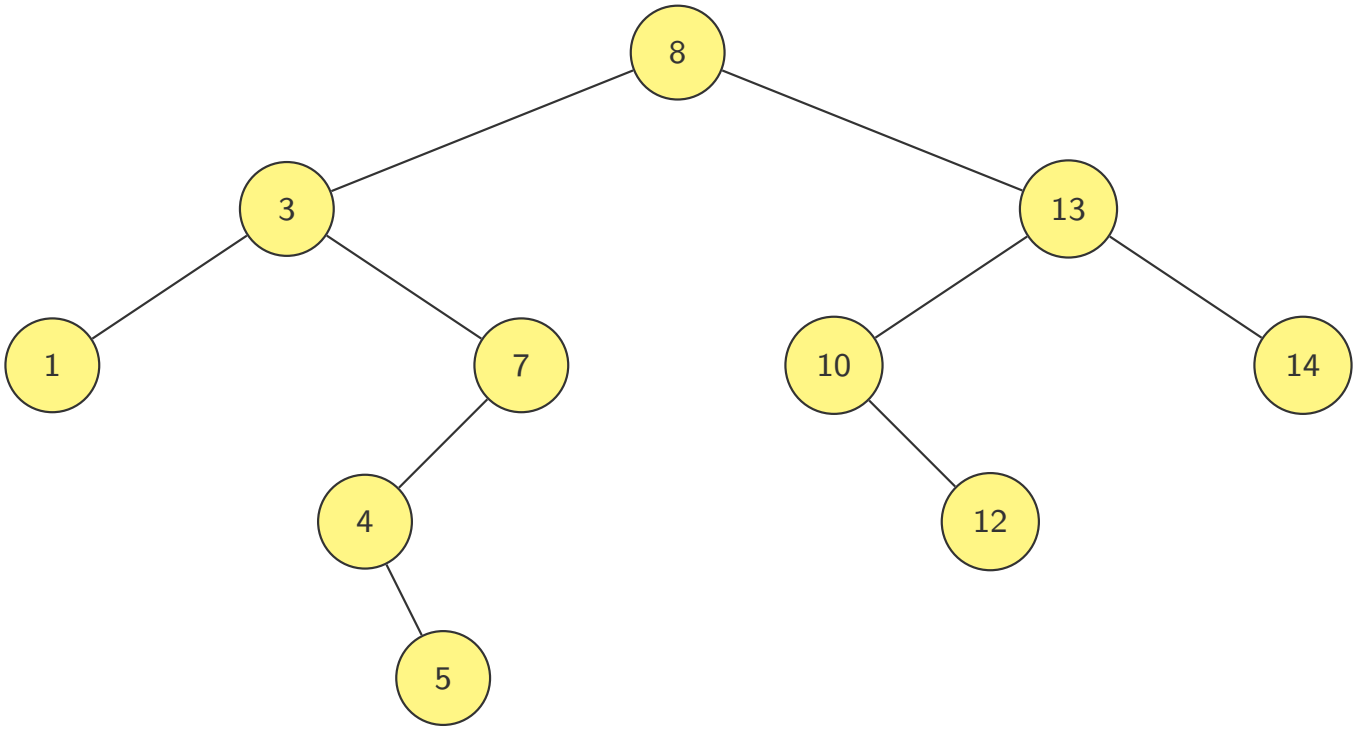
Onde está o nó com a menor chave de uma árvore?



Quem é o mínimo para essa árvore?

Mínimo da Árvore Binária de Busca

Onde está o nó com a menor chave de uma árvore?



Quem é o mínimo para essa árvore?

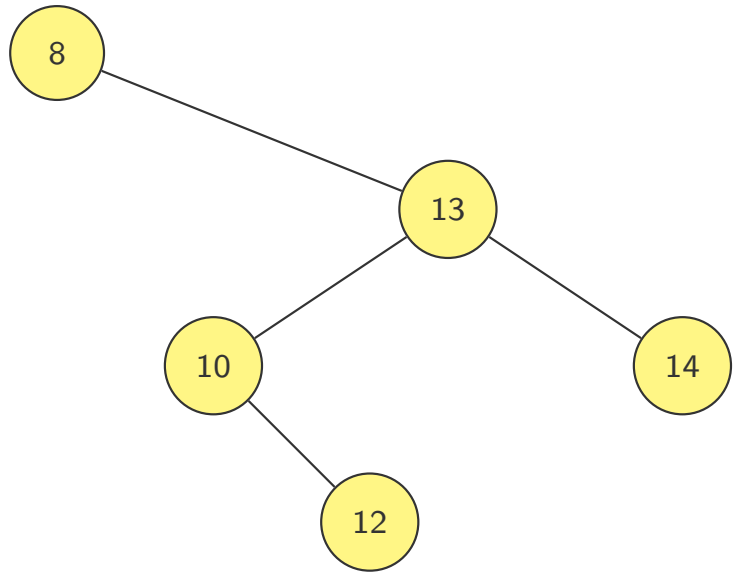
- É o mínimo da subárvore esquerda

Mínimo da Árvore Binária de Busca

Onde está o nó com a menor chave de uma árvore?

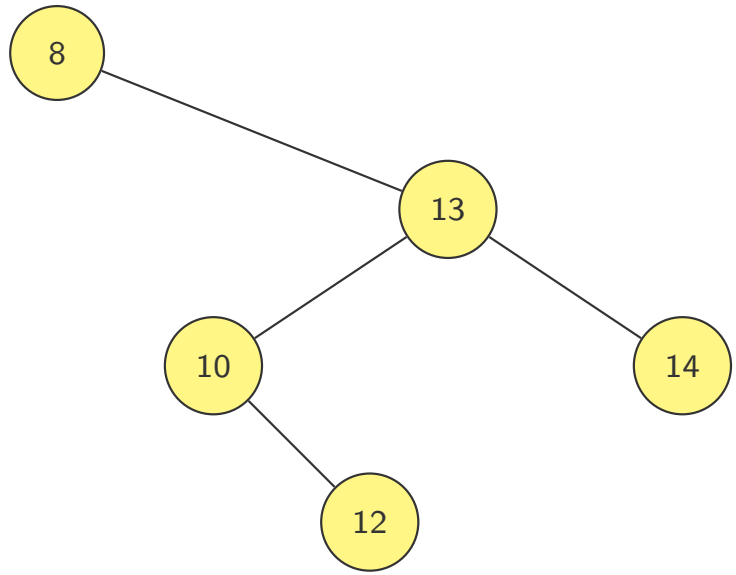
Mínimo da Árvore Binária de Busca

Onde está o nó com a menor chave de uma árvore?



