Aula 20: Árvores Binárias de Busca

- Conceitos básicos
- Algoritmos de busca e inserção
- Caminhos externo e interno

Problema de busca

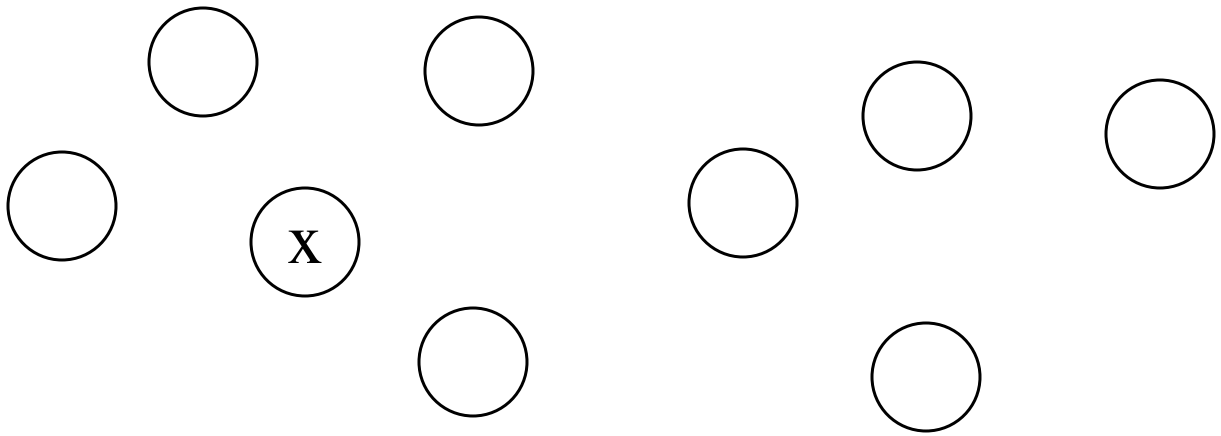
- s_i = chave

  $S = \{ s_1, ..., s_n \}, s_1 < ... < s_n$

  x = valor dado

  x pertence a S ?

  Em caso positivo, encontrar s_j tal que $x = s_j$



- Idéia: Estruturar o conjunto de chaves, de modo a facilitar a busca de x.

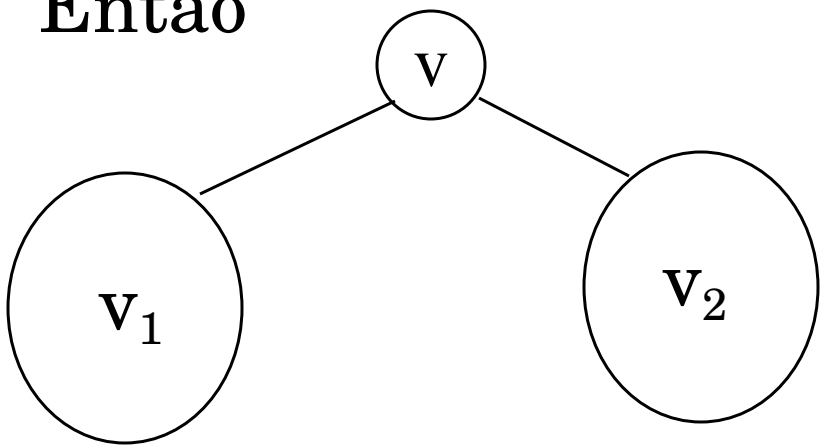
Problema de busca

Utilizar uma árvore binária rotulada T:

- (i) T possui n nós. Cada nó v corresponde a uma chave distinta s_j pertencente a S e possui como rótulo o valor $r(v) = s_j$
- (ii) Sejam v, v_1 , v_2 nós distintos de T, sendo v_1 pertencente à subárvore esquerda de v, e v_2 à subárvore direita de v. Então

$$r(v_1) < r(v) \text{ e}$$

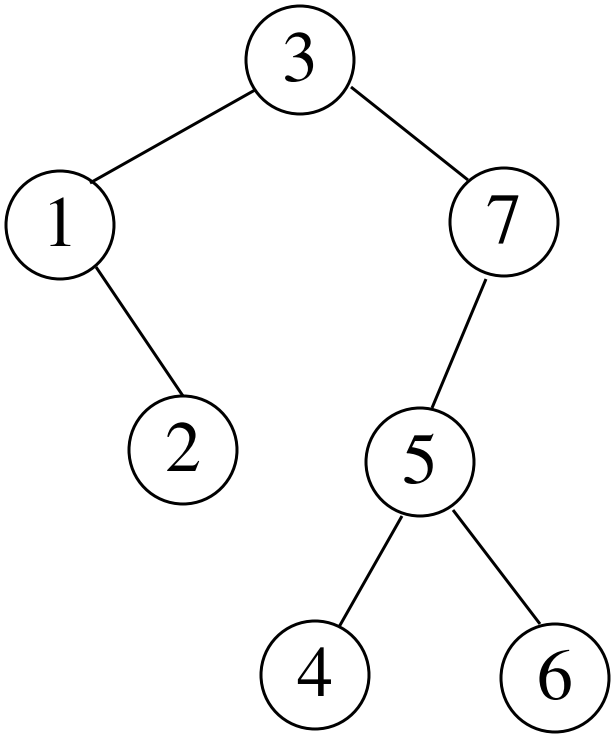
$$r(v_2) > r(v)$$

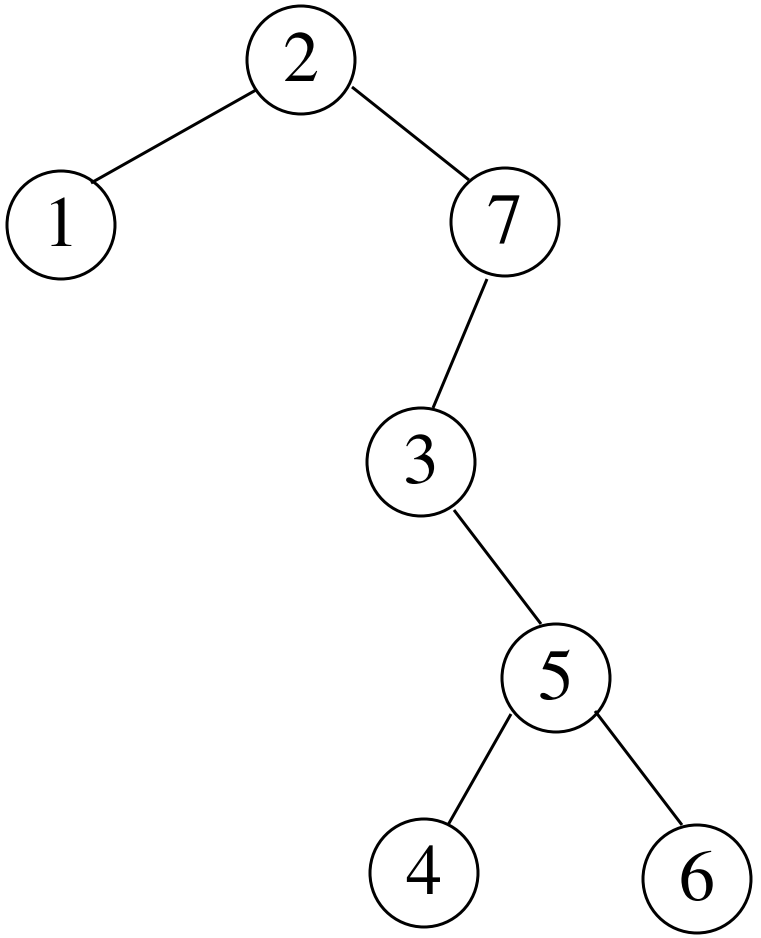


T é denominada
<u>árvore binária de busca</u>

Exemplo

- Exemplo de árvore binária de busca
  - S = { 1, 2, 3, 4, 5, 6, 7 }





Árvore Binária de Busca

- Uso da árvore binária de busca para resolver o problema de busca:
  - Objetivo: Encontrar a chave s_j tal que $x = s_j$
  - Para tal: percorrer o caminho em T, desde a raiz até s_j

