UFMG UNIVERSIDADE FEDERAL DE MINAS GERAIS

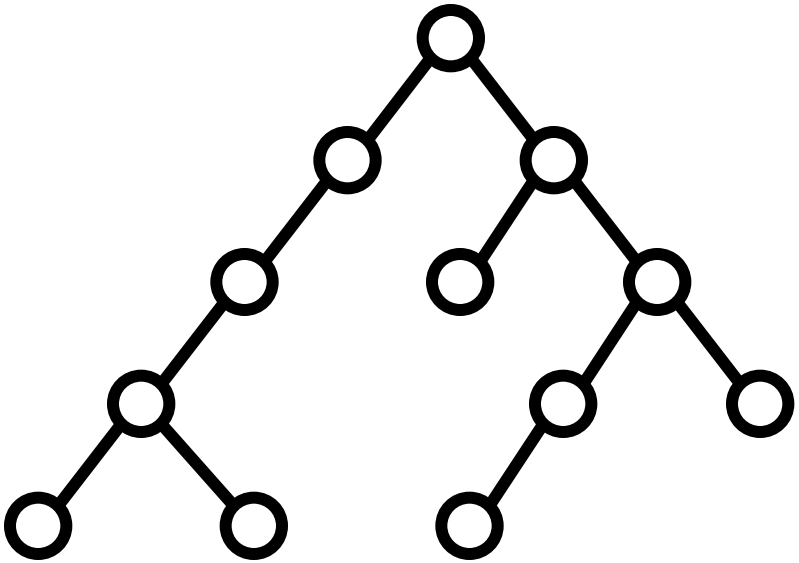
Árvores

Gisele Pappa
glpappa@dcc.ufmg.br
Raquel Minardi
raquelcm@dcc.ufmg.br

Árvores

UFMG COMPUTER SCIENCE

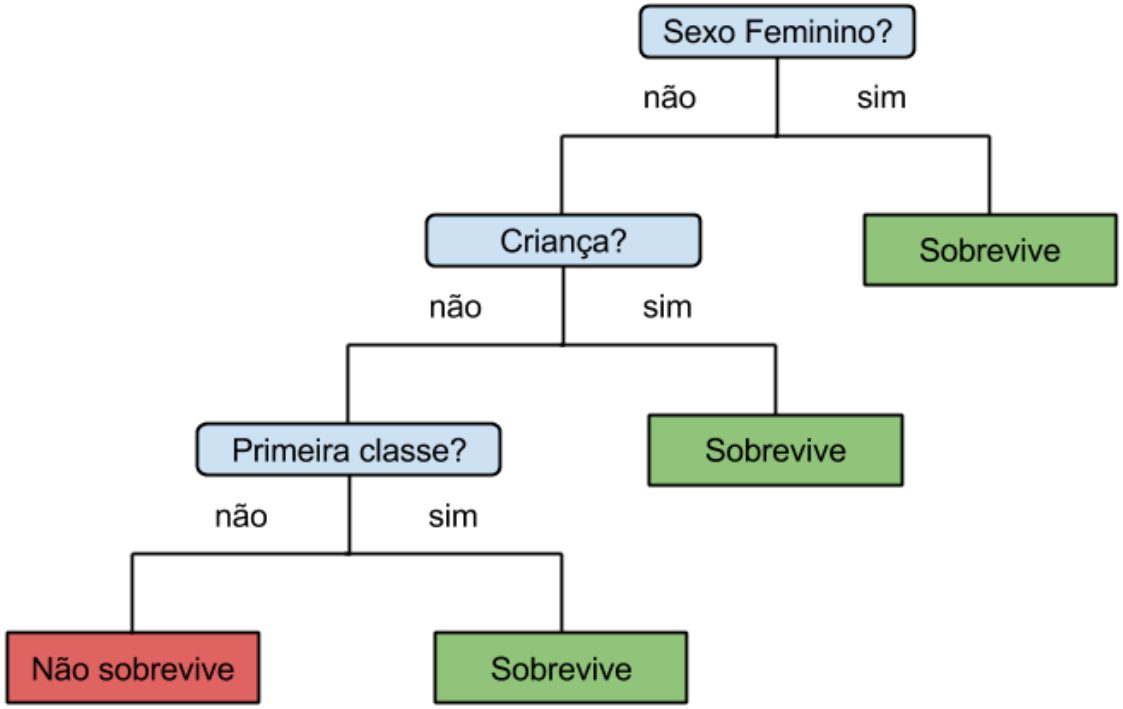
- Organizam dados de forma hierárquica
  - Acontecem com frequência na natureza
- Fáceis de representar e manipular com computadores
  - Úteis para várias tarefas



