Estrutura de Dados e Algoritmos

Árvore Binária de Pesquisa

Dr. Nilton Correia da Silva, Dr. Fabricio Ataides Braz

Conceito

- Uma árvore binária também pode ser utilizada para armazenar uma tabela de símbolos para fins de minimizar a complexidade do algortimo de pesquisa destes símbolos;
- Quando utilizamos Árvores Binárias com este propósito, dizemos que tais árvores são **Árvores Binárias de Pesquisa (ABP)**;

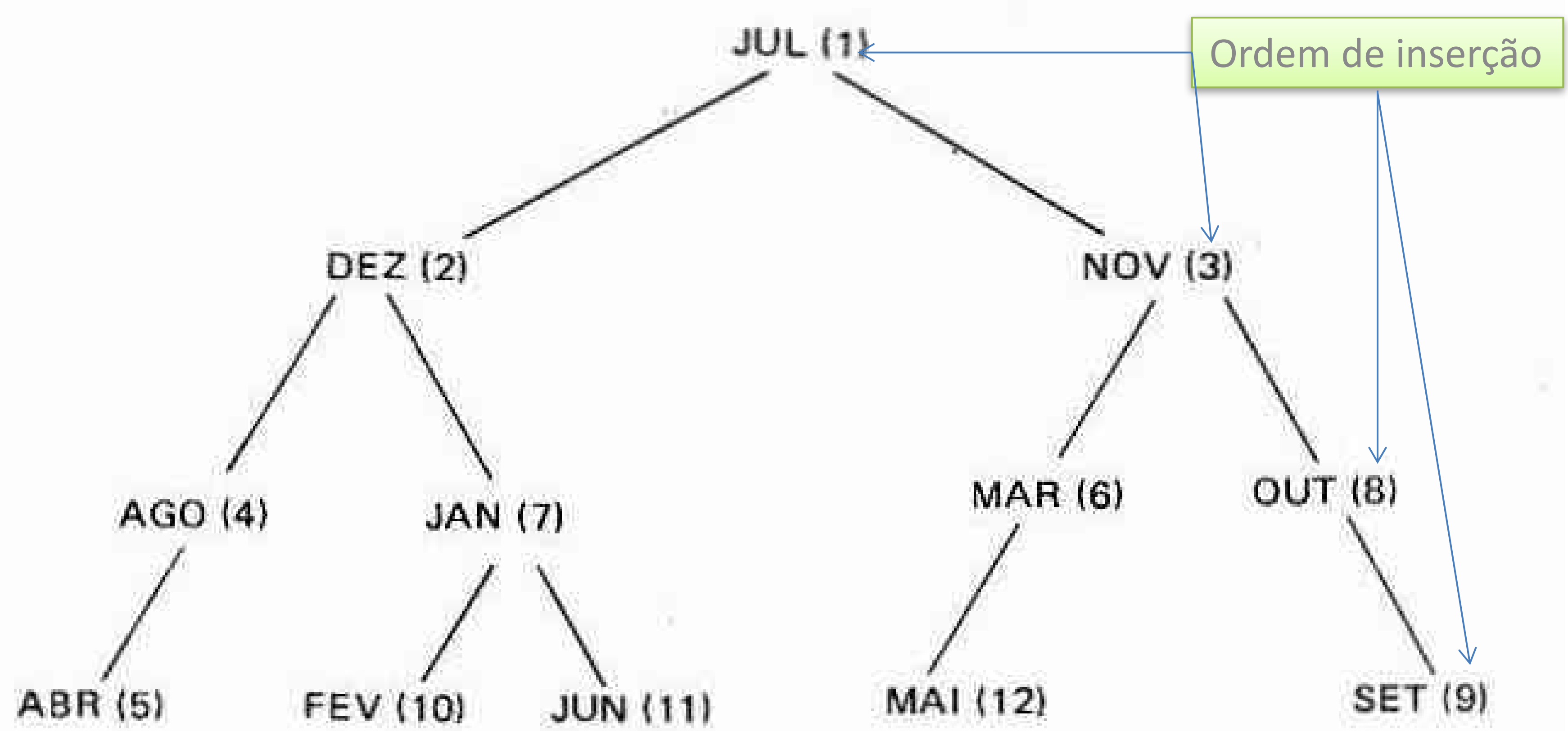
Construção

- Para se construir uma ABP, há de se considerar uma relação de ordem entre seus elementos. Isto é, dados dois elementos e1 e e2 quaisquer, deve ser possível decidir se e1<e2 ou e1=e2 ou e1>e2.
- Assim, se os elementos forem numéricos, podemos considerar a relação de ordem implícita nos mesmos. Caso sejam alfanuméricos, podemos considerar a ordem alfabética (lexicográfica) entre eles.

Construção

- Exemplo: Criar uma ABP baseada na ordem alfabética dos meses do ano:

Jul, Dez, Nov, Ago, Abr, Mar, Jan, Out, Set, Fev, Jun, Mai

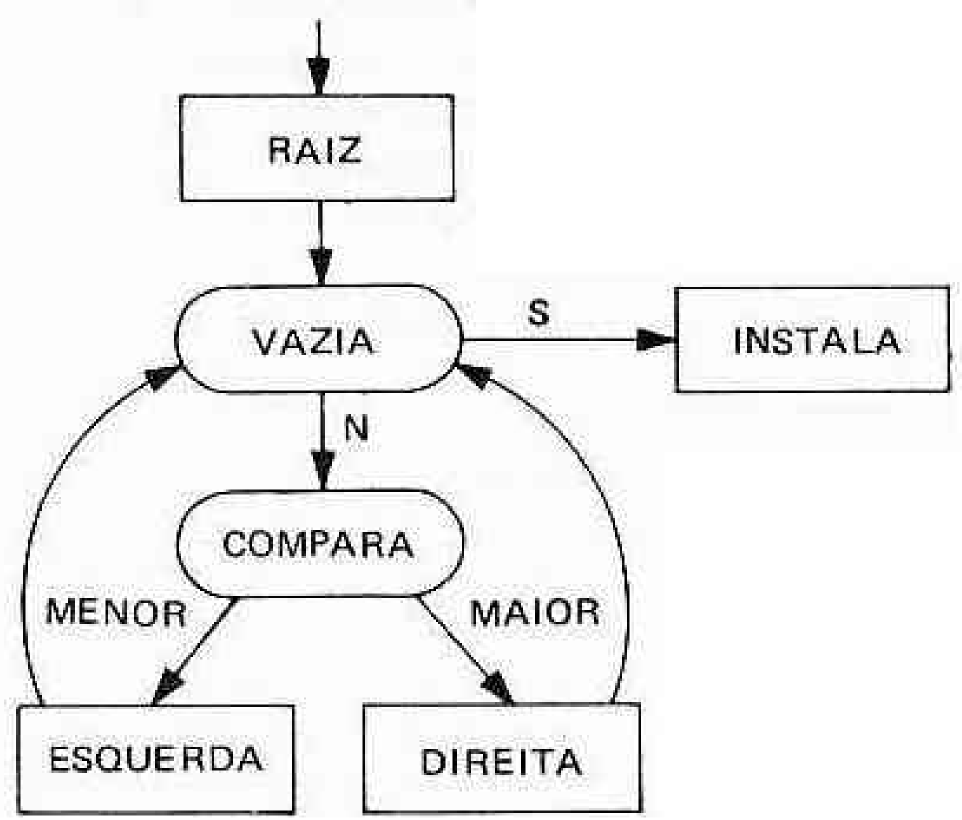


Construção

- A instalação de um símbolo ***s*** em uma ABP ***A*** segue o seguinte princípio:
  a. Se ***A*** for vazia, instalar ***s*** na raiz de ***A***;
  b. Caso contrário:
     a. Se ***s*** for menor que raiz de ***A***: Instalar ***s*** na subárvore da esquerda de ***A***;
     b. Se ***s*** for maior que raiz de ***A***: Instalar ***s*** na subárvore da direita de ***A***;

Construção

- A instalação de um símbolo ***s*** em uma ABP ***A*** segue o seguinte princípio:

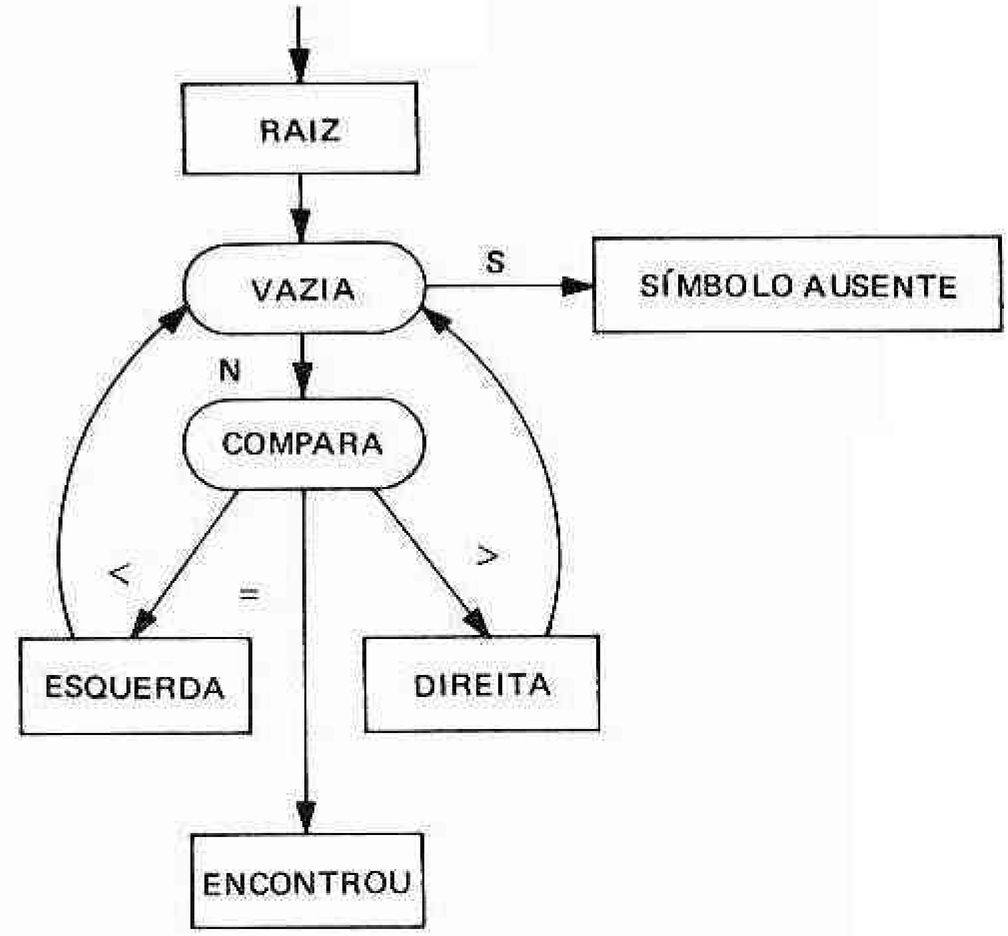


Construção

```
instala(TABP *A, TSimbolo s)
        se (A≠Nulo)
                se (s < A->Simbolo)
                        instala(A->Esquerda, s)
                senão
                        instala(A->Direita, s)
                fimse
        senão
                A = AloqueNo()
                A->Simbolo = s
                A->Esquerda = Nulo
                A->Direita = Nulo
        fimse
fim
```