Árvore Binária de Busca

Método:

- Passo inicial: considerar a raiz de T.
- Passo geral:

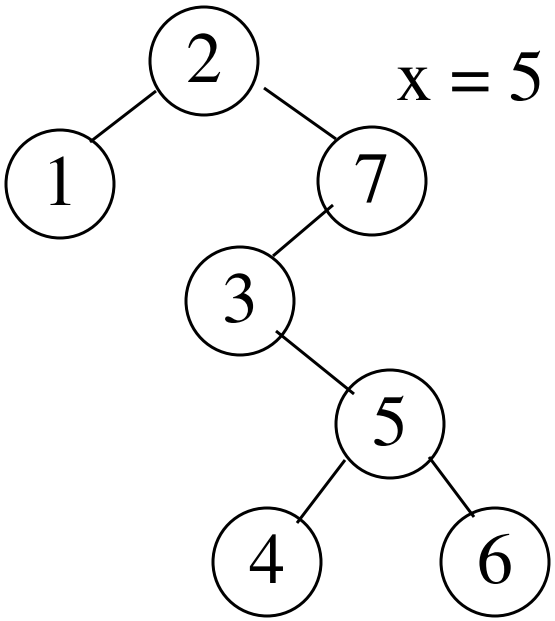
  Seja v o nó considerado.
  Se x = r (v), a busca termina (sucesso)
  Caso contrário, o novo nó a considerar será

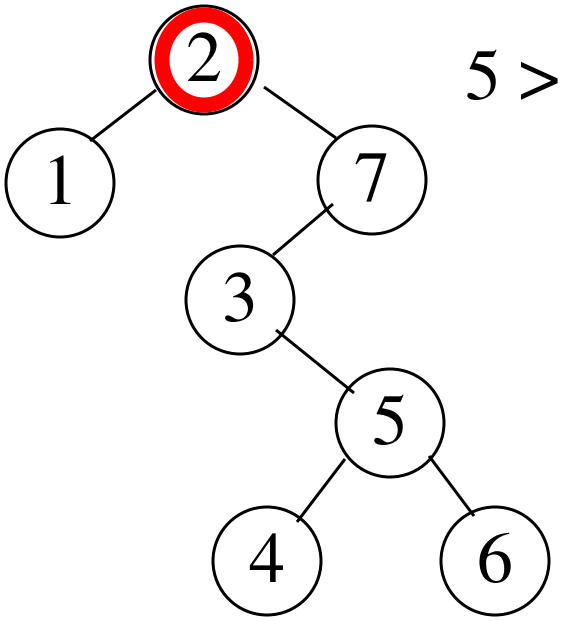
  - o filho esquerdo w_1 de v, se x < r (v) ou
  - o filho direito w_2 de v, se x > r (v).

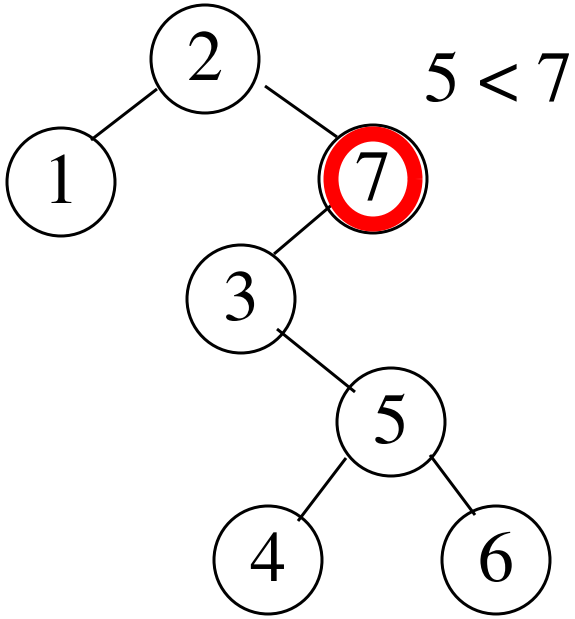
  Se não existir o nó que deveria ser considerado, w_1 ou w_2, a busca termina (insucesso).

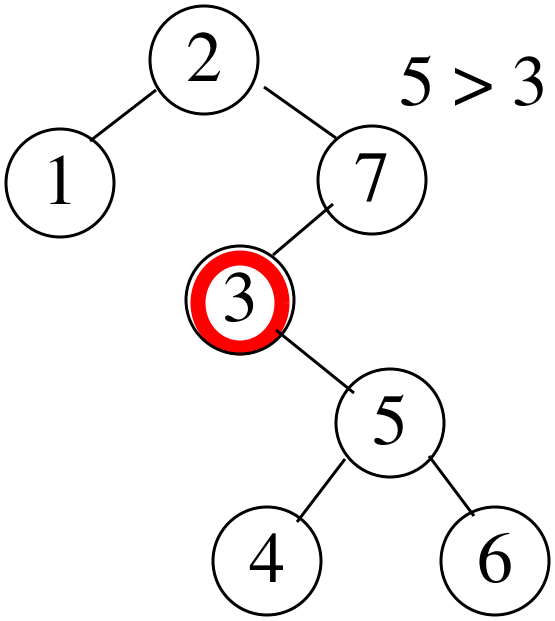
Árvore Binária de Busca

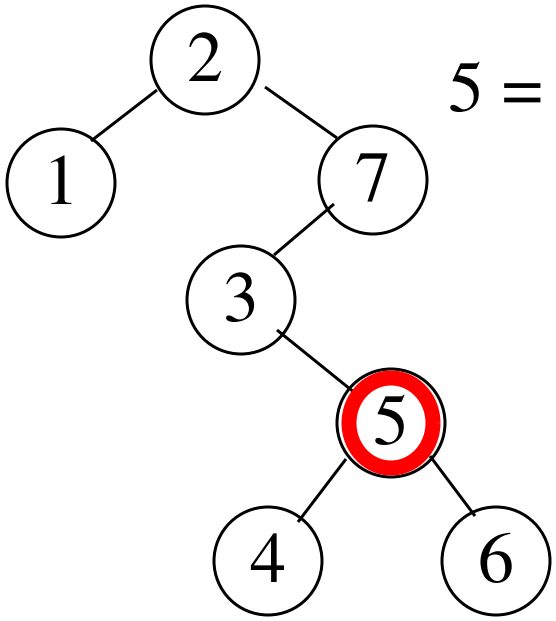
Exemplo de busca com sucesso:





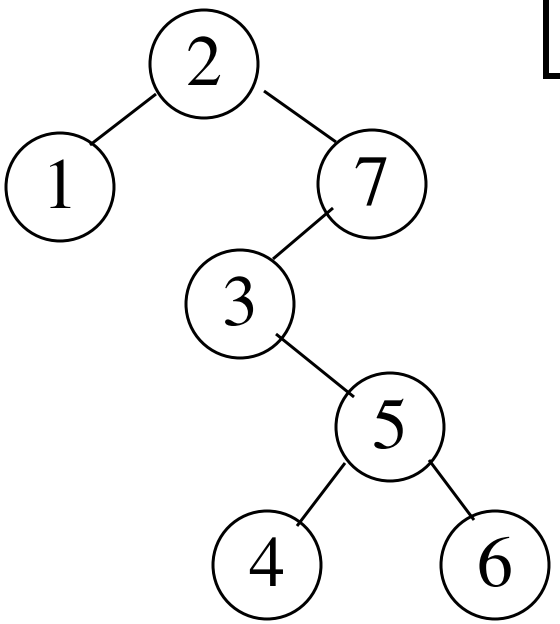






Árvore Binária de Busca

x = 5

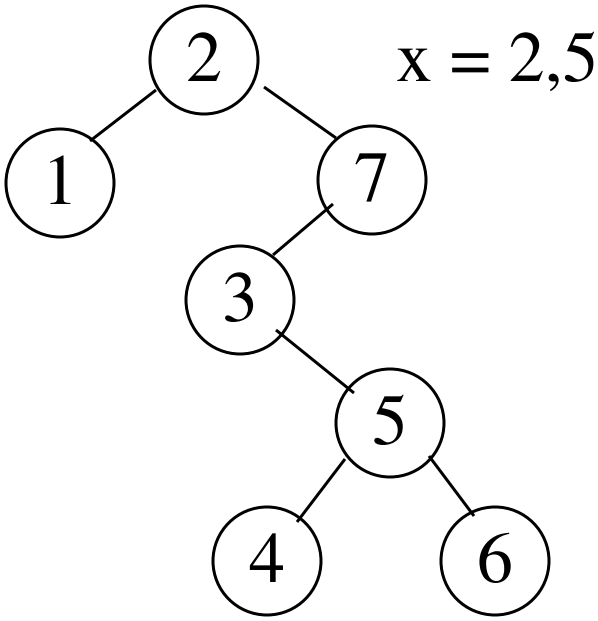


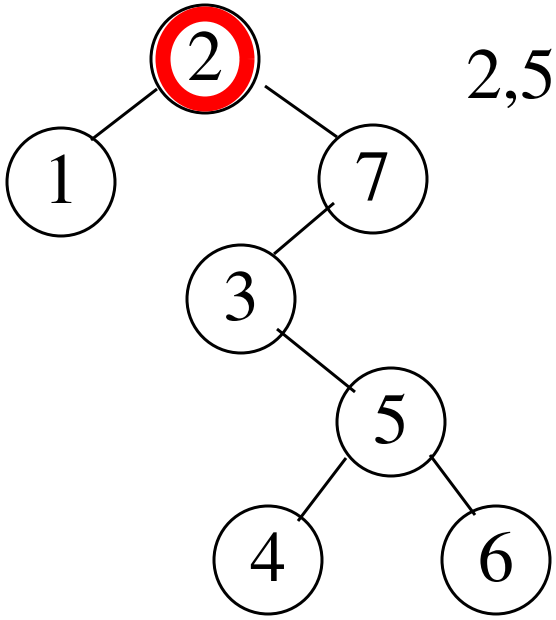
Raiz

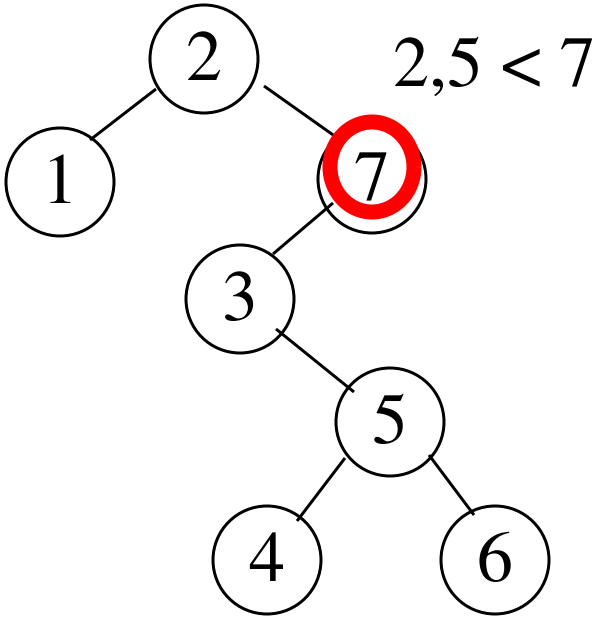
Animar Voltar

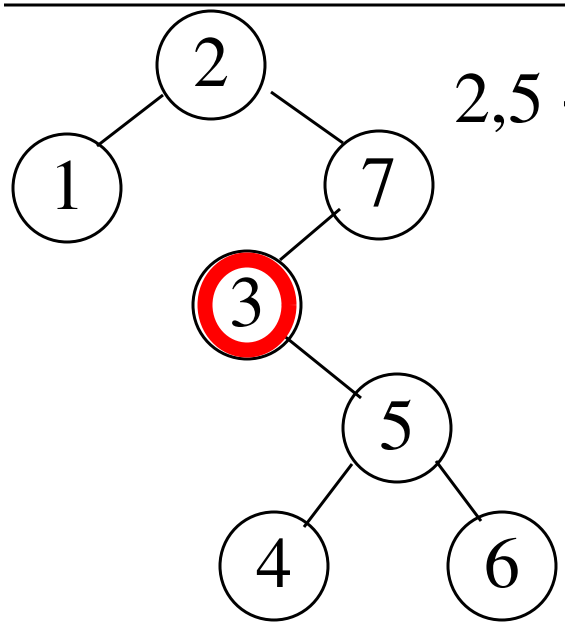
Árvore Binária de Busca

Exemplo de busca sem sucesso:



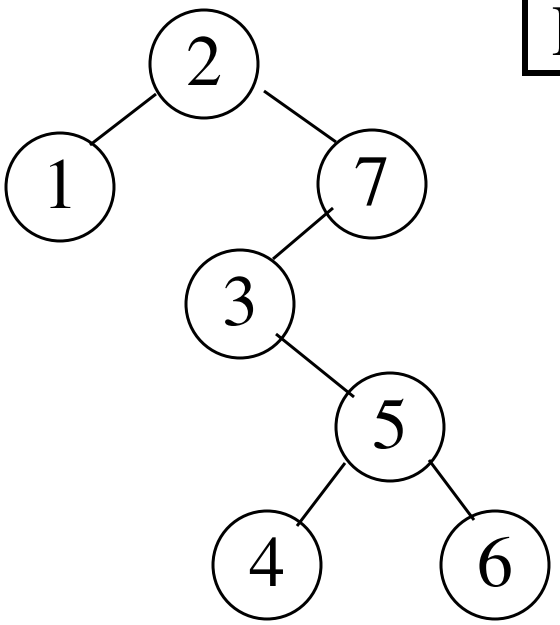






Árvore Binária de Busca

x = 2,5

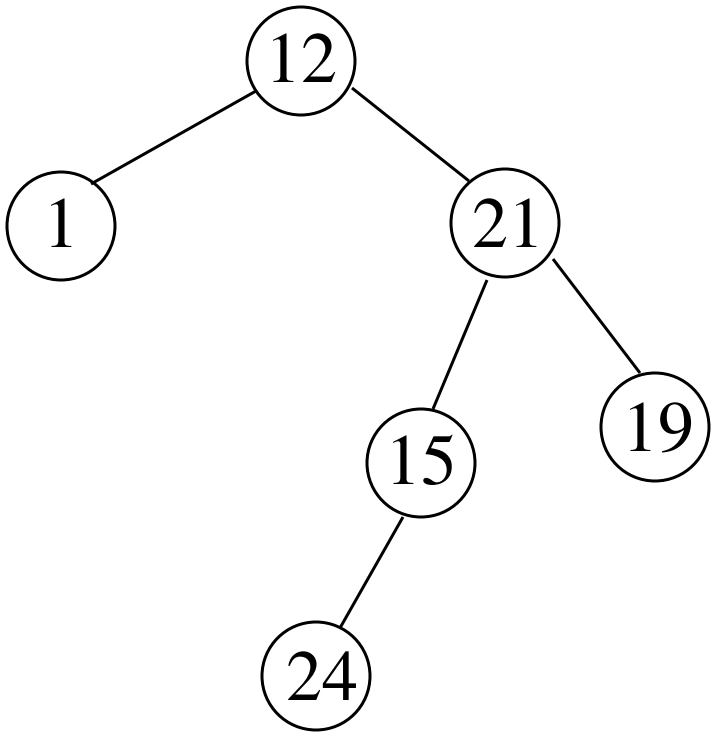


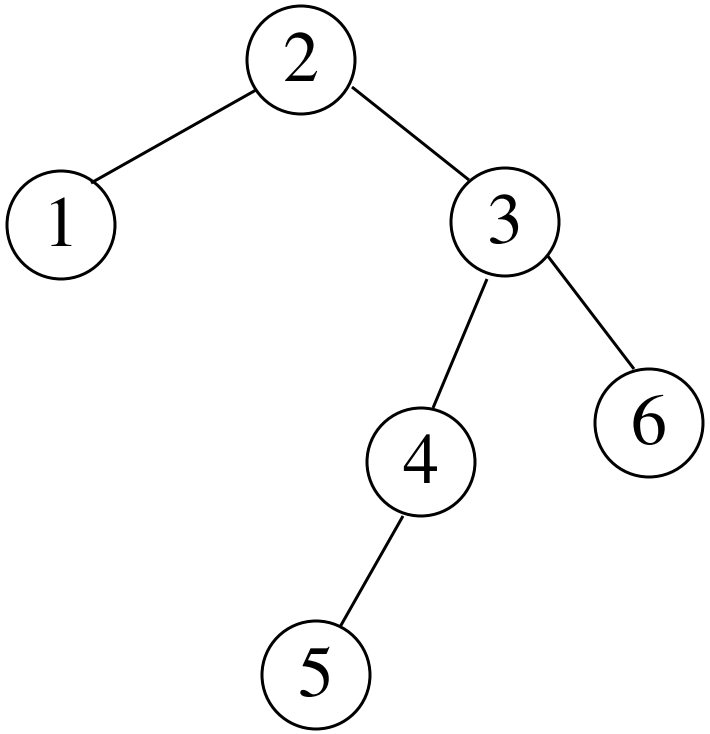
Raiz

Animar Voltar

Exercício

Verificar se as árvores binárias seguintes são árvores binárias de busca.





Tempo: 1 minuto

Solução

⇨ Nenhuma das duas árvores é binária de busca.

Algoritmo de Busca

- `ptraiz` = ponteiro para raiz da árvore
- variável `f`: natureza final da busca
  - $f = 0$ árvore vazia.
  - $f = 1, x \in S$
    Nesse caso, `pt` aponta o nó procurado.
  - $f > 1, x \notin S$

Algoritmo de Busca

Algoritmo: busca em árvore binária de busca

```
procedimento busca-árvore(x, pt, f)
    se pt = λ então f := 0
    senão se x = pt↑.chave então f := 1
          senão se x < pt↑.chave
                então
                    se pt↑.esq = λ então f := 2
                    senão
                        pt := pt↑.esq
                        busca-árvore(x, pt, f)
                senão
                    se pt↑.dir = λ então f := 3
                    senão
                        pt := pt↑.dir
                        busca-árvore( x, pt, f)

pt := ptraiz ; busca-árvore( x, pt, f )
```

Complexidade do Algoritmo

- Em cada passo, número constante de operações.
- Complexidade = número total de chamadas =
  = número de nós existentes no caminho, desde a raiz de T até o nó v onde termina o processo.
- Pior caso: v pode se encontrar a uma distância O(n) da raiz.
- Complexidade = O(n)
- Conclusão: complexidade = altura da árvore

melhor complexidade <-> árvore de altura mínima
melhor árvore <-> árvore completa

Árvore Binária de Busca Ótima

- A árvore binária de busca ótima é a árvore completa.
- Complexidade = altura.
- Lema: seja uma árvore binária completa com n nós e altura h. Então,

$$2^{h-1} \leq n \leq 2^{h} - 1$$

- Prova: o valor $2^{h} - 1$ ocorre quando a árvore é cheia.
  Nesse caso, o número de nós em cada nível é igual ao dobro do número de nós do nível anterior.
  O valor 2^{h-1} corresponde ao caso em que há apenas um nó no último nível.
- Portanto, complexidade = O (log n) (busca com sucesso)
  Para a busca sem sucesso vale a mesma expressão.

Exercício

Seja S = { 1, 2, 3, 4, 5, 6, 7 } um conjunto de chaves.

Desenhar uma árvore binária de busca T para S, nos seguinte casos:

(i) T possui altura máxima.

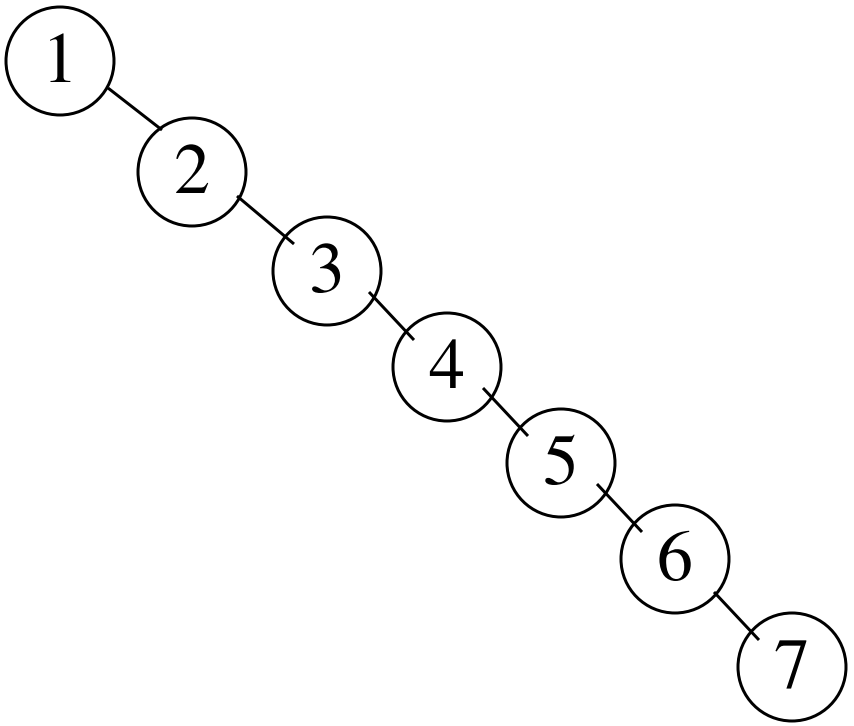
(ii) T possui altura mínima.