Mínimo da Árvore Binária de Busca

Onde está o nó com a menor chave de uma árvore?

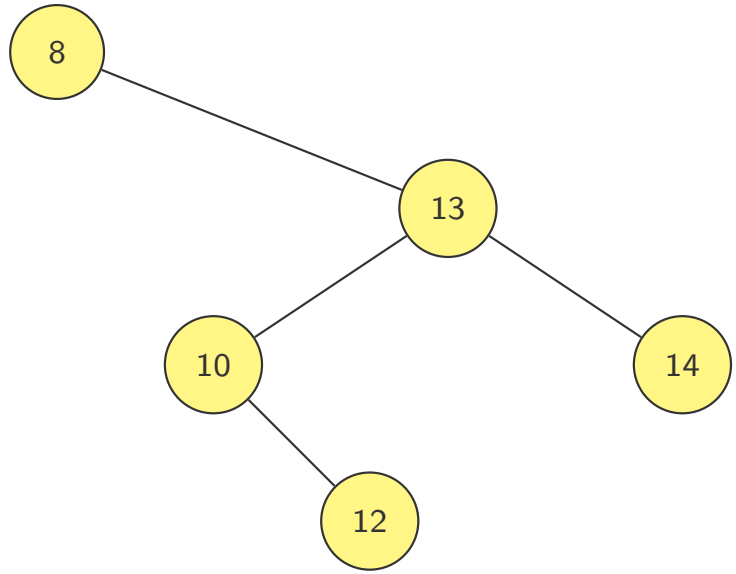


Quem é o mínimo para essa árvore?

Mínimo da Árvore Binária de Busca

Universidade Federal do Ceará
Campus Quixadá

Onde está o nó com a menor chave de uma árvore?



Quem é o mínimo para essa árvore?

- É a própria raiz

Mínimo - Implementação

Mínimo - Implementação


```
1 // Retorna um ponteiro para o no com valor minimo,
2 // ou nulo se nao existir (arvore vazia)
3 NoArv* abb_minimo(NoArv *node) {
4     if(node == NULL) return NULL;
5     if(node->esq == NULL) return node;
6     else return abb_minimo(node->esq);
7 }
```

Mínimo - Implementação


```
1 // Retorna um ponteiro para o no com valor minimo,
2 // ou nulo se nao existir (arvore vazia)
3 NoArv* abb_minimo(NoArv *node) {
4     if(node == NULL) return NULL;
5     if(node->esq == NULL) return node;
6     else return abb_minimo(node->esq);
7 }
```

Para encontrar o máximo, basta fazer a operação simétrica (Tarefa)

Mínimo - Implementação


```
1 // Retorna um ponteiro para o no com valor minimo,
2 // ou nulo se nao existir (arvore vazia)
3 NoArv* abb_minimo(NoArv *node) {
4     if(node == NULL) return NULL;
5     if(node->esq == NULL) return node;
6     else return abb_minimo(node->esq);
7 }
```

Para encontrar o máximo, basta fazer a operação simétrica (Tarefa)

- Se a subárvore direita existir, é o seu máximo

Mínimo - Implementação


```
1 // Retorna um ponteiro para o no com valor minimo,
2 // ou nulo se nao existir (arvore vazia)
3 NoArv* abb_minimo(NoArv *node) {
4     if(node == NULL) return NULL;
5     if(node->esq == NULL) return node;
6     else return abb_minimo(node->esq);
7 }
```

Para encontrar o máximo, basta fazer a operação simétrica (Tarefa)