Exemplos

UFMG COMPUTER SCIENCE

- Árvores de decisão (inteligência artificial)
  - "Dado um passageiro do Titanic, qual a probabilidade de ele ter sobrevivido?"



Exemplos

UFMG COMPUTER SCIENCE

- Árvore de Huffman (compressão de dados)
  - "this is an example of a huffman tree"

	#	ASCII			
		dec	bin	bits	total
' '	7	32	00100000	8	56
a	4	97	01100001	8	32
e	4	101	01100101	8	32
f	3	102	01100110	8	24
h	2	104	01101000	8	16
i	2	105	01101001	8	16
m	2	109	01101101	8	16
n	2	110	01101110	8	16
s	2	115	01110011	8	16
t	2	116	01110100	8	16
l	1	108	01101100	8	8
o	1	111	01101111	8	8
p	1	112	01110000	8	8
r	1	114	01110010	8	8
u	1	117	01110101	8	8
x	1	120	01111000	8	8

Exemplos

UFMG COMPUTER SCIENCE

- Árvore de Huffman (compressão de dados)
  - "this is an example of a huffman tree"

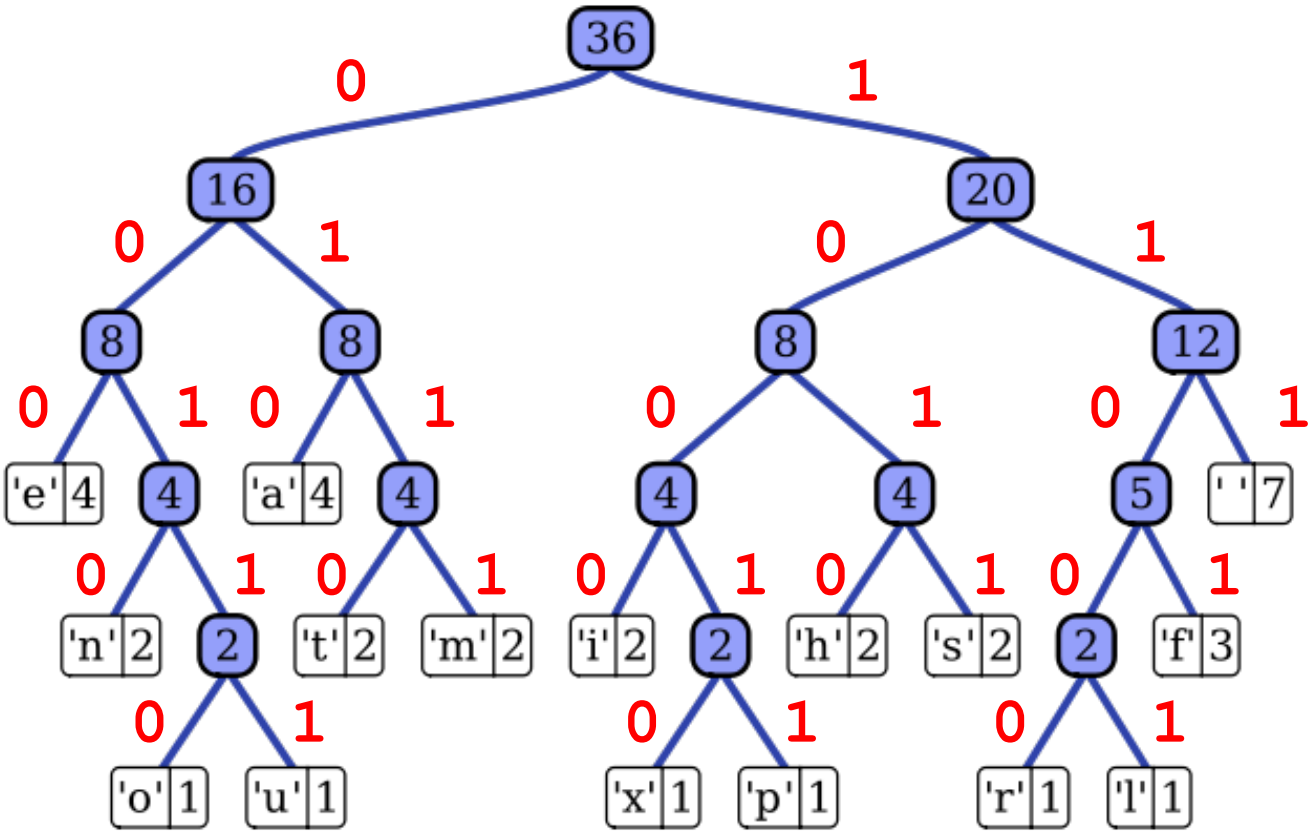
	#	ASCII			
		dec	bin	bits	total
' '	7	32	00100000	8	56
a	4	97	01100001	8	32
e	4	101	01100101	8	32
f	3	102	01100110	8	24
h	2	104	01101000	8	16
i	2	105	01101001	8	16
m	2	109	01101101	8	16
n	2	110	01101110	8	16
s	2	115	01110011	8	16
t	2	116	01110100	8	16
l	1	108	01101100	8	8
o	1	111	01101111	8	8
p	1	112	01110000	8	8
r	1	114			8
u	1	117			8
x	1	120			8

288 bits

Exemplos

UFMG COMPUTER SCIENCE

- Árvore de Huffman (compressão de dados)
  - “this is an example of a huffman tree”



Exemplos:

a = 010

x = 10010

Exemplos

UFMG COMPUTER SCIENCE

- Árvore de Huffman (compressão de dados)
  - "this is an example of a huffman tree"

	#	ASCII				HUFFMAN		
		dec	bin	bits	total	bin	bits	total
' '	7	32	00100000	8	56	111	3	21
a	4	97	01100001	8	32	010	3	12
e	4	101	01100101	8	32	000	3	12
f	3	102	01100110	8	24	1101	4	12
h	2	104	01101000	8	16	1010	4	8
i	2	105	01101001	8	16	1000	4	8
m	2	109	01101101	8	16	0111	4	8
n	2	110	01101110	8	16	0010	4	8
s	2	115	01110011	8	16	1011	4	8
t	2	116	01110100	8	16	0110	4	8
l	1	108	01101100	8	8	11001	5	5
o	1	111	01101111	8	8	00110	5	5
p	1	112	01110000	8	8	10011	5	5
r	1	114			8	11000	5	5
u	1	117			8	00111	5	5
x	1	120			8	10010	5	5

288 bits

Exemplos

UFMG COMPUTER SCIENCE

- Árvore de Huffman (compressão de dados)
  - "this is an example of a huffman tree"

	#	ASCII				HUFFMAN		
		dec	bin	bits	total	bin	bits	total
' '	7	32	00100000	8	56	111	3	21
a	4	97	01100001	8	32	010	3	12
e	4	101	01100101	8	32	000	3	12
f	3	102	01100110	8	24	1101	4	12
h	2	104	01101000	8	16	1010	4	8
i	2	105	01101001	8	16	1000	4	8
m	2	109	01101101	8	16	0111	4	8
n	2	110	01101110	8	16	0010	4	8
s	2	115	01110011	8	16	1011	4	8
t	2	116	01110100	8	16	0110	4	8
l	1	108	01101100	8	8	11001	5	5
o	1	111	01101111	8	8	00110	5	5
p	1	112	01110000	8	8	10011	5	5
r	1	114			8			5
u	1	117			8			5
x	1	120			8			5

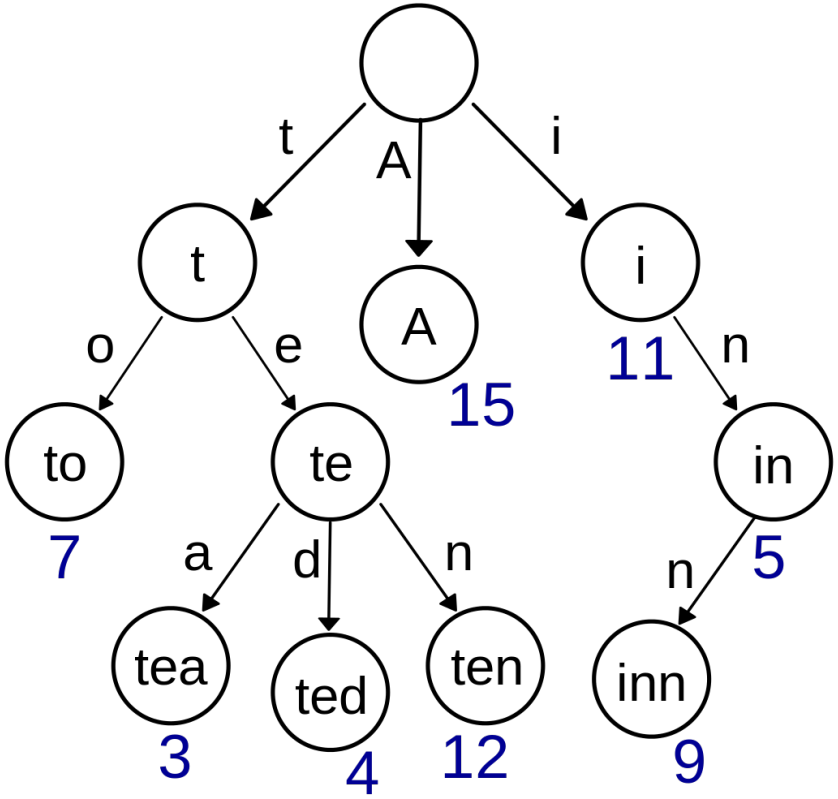
288 bits

135 bits

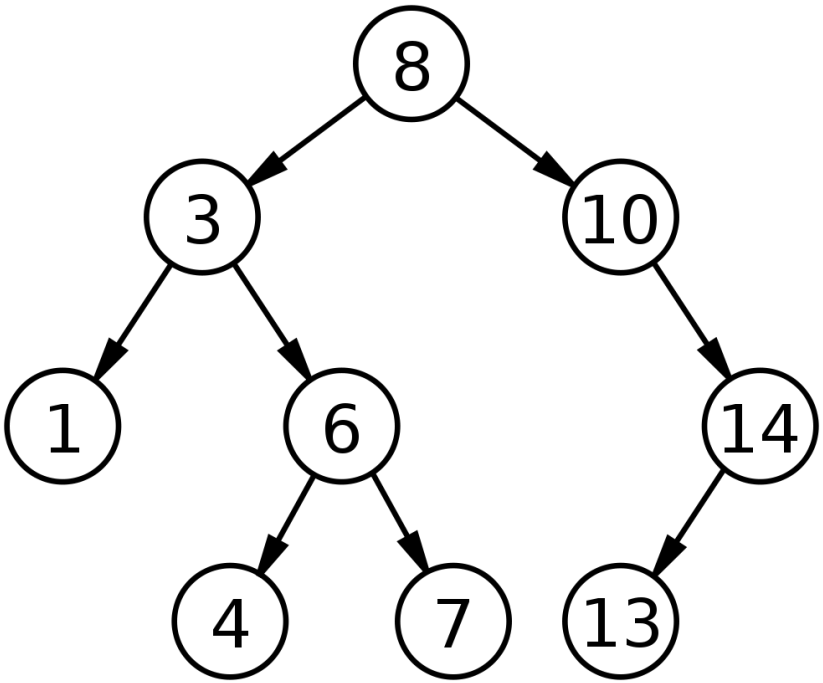
Árvores em AEDS2

UFMG COMPUTER SCIENCE

Árvores digitais (tries)

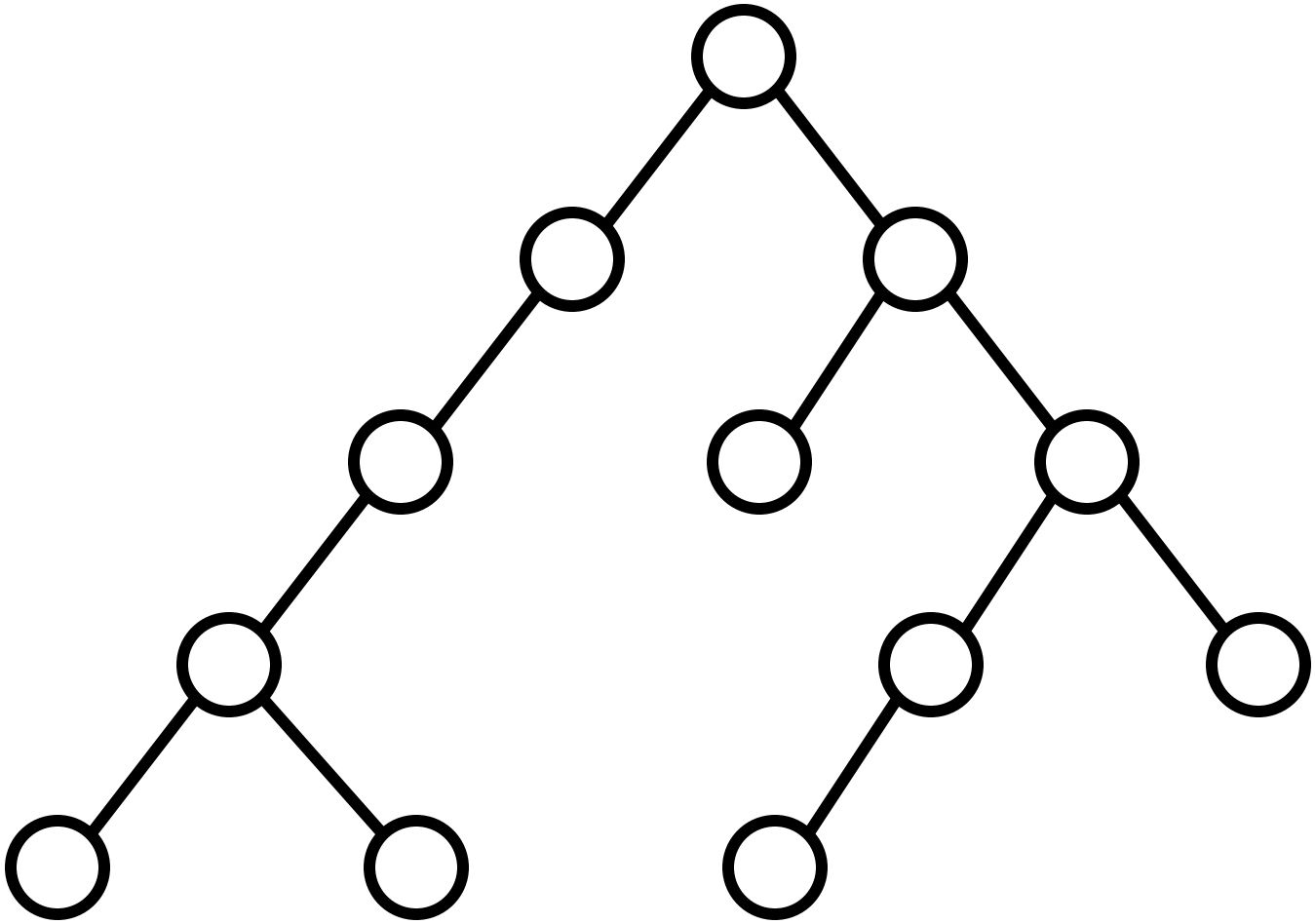


Árvore binária de busca (ABB)



Terminologia

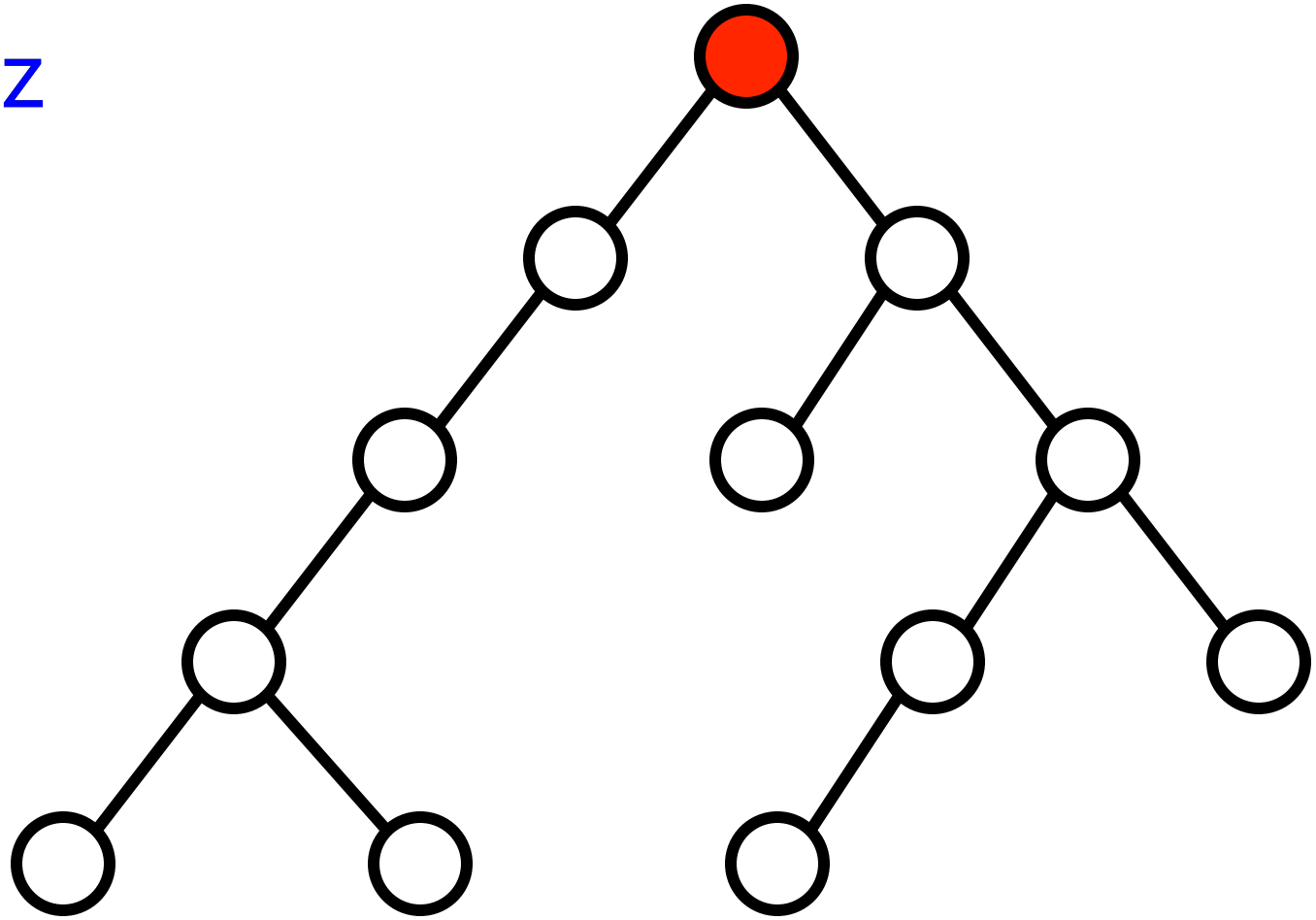
UFMG
COMPUTER
SCIENCE



Terminologia

UFMG COMPUTER SCIENCE