Quantas árvores distintas existem em cada caso?

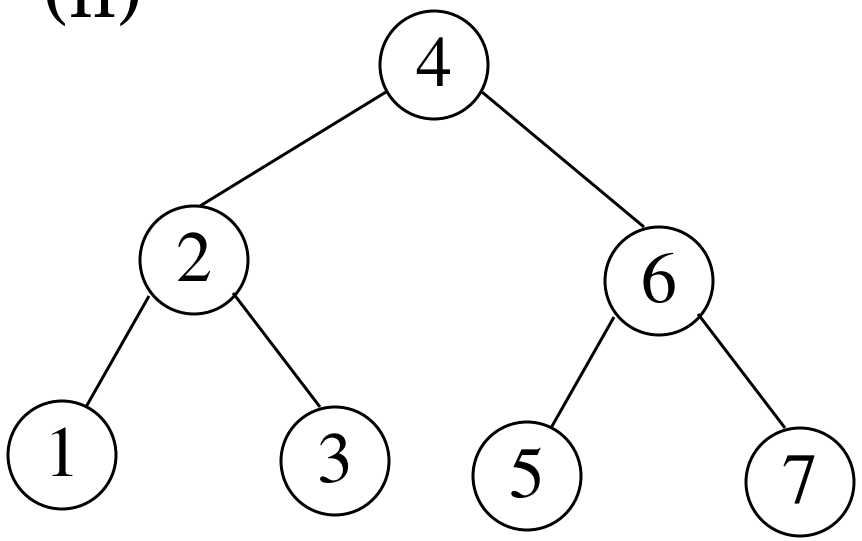
Tempo: 5 minutos

Solução

(i)



(ii)



Número de árvores distintas:

(i) $2^6 = 64$

(ii) 1

Inserção

- Inserção em árvores binárias de busca
- s_i = chave

  $S = \{ s_1, ..., s_n \}$, $s_1 < ... < s_n$

  T = árvore binária de busca para S

  x = chave a ser inserida em T
- Idéia do algoritmo:
  - utilizar o algoritmo de busca (procedimento busca-árvore)
  - verificar se $x \in S$
  - em caso positivo, inserção inválida (chave duplicada)
  - em caso negativo, a chave de valor x será o rótulo de algum novo nó w, situado à esquerda de v, se f = 2, ou à direita de v, quando f = 3. O nó v é aquele onde o procedimento busca-árvore termina.

Algoritmo de Inserção

Algoritmo: inserção em árvore binária de busca

```
pt := ptraiz; busca-árvore( x, pt, f )
se f = 1 então
            ¨inserção inválida¨
senão ocupar (pt1)
            pt1↑.chave := x; pt1↑.info := novo-valor;
            pt1↑.esq := λ; pt1↑.dir := λ;
            se f = 0 então ptraiz := pt1
            senão se f = 2
                  então
                     pt↑.esq := pt1
                  senão
                     pt↑.dir := pt1
```

Complexidade de inserção = complexidade da busca

Árvore Binária de Busca

- Construção de árvore binária de busca
- S = { s_1, ..., s_n }, conjunto de chaves
  Construir uma árvore binária de busca T para S.
- Idéia de um método:
  Construir uma árvore binária de busca T(i) para { s_1, ..., s_i } iterativamente.

  Passo inicial:

  seja T(1) a árvore formada pelo único nó s_1.

  Passo geral:

  para j=2, ..., n, utilizando o algoritmo de inserção, inserir em T(j - 1) o nó de rótulo s_j, denotando por T(j) a árvore assim obtida.
  Ao final, T = T(n).

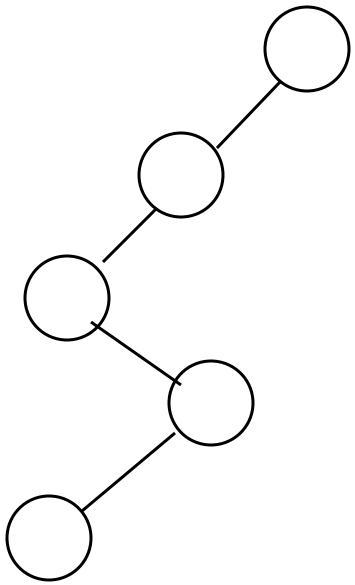
Árvore Binária de Busca

- Complexidade do algoritmo de inserção:

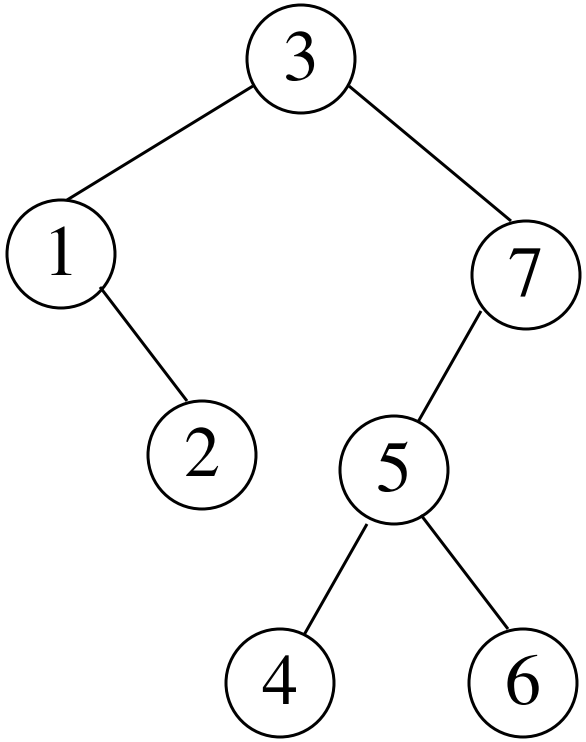
  n . altura da árvore

  Pior caso, O (n^2)

- Pior caso, árvore zigue zague.