- Se a subárvore direita existir, é o seu máximo
- Senão, é a própria raiz

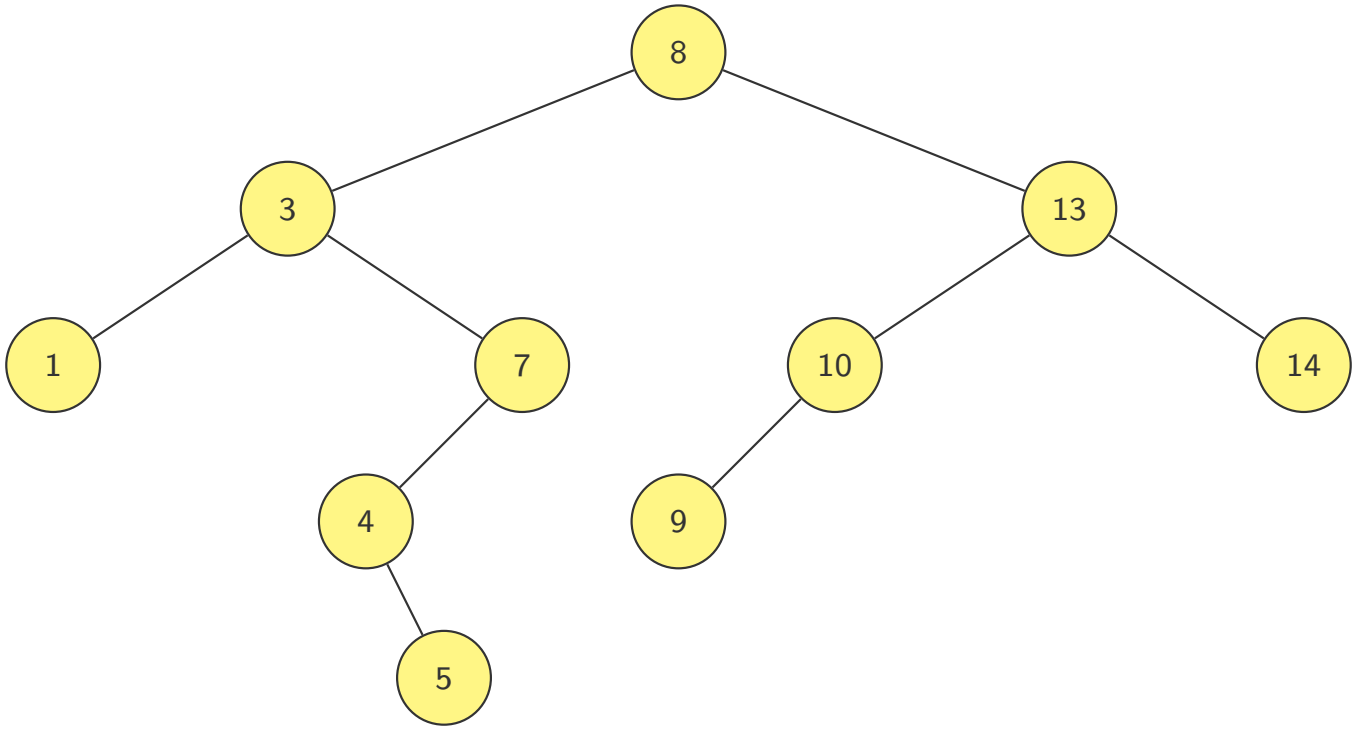
Remoção

Remoção

- Considere o problema de remover um nó de uma árvore binária de busca de tal forma que a árvore resultante continue de busca.
- Primeiro, precisamos fazer uma busca pelo nó a ser removido.
- Uma vez encontrado o nó, quais dificuldades podem surgir que dificultam a simples remoção do nó?