Pesquisa de Um Símbolo

- A operação de localização (pesquisa) de um símbolo é semelhante à inserção:



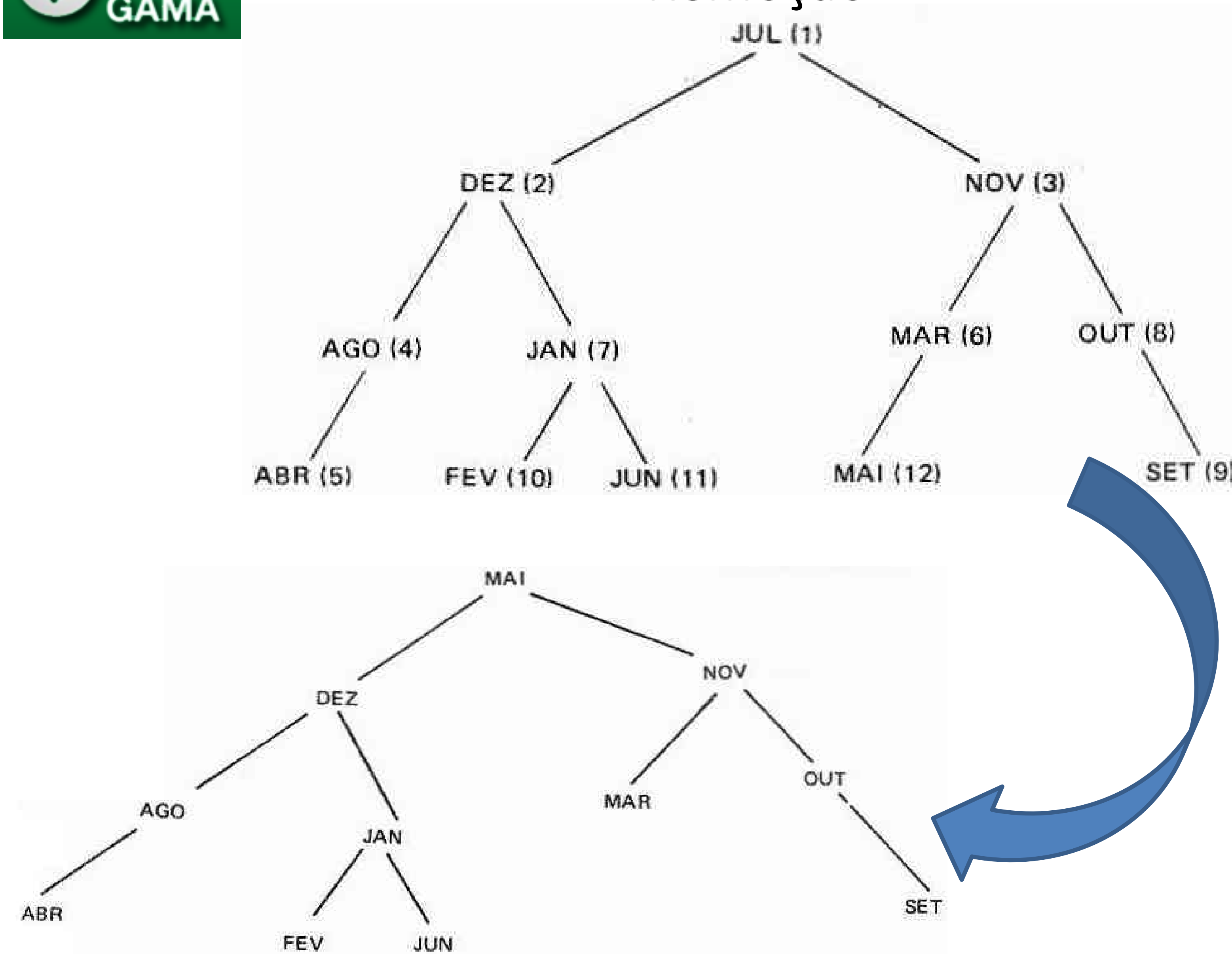
Pesquisa de Um Símbolo

```
TABP *procura(TABP *A, TSimbolo s)
        se (A≠Nulo)
                se (s = A->Simbolo)
                        retorne A;
                senão se (s < A->Simbolo)
                        retorne procura(A->Esquerda, s)
                senão
                        retorne procura (A->Direita, s)
                fimse
        senão
                retorne Nulo
        fimse
fim
```

Remoção

- A terceira operação básica em uma ABP é a remoção de um símbolo de sua estrutura.
- Esta é a operação que carrega um pouco mais de dificuldade, pois pode demandar uma reestruturação da ABP.
- Para ilustrar este fato, suponha que queiramos excluir o mês de julho de nossa ABP de exemplo.