Exemplo



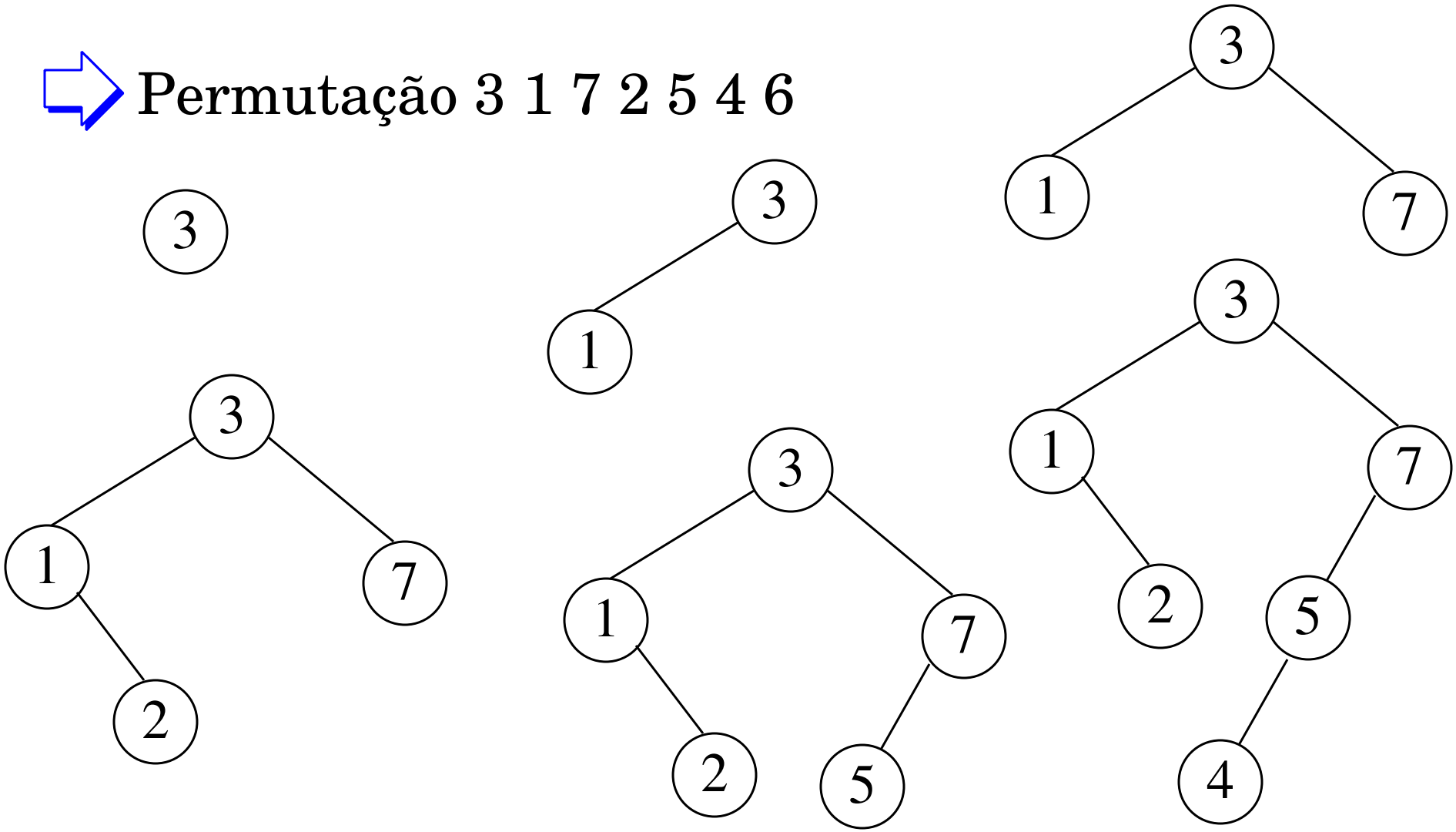
- Pode ser construída com a permutação das chaves 3 1 7 2 5 4 6

- Há várias outras permutações que conduzem a esta mesma árvore. Exemplo: 3 7 5 6 1 2 4

- A permutação 3 1 2 5 4 7 6 não conduz à construção dessa árvore.

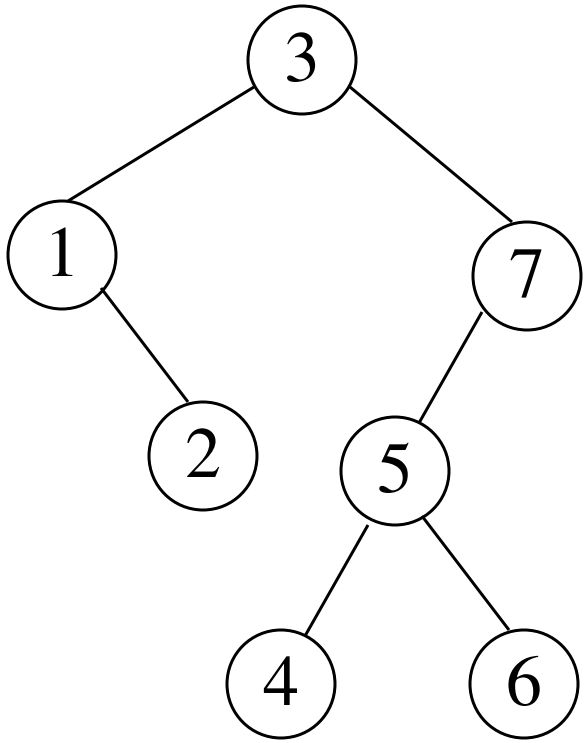
Árvore Binária de Busca

Permutação 3 1 7 2 5 4 6



Árvore Binária de Busca

- permutação 3 1 7 2 5 4 6



Árvore Binária de Busca

- Construção de uma árvore binária de busca completa
- Aplicar a construção geral, com uma ordenação adequada para o conjunto de chaves.
- s_0 , s_{n+1} = chaves fictícias, $s_0 < s_1$ e $s_{n+1} > s_n$, definidas como já inseridas.
- A cada passo, inserir em T uma nova chave s_j de índice médio entre duas chaves s_i e s_j , já inseridas, e tais que não hajam chaves já inseridas entre s_i e s_j . Definir s_j como já inserida e repetir o processo.
- Complexidade: O (n log n)

Exercício

- Sejam p_1, p_2 duas permutações de um conjunto de chaves S, e T_1, T_2 as árvores binárias de busca correspondentes a p_1, p_2 respectivamente. Então

  $p_1 \neq p_2 \iff T_1 \neq T_2$

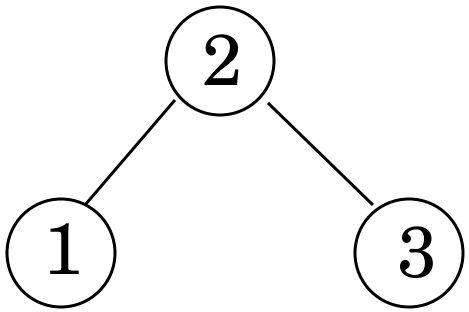
- Provar ou dar contra exemplo.