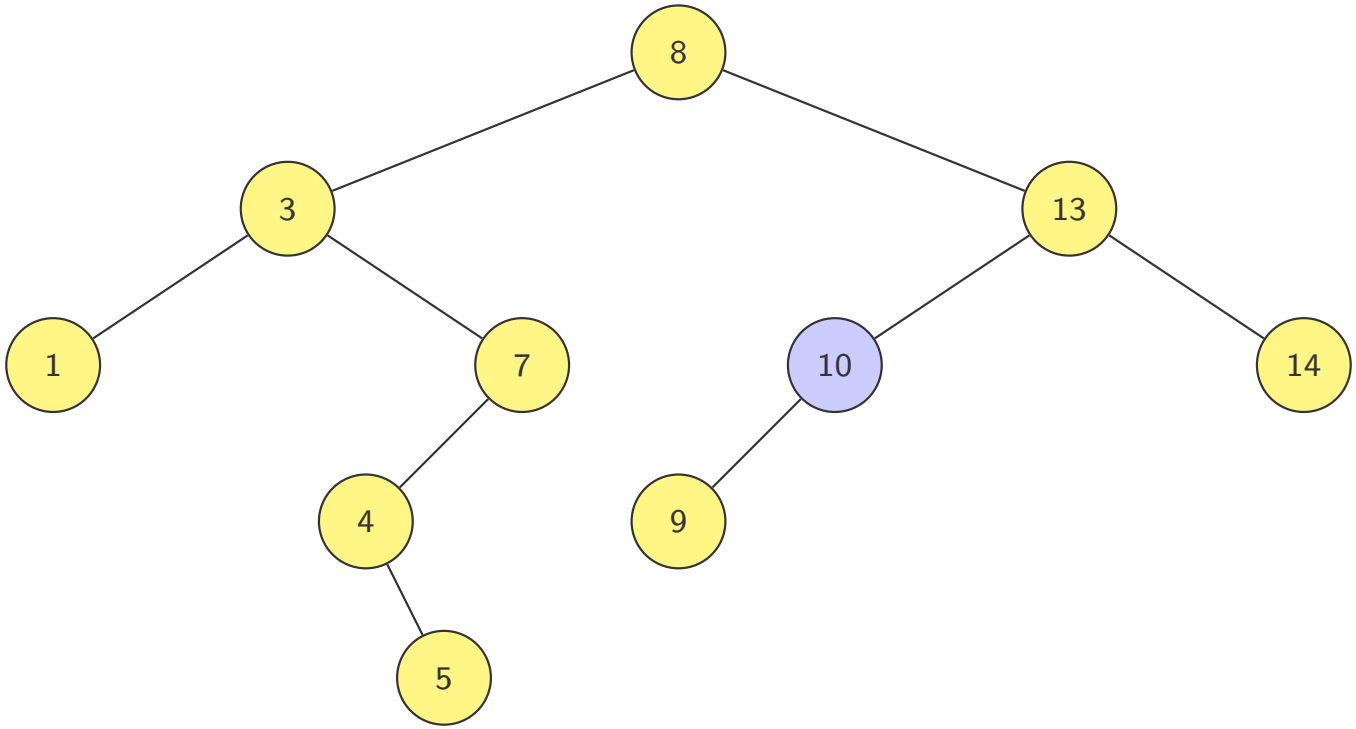
Remoção

Exemplo: queremos remover a chave 10



Remoção

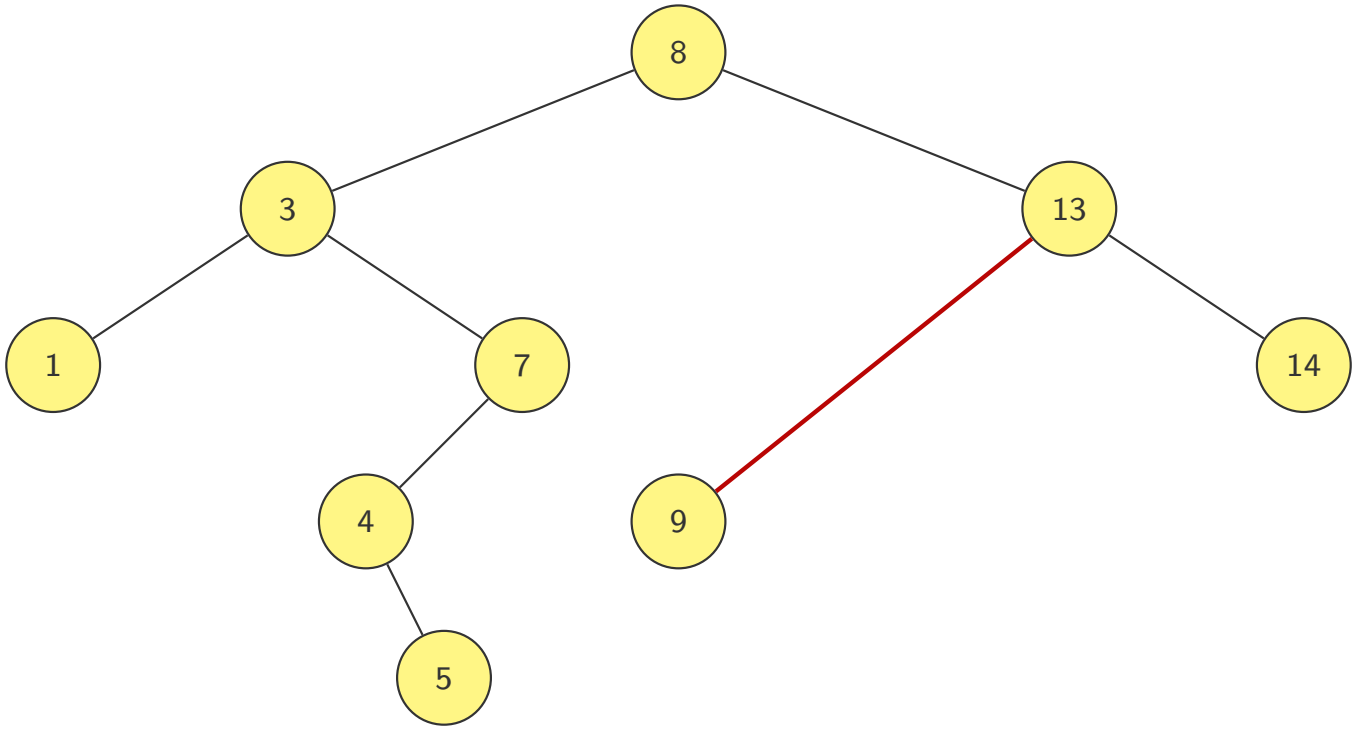
Exemplo: queremos remover a chave 10



- O nó x a ser removido pode ter exatamente um filho.

Remoção

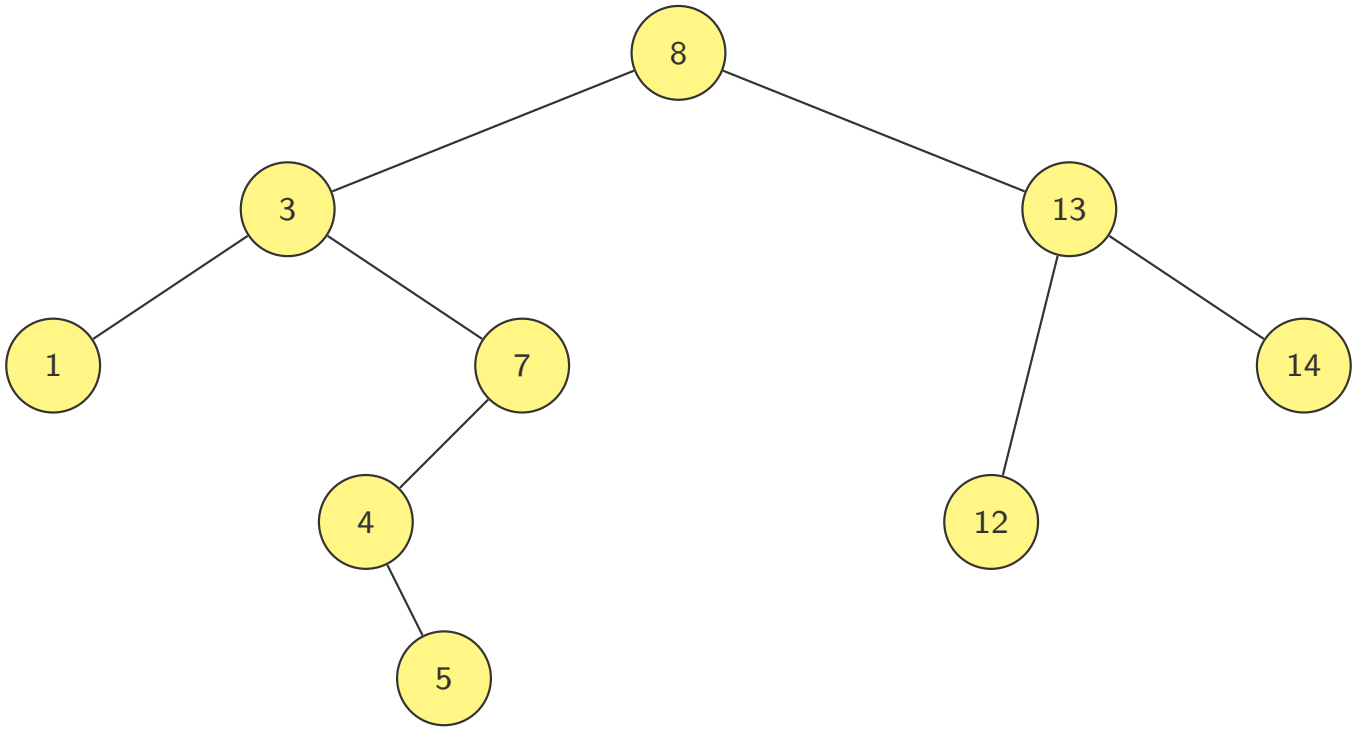
Exemplo: queremos remover a chave 10



- O nó x a ser removido pode ter exatamente um filho.
- Neste caso, fazemos o único filho de x ser filho do seu pai e depois removemos o nó x.

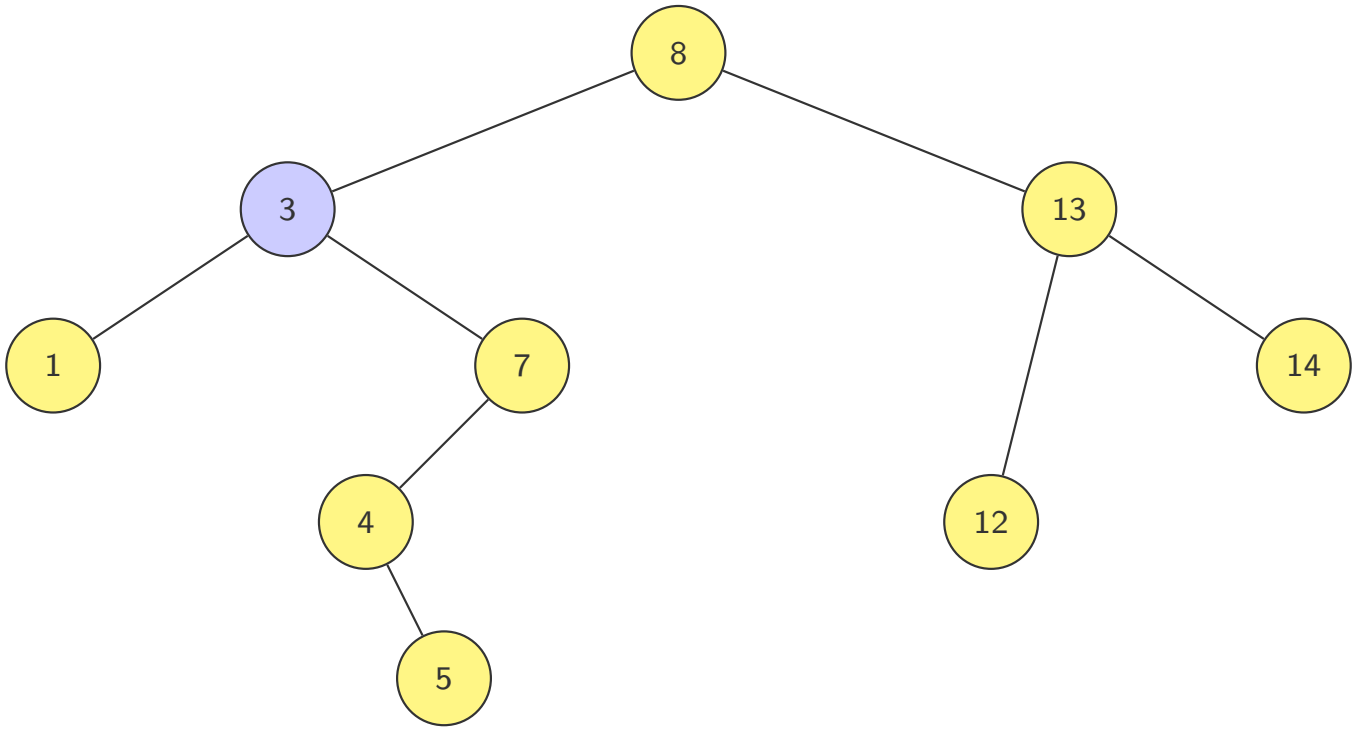
Remoção

Exemplo: removendo a chave 3



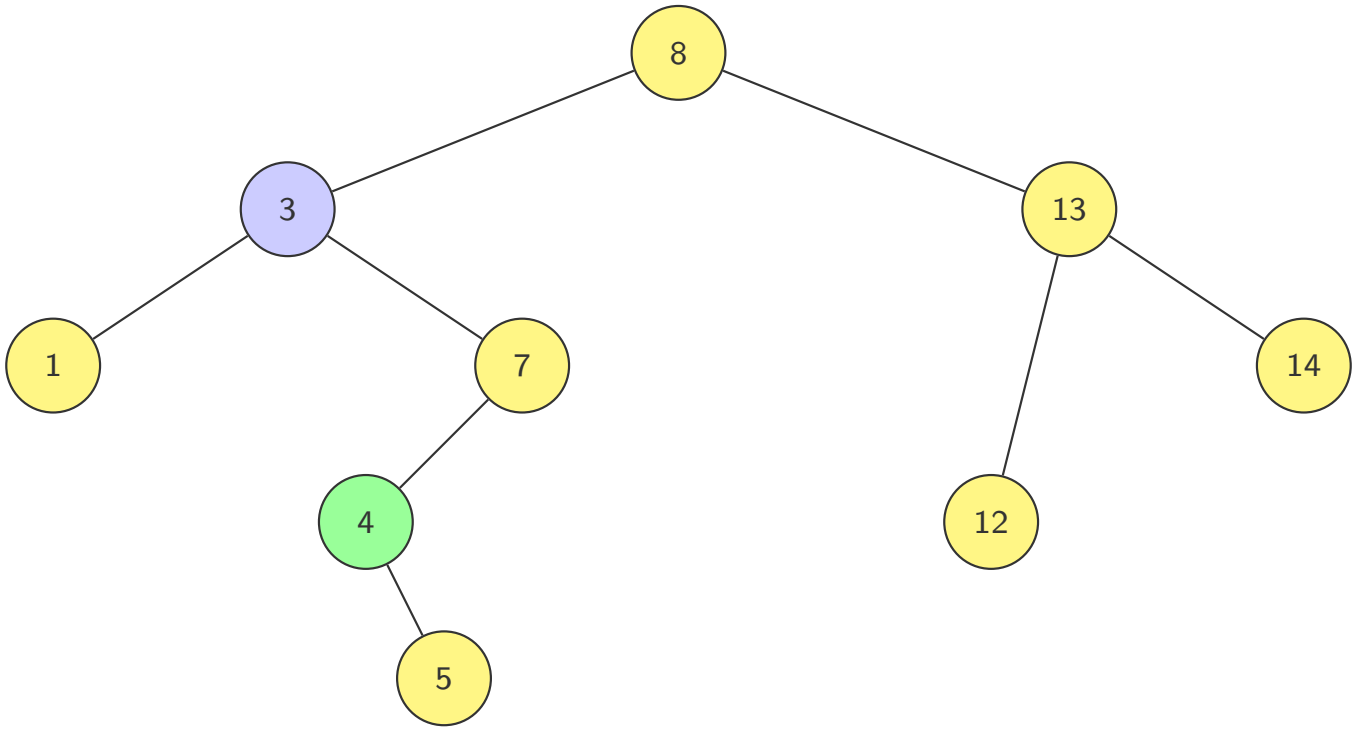
Remoção

Exemplo: removendo a chave 3



Remoção

Exemplo: removendo a chave 3

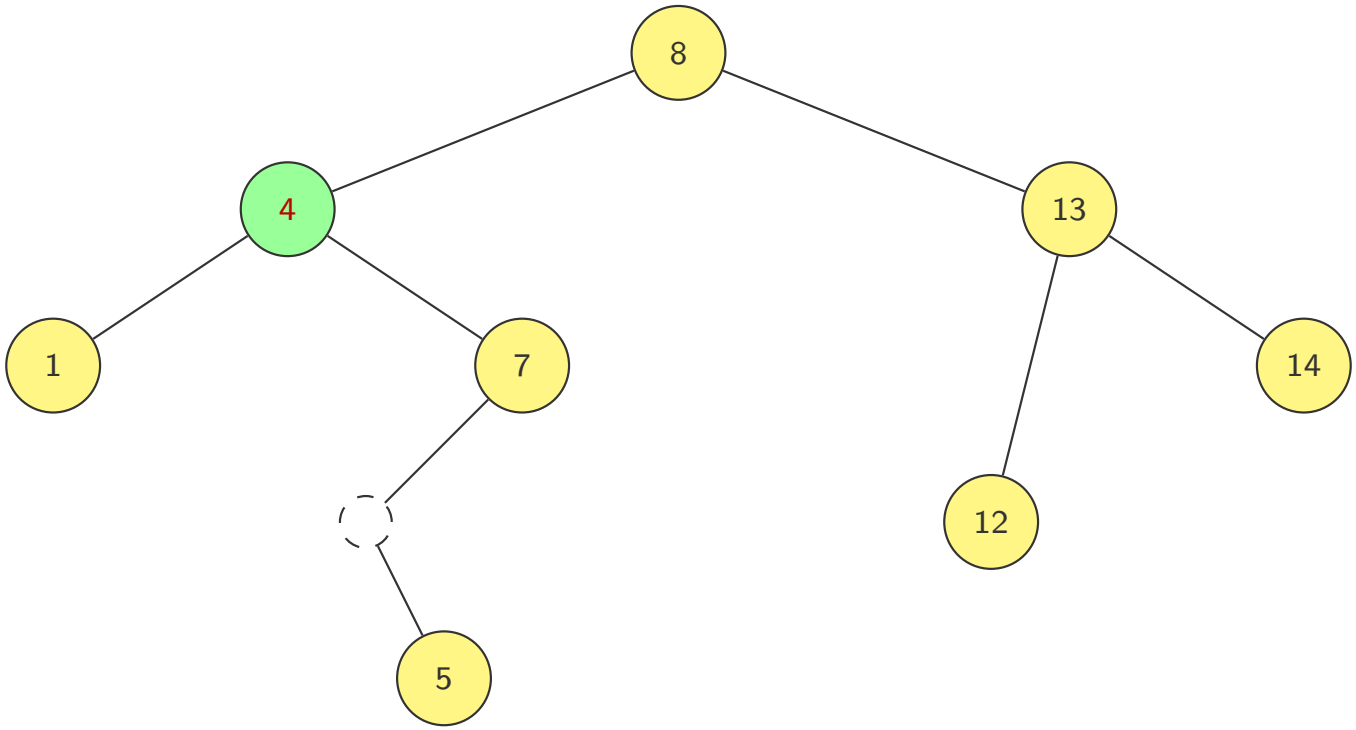


Podemos colocar o sucessor de 3 em seu lugar

- Isso mantém a propriedade da árvore binária de busca

Remoção

Exemplo: removendo a chave 3

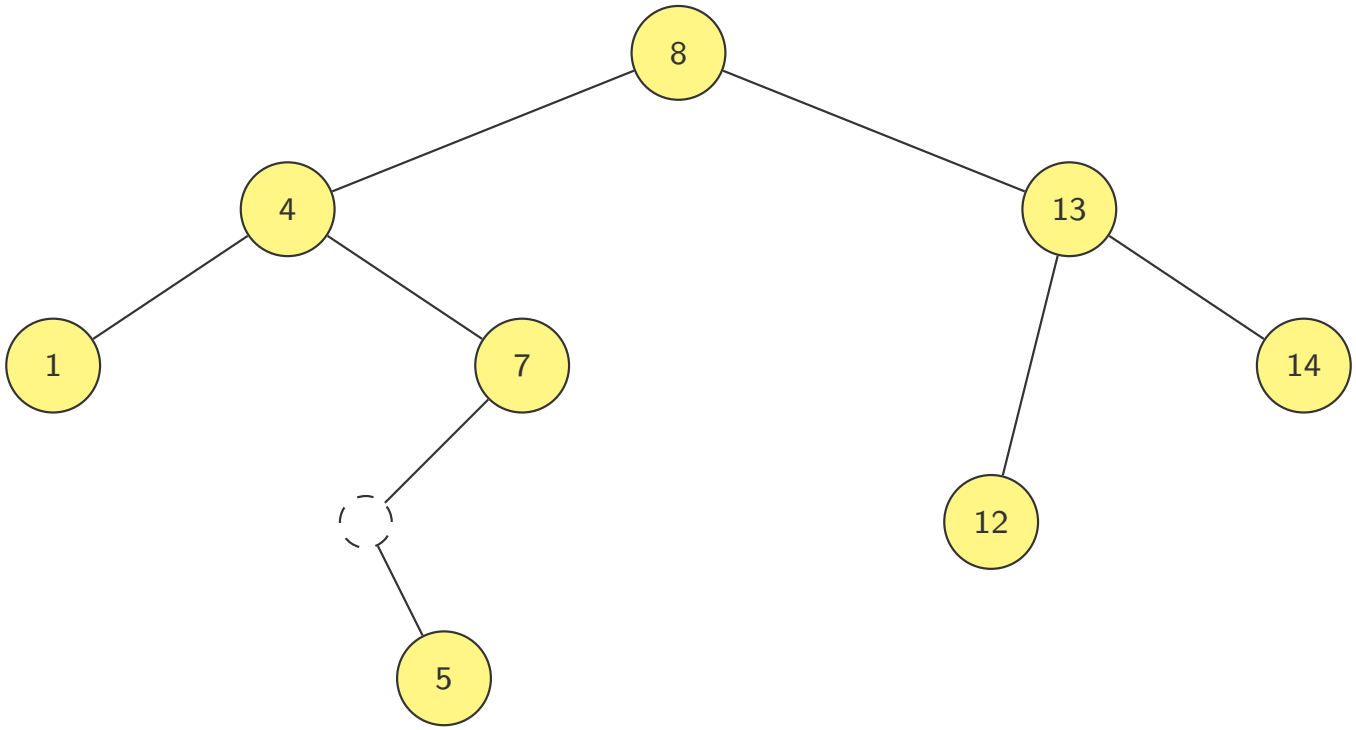


Podemos colocar o sucessor de 3 em seu lugar

- Isso mantém a propriedade da árvore binária de busca

Remoção

Exemplo: removendo a chave 3



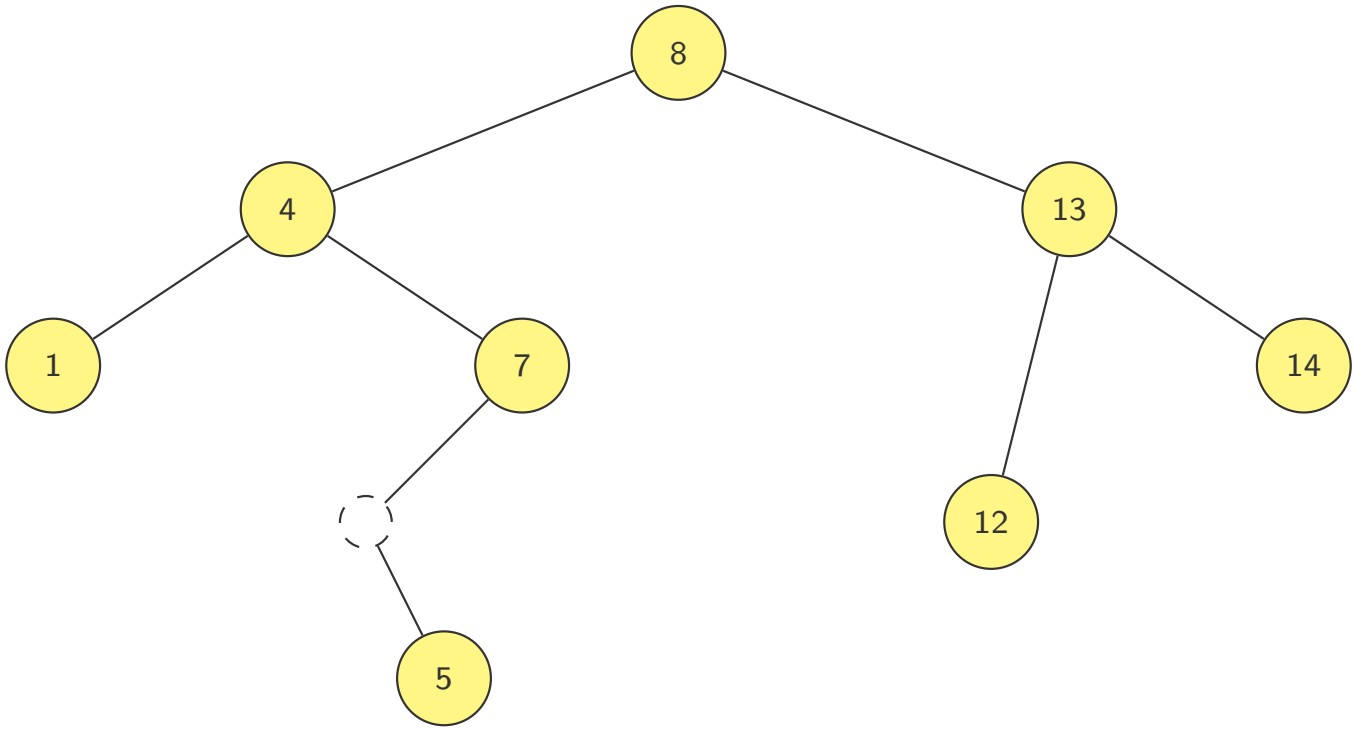
Podemos colocar o sucessor de 3 em seu lugar

- Isso mantém a propriedade da árvore binária de busca

E agora colocamos o filho direito do sucessor no seu lugar

Remoção

Exemplo: removendo a chave 3



Podemos colocar o sucessor de 3 em seu lugar

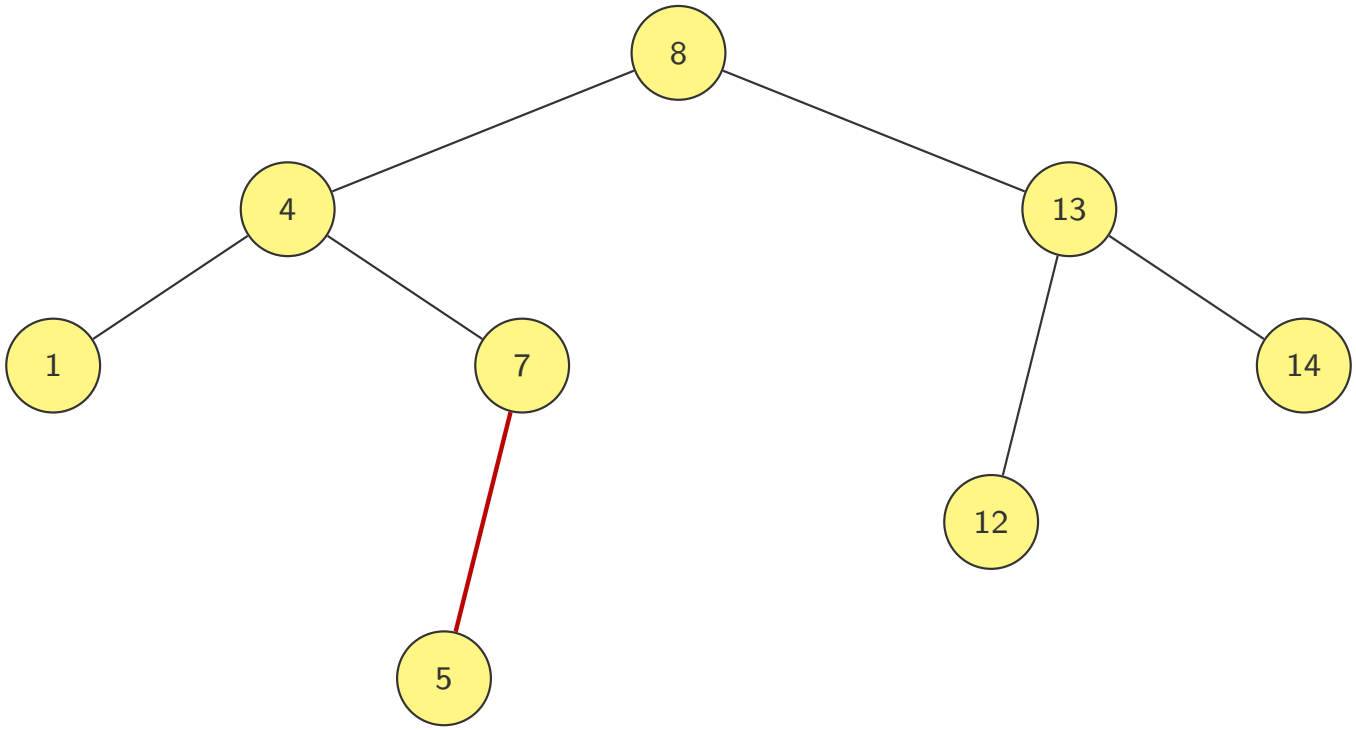
- Isso mantém a propriedade da árvore binária de busca

E agora colocamos o filho direito do sucessor no seu lugar

- O sucessor nunca tem filho esquerdo!

Remoção

Exemplo: removendo a chave 3



Podemos colocar o sucessor de 3 em seu lugar

- Isso mantém a propriedade da árvore binária de busca

E agora colocamos o filho direito do sucessor no seu lugar

- O sucessor nunca tem filho esquerdo!

Remoção

- Note que o nó a ser removido é a raiz de uma árvore. Essa raiz pode ou não ter filhos.
- Logo, convém tratar esse problema como a remoção da raiz de uma árvore.
- Seguimos os seguintes passos na remoção da raiz:
  1. Se a raiz não tiver filho direito, então o filho esquerdo assume o papel da raiz (independente de quem ele seja);
  2. Senão, basta fazer com que o nó sucessor à raiz assuma o papel da raiz.

Remoção - Implementação


```
1  // Remove o no com o valor fornecido, se ele existir
2  NoArv* abb_remove(NoArv *node, int valor) {
3      if(node == NULL) // arvore vazia
4          return NULL;
5      if(valor == node->chave) // achou o no a ser removido
6          return abb_remove_raiz(node);
7      // ainda nao achamos o valor, vamos busca-lo
8      if(valor < node->chave)
9          node->esq = abb_remove(node->esq, valor);
10     else
11         node->dir = abb_remove(node->dir, valor);
12     return node;
13 }
```

Remoção - Implementação (continuação)


```
1  // Recebe um ponteiro para a raiz de uma arvore e
2  // remove a raiz, rearranjando a arvore de mode que ela
3  // continue sendo de busca. Devolve a raiz da nova arvore
4  NoArv* abb_remove_raiz(NoArv *node) {
5      NoArv *raiz;
6      if(node->dir == NULL)
7          raiz = node->esq;
8      else {
9          NoArv *pai = node;
10         raiz = node->dir;
11         while(raiz->esq != NULL) {
12             pai = raiz;
13             raiz = raiz->esq;
14         }
15         if(pai != node) {
16             pai->esq = raiz->dir;
17             raiz->dir = node->dir;
18         }
19         raiz->esq = node->esq;
20     }
21     free(node);
22     return raiz;
23 }
```

main.c


```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include "ABB.h"

int main() {
    srand(time(NULL));
    NoArv *raiz = abb_cria();

    for(int i = 1; i <= 10; i++) {
        int valor = rand() % 1000;
        raiz = abb_insere(raiz, valor);
    }

    abb_mostrar(raiz);

    raiz = abb_libera(raiz);

    return 0;
}
```

Exercícios

Exercícios

- Conclua a implementação das funções que foram deixadas em aberto nos slides anteriores.

- Suponha que todo nó da BST tenha agora um ponteiro para nó pai. Reimplemente as operações vistas nessa aula considerando este novo ponteiro.

- Escreva uma função que receba como argumento uma BST vazia e um vetor $A[p..q]$ com $q - p + 1$ inteiros em ordem crescente e popule a BST com os inteiros do vetor A de modo que ela seja uma árvore binária de busca completa (altura igual a $\lceil \log_2 (n + 1) \rceil$). Sua função pode ter o seguinte protótipo:
  `void construirBST(BST *t, int A[], int p, int q);`

- Escreva uma função que transforme uma árvore binária de busca em um vetor crescente.

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

FIM