Remoção



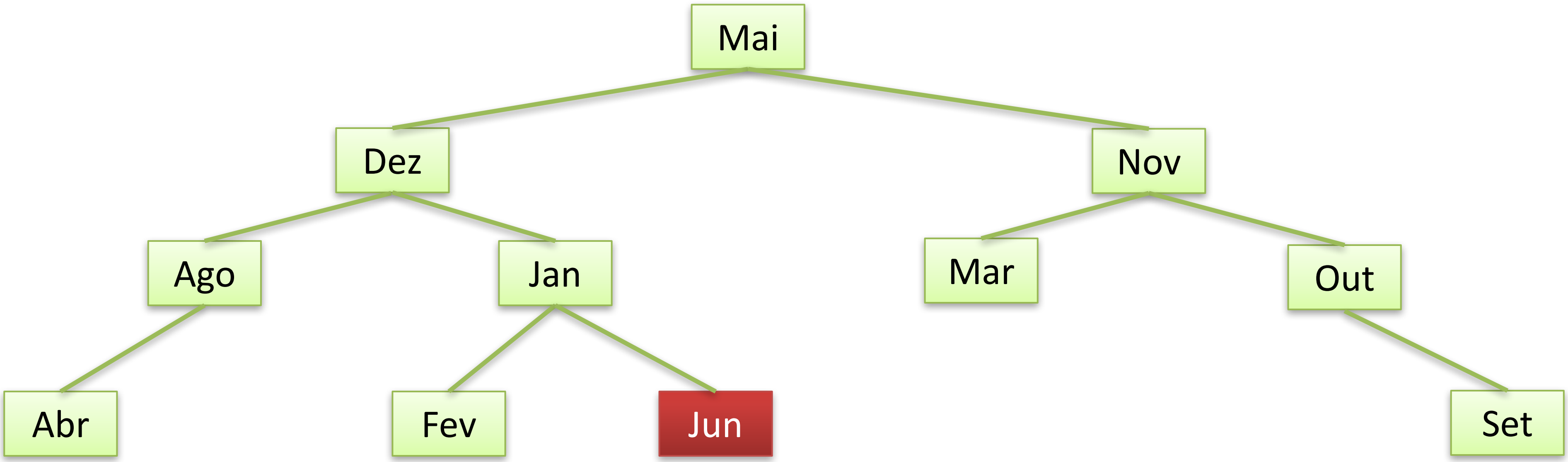
JUL (1)
DEZ (2)
NOV (3)
AGO (4)
JAN (7)
MAR (6)
OUT (8)
ABR (5)
FEV (10)
JUN (11)
MAI (12)
SET (9)
MAI
DEZ
NOV
AGO
JAN
MAR
OUT
ABR
FEV
JUN
SET

Remoção

- Em uma operação de remoção devemos considerar duas situações distintas:
  a. O nó a ser removido possui zero ou uma subárvore;
  b. O nó a ser removido possui as duas subávores;

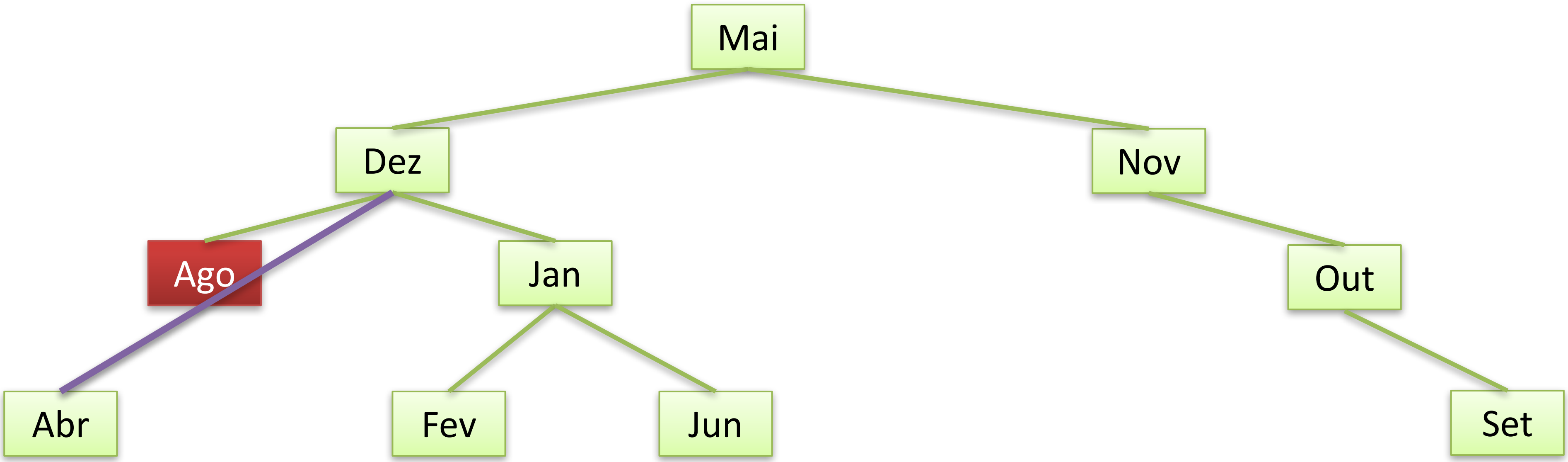
Remoção de Nó Sem Subárvore

- Simplesmente Remova o Nó:



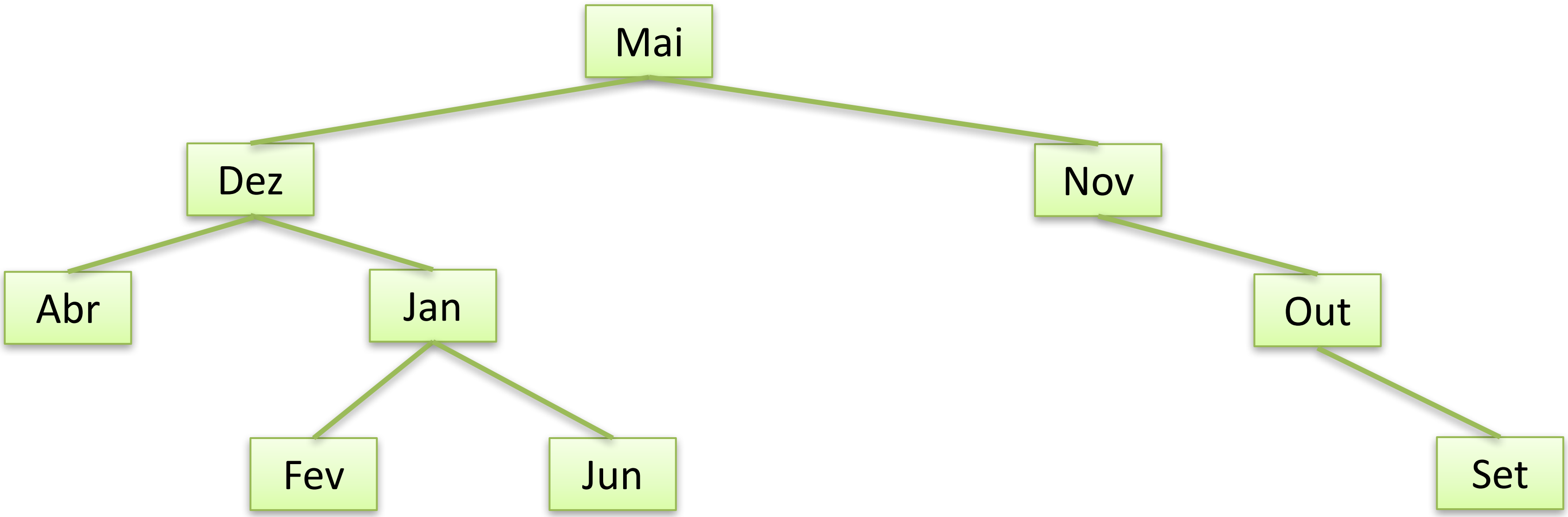
Remoção de Nó Com Uma Subárvore

- A raiz da subárvore passa a ocupar a área do nó:



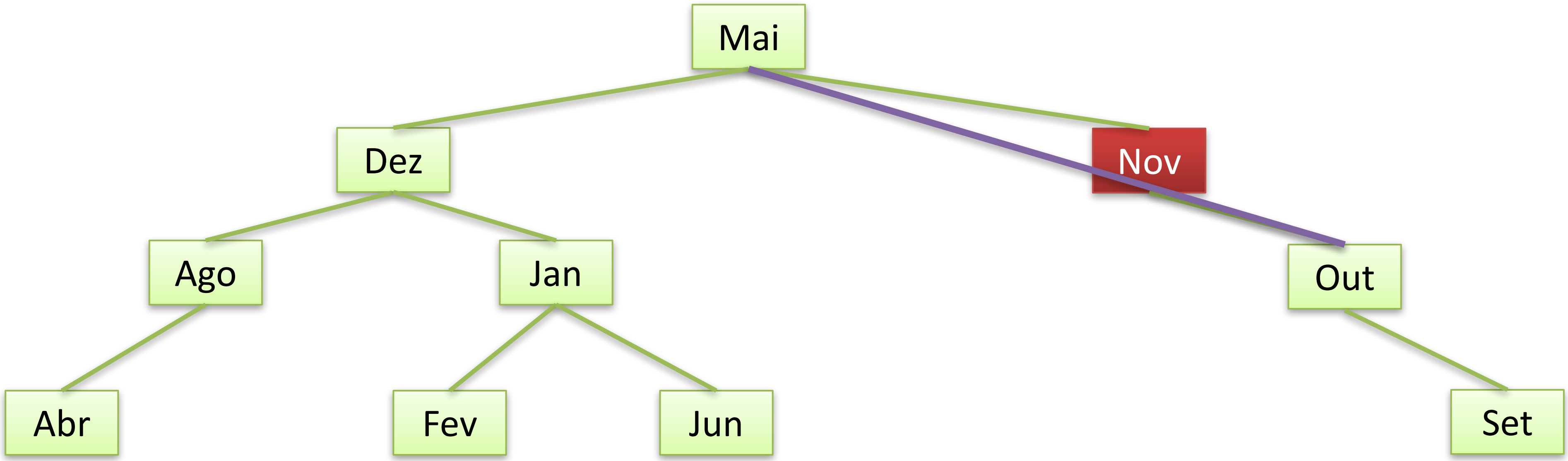
Remoção de Nó Com Uma Subárvore

- A raiz da subárvore passa a ocupar a área do nó:



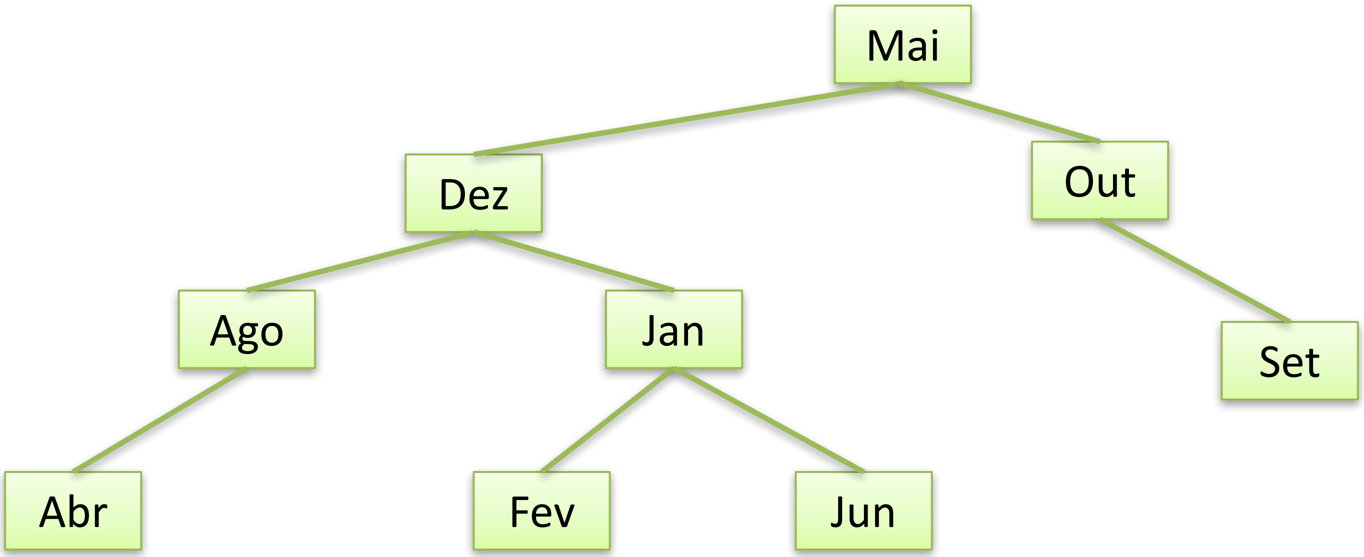
Remoção de Nó Com Uma Subárvore

- A raiz da subárvore passa a ocupar a área do nó:



Remoção de Nó Com Uma Subárvore

- A raiz da subárvore passa a ocupar a área do nó:

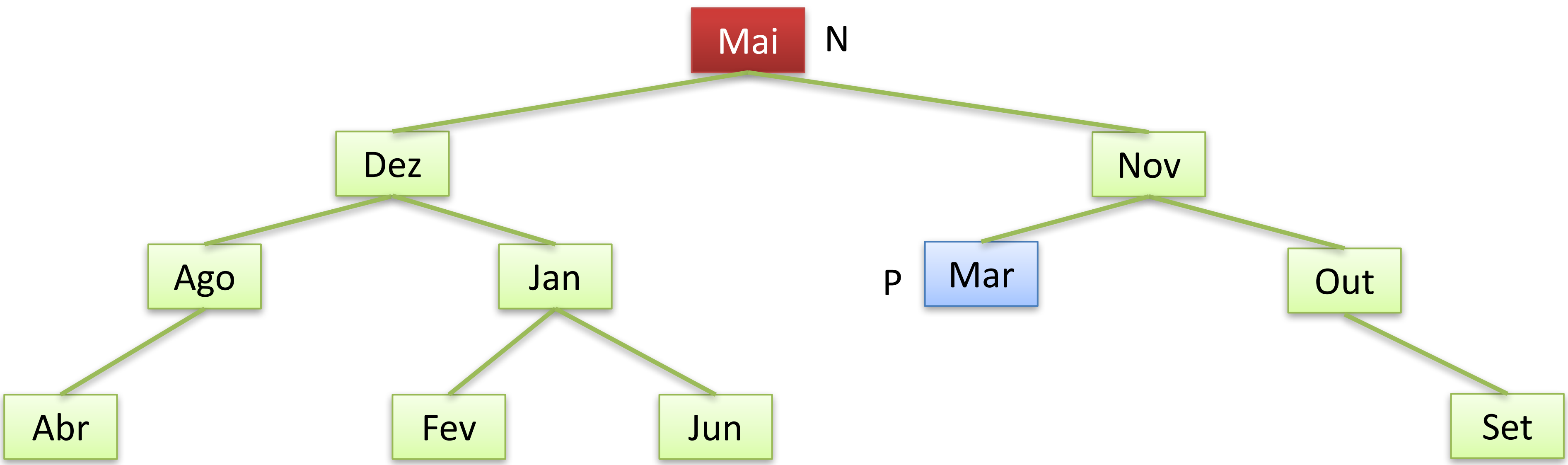


Remoção de Nó Com Duas Subárvores

- Seja:
  - N: O nó a ser excluído;
  - P: O nó com valor imediatamente maior que valor de N;

a. Substituir valor de N pelo valor de P;

b. Remover P.

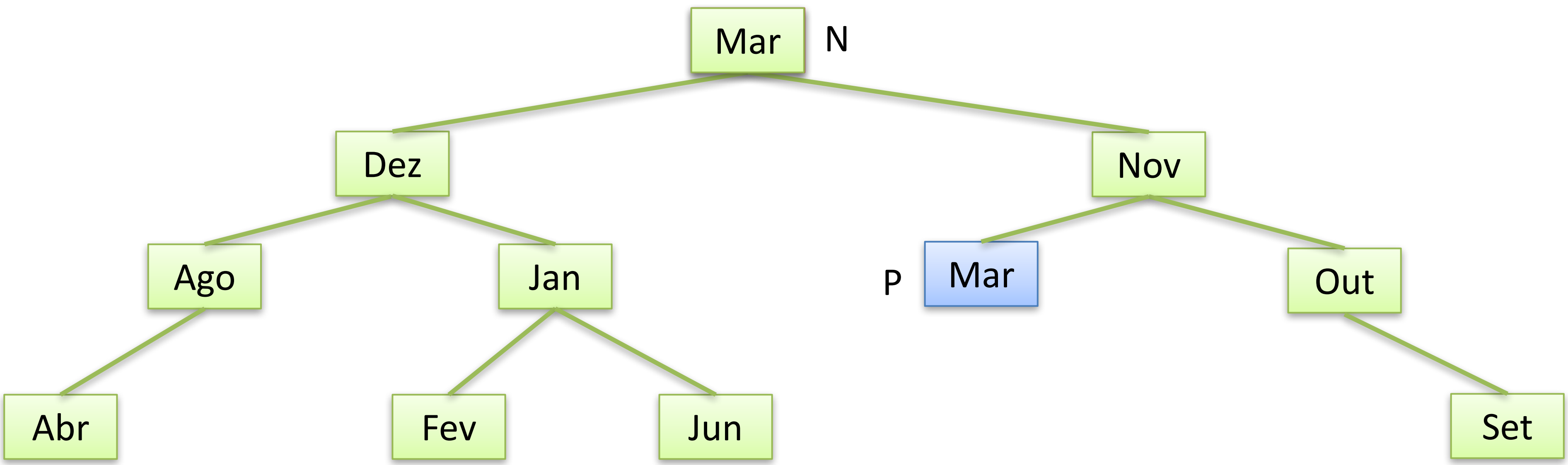


Remoção de Nó Com Duas Subárvores

- Seja:
  - N: O nó a ser excluído;
  - P: O nó com valor imediatamente maior que valor de N;

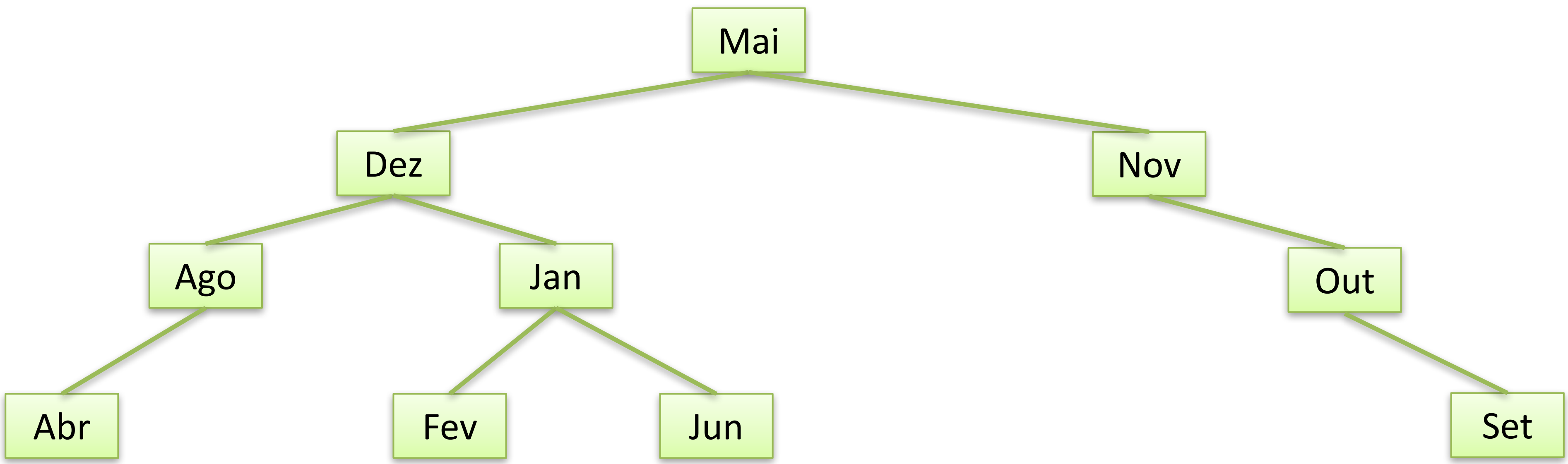
a. Substituir valor de N pelo valor de P;

b. Remover P.



Remoção de Nó Com Duas Subárvores

- A raiz da subárvore passa a ocupar a área do nó:

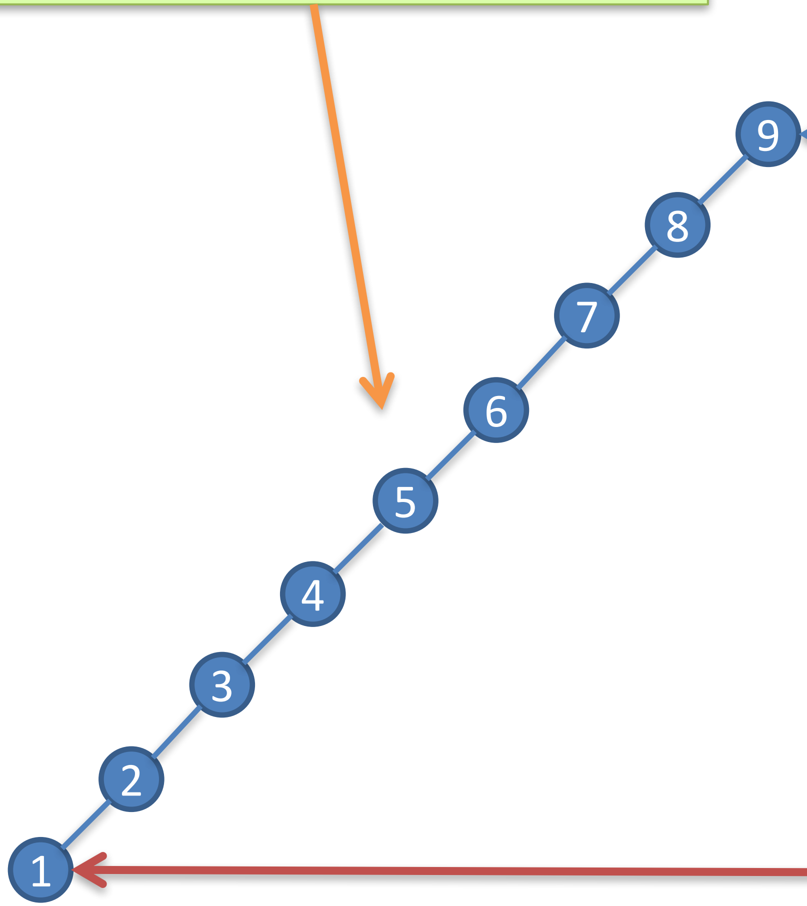


Complexidade da Pesquisa

- Dada uma ABP com ***n*** chaves e dada uma determinada chave ***i*** do conjunto das n chaves, quantas comparações são necessárias para se localizar a chave ***i*** na ABP?
- Seja h, a altura de uma ABP.

Complexidade da Pesquisa

9,8,7,6,5,4,3,2,1.

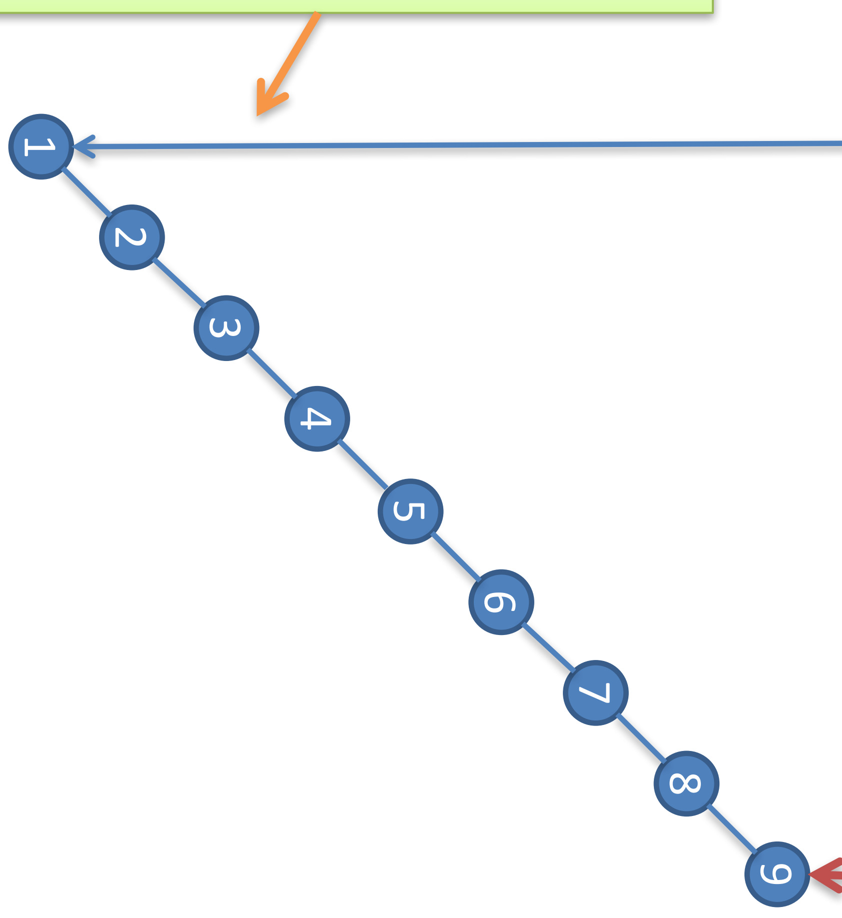


Melhor Caso: encontrar 9 é O(1).

Pior Caso: encontrar 1 é O(h).

Complexidade da Pesquisa

1,2,3,4,5,6,7,8,9.

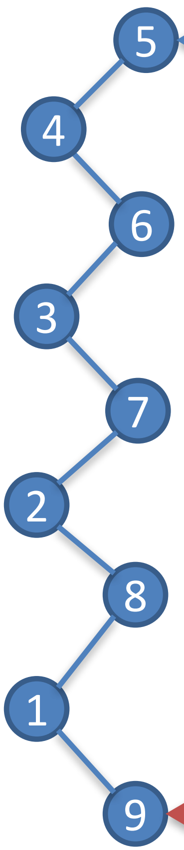


Melhor Caso: encontrar 1 é O(1).

Pior Caso: encontrar 9 é O(h).

Complexidade da Pesquisa

5,4,6,3,7,2,8,1,9.

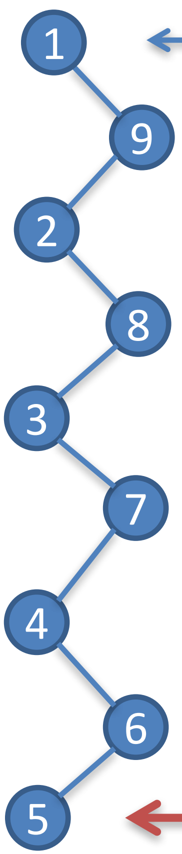


Melhor Caso: encontrar 5 é O(1).

Pior Caso: encontrar 9. é O(h).

Complexidade da Pesquisa

1,9,2,8,3,7,4,6,5.



Melhor Caso: encontrar 1 é O(1).

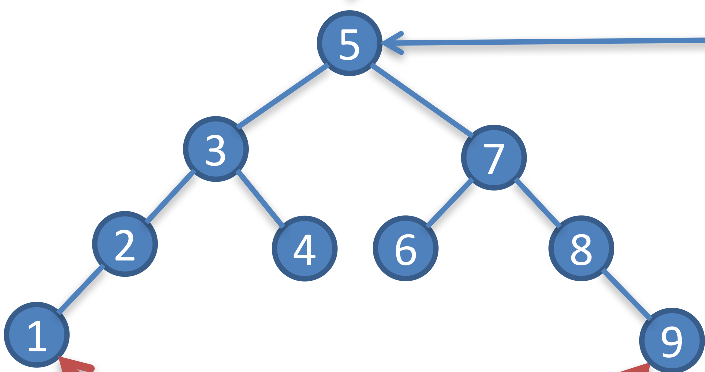
Pior Caso: encontrar 5. é O(h).

Complexidade da Pesquisa

- ABP que possuem estruturas tal como uma lista (h=n) são ditas ABP degeneradas.
- Tal como em uma lista, o melhor caso é O(1), o pior caso de uma pesquisa é O(n) (pois h=n) e o caso médio é O(n/2).

Complexidade da Pesquisa

5,7,3,6,8,2,1,9,4.



Melhor Caso: encontrar 5 é O(1).

Árvore Balanceada: As alturas das duas subárvores a partir de cada nó difere no máximo em uma unidade

Pior Caso: encontrar 1 ou 9. é ≈ $O(\log_2 N)$.

- Dada uma ABP com ***n*** chaves e dada uma determinada chave ***i*** do conjunto das n chaves, quantas comparações são necessárias para se localizar a chave ***i*** na ABP?
- Isso depende da estrutura da árvore, que depende por sua vez da ordem em que as n chaves foram inseridas.