Tempo: 2 minutos

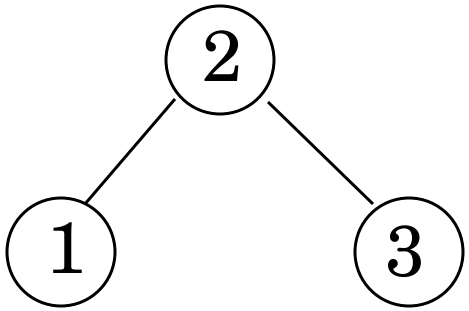
Solução

Contra exemplo: $p_1 = 2\ 1\ 3$, $p_2 = 2\ 3\ 1$

T_1



T_2



Árvore Binária de Busca

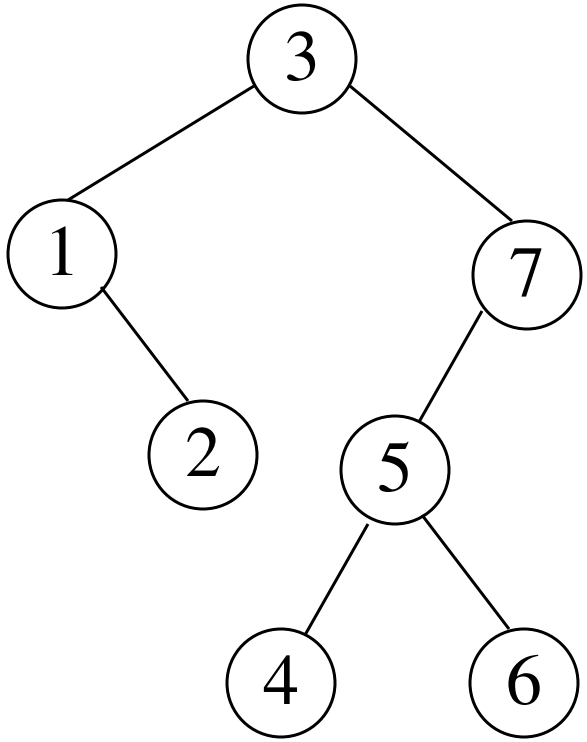
- Comprimento do Caminho Interno

- S = { s_1 , ..., s_n } , conjunto de chaves
  T = árvore binária de busca para S
  l_k = nível em T da chave s_k

  $\sum_{1 \le k \le n} l_k$ = comprimento do caminho interno de T = I(T)

Árvore Binária de Busca

Exemplo:



$$I(T) = 1 + 2 + 2 + 3 + 3 + 4 + 4 = 19$$

O comprimento de caminho interno representa o número total de comparações efetuadas pelo algoritmo de busca, para o acesso a todos os nós da árvore.

Modelagem

Modelagem da busca sem sucesso

- R = conjunto dos valores possíveis para as chaves

  $R_0 = \{ x \in R \mid x < s_1 \}$

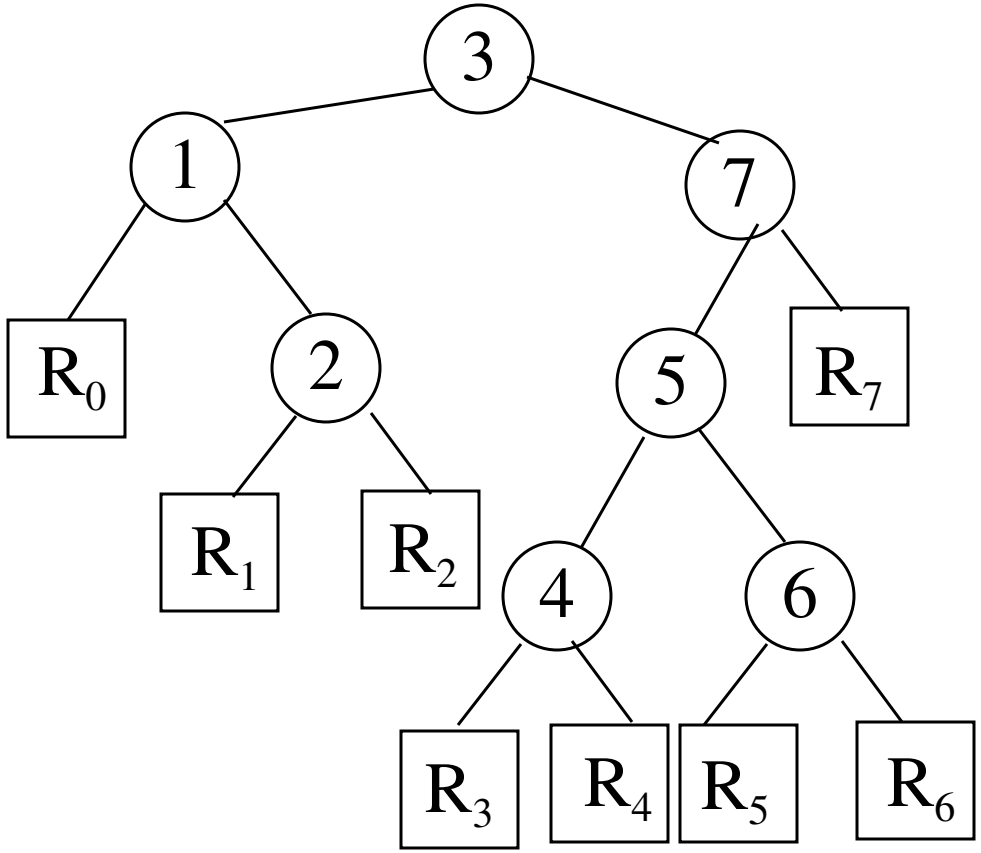
  $R_n = \{ x \in R \mid x > s_n \}$

  $R_j = \{ x \in R \mid s_j < x < s_{j+1} \}, j = 1, ..., n-1$

- Os n + 1 conjuntos R_j, $0 \leq j \leq n$, representam os diferentes intervalos onde se localizam as chaves correspondentes às buscas sem sucesso.
- Os conjuntos R_j correspondem as n + 1 subárvores vazias da árvore binária de busca para $\{ s_1, ..., s_n \}$
- Os conjuntos R_j ocorrem da esquerda para a direita em T segundo valores crescentes de seus índices.

Modelagem

- Nós externos



- As chaves correspondem a círculos (nós internos).
- As ausências de chaves são as folhas da árvore, representadas por quadrados (nós externos).
- Os nós externos são as folhas da árvore.
- A árvore com nós externos é estritamente binária.

Modelagem

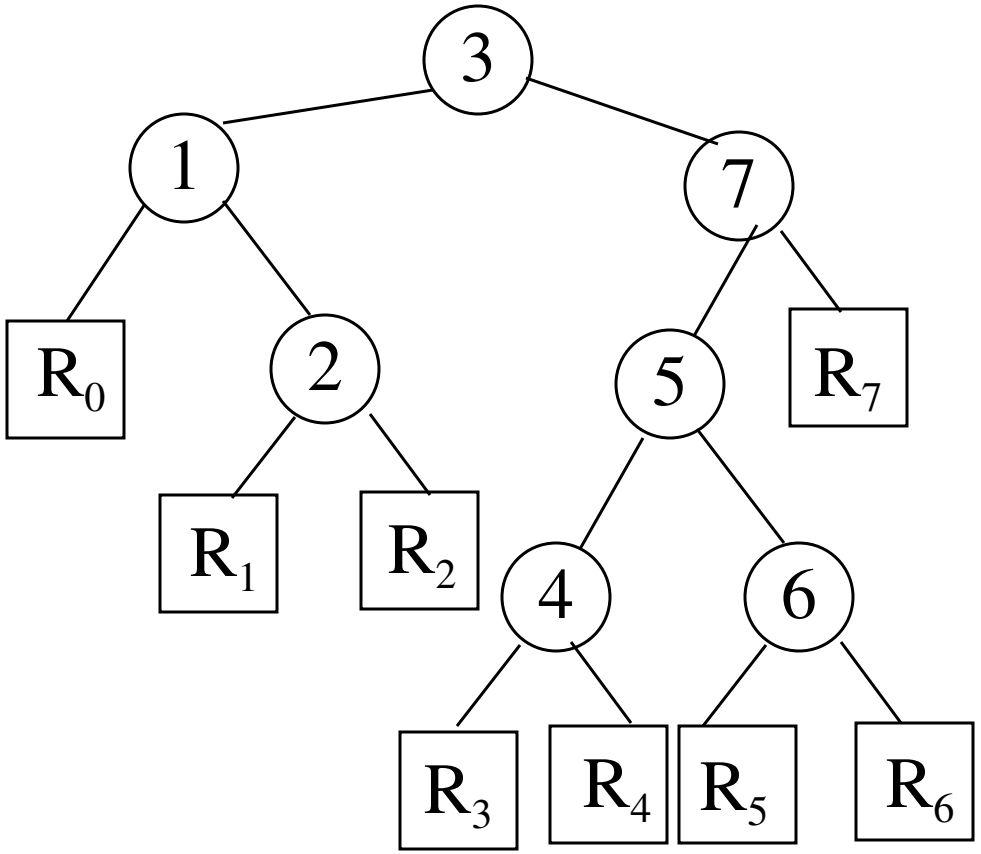
- Comprimento de caminho externo

- Seja T uma árvore binária de busca com nós externos.
  $R_0, \ldots, R_n$ = nós externos de T
  l_k' = nível do nó R_k em T

$$\sum_{0 \le k \le n} (l_k' - 1) = \text{comprimento do caminho externo de T} = E(T)$$

Modelagem

Exemplo de cálculo do comprimento do caminho externo:



E (T) = 2 + 2 + 3 + 3 + 4 + 4 + 4 + 4 = 26

Modelagem

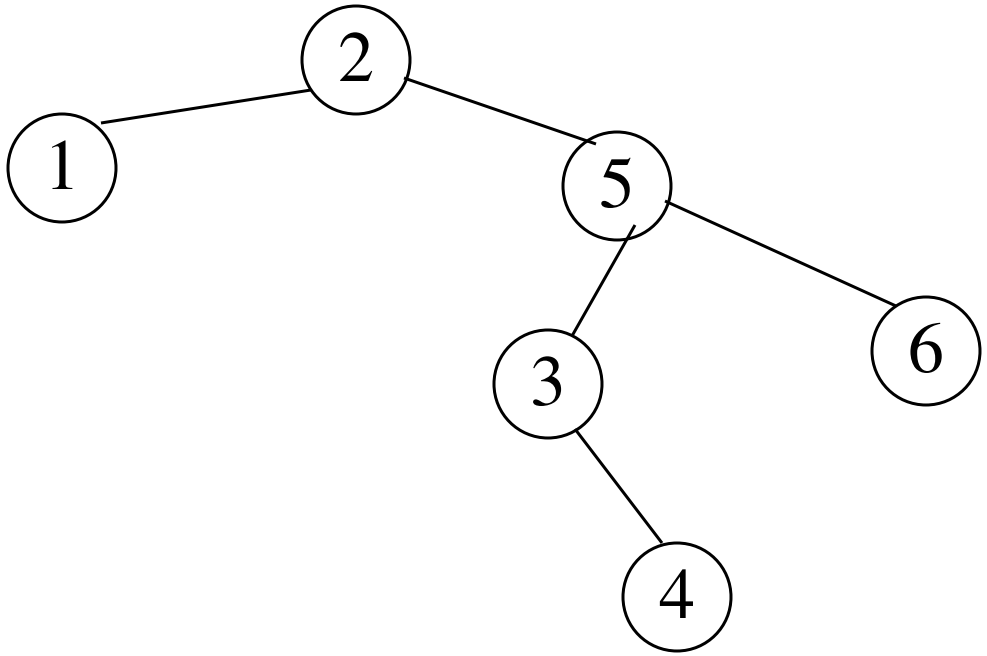
- O comprimento de caminho externo representa o número total de comparações para as buscas sem sucesso, considerando que cada intervalo R_j seja acessado uma vez.

- Os valores I(T) e E(T) exprimem a qualidade da árvore (quanto menor, melhor).

Exercício

Exercício

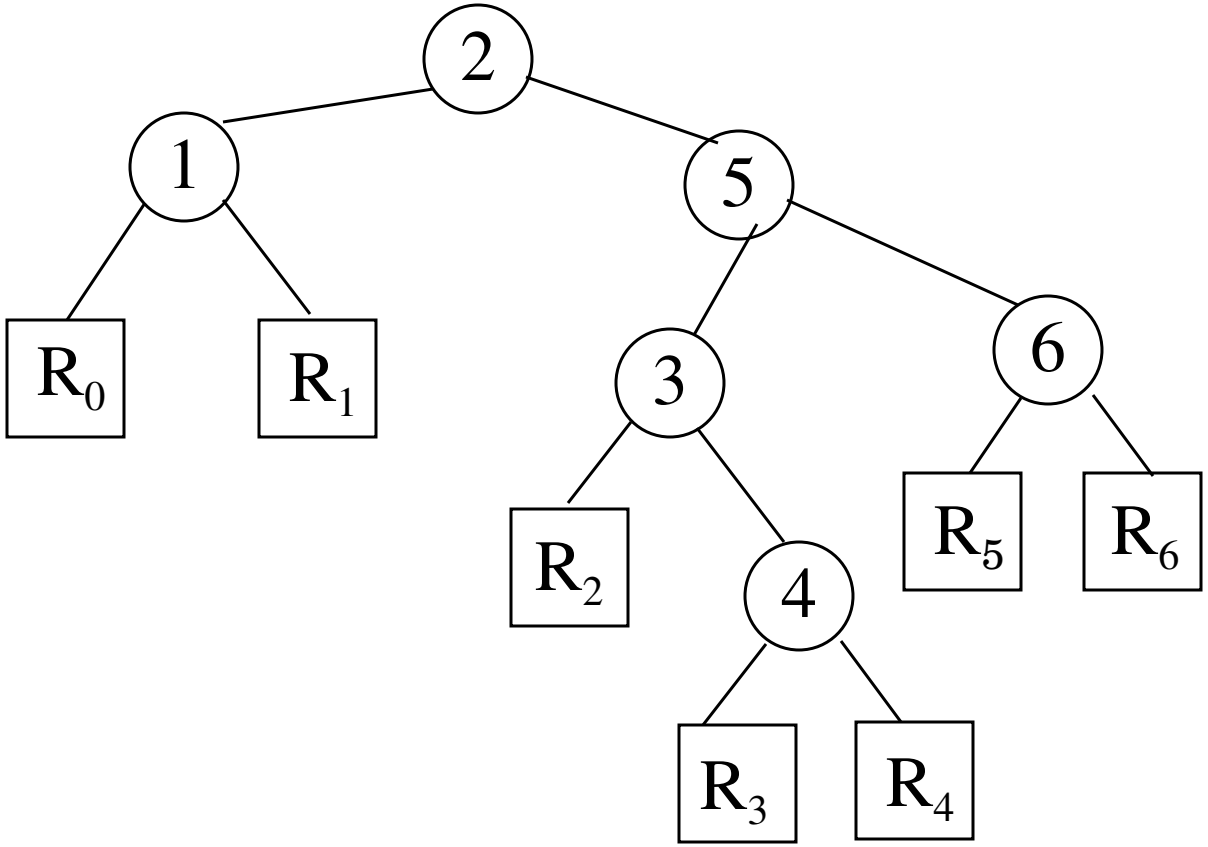
Determinar o comprimento de caminho interno e externo da seguinte árvore binária T.



Tempo: 3 minutos

Solução

Solução



I(T) = 1 + 2 + 2 + 3 + 3 + 4 = 15

E(T) = 2 + 2 + 3 + 3 + 3 + 4 + 4 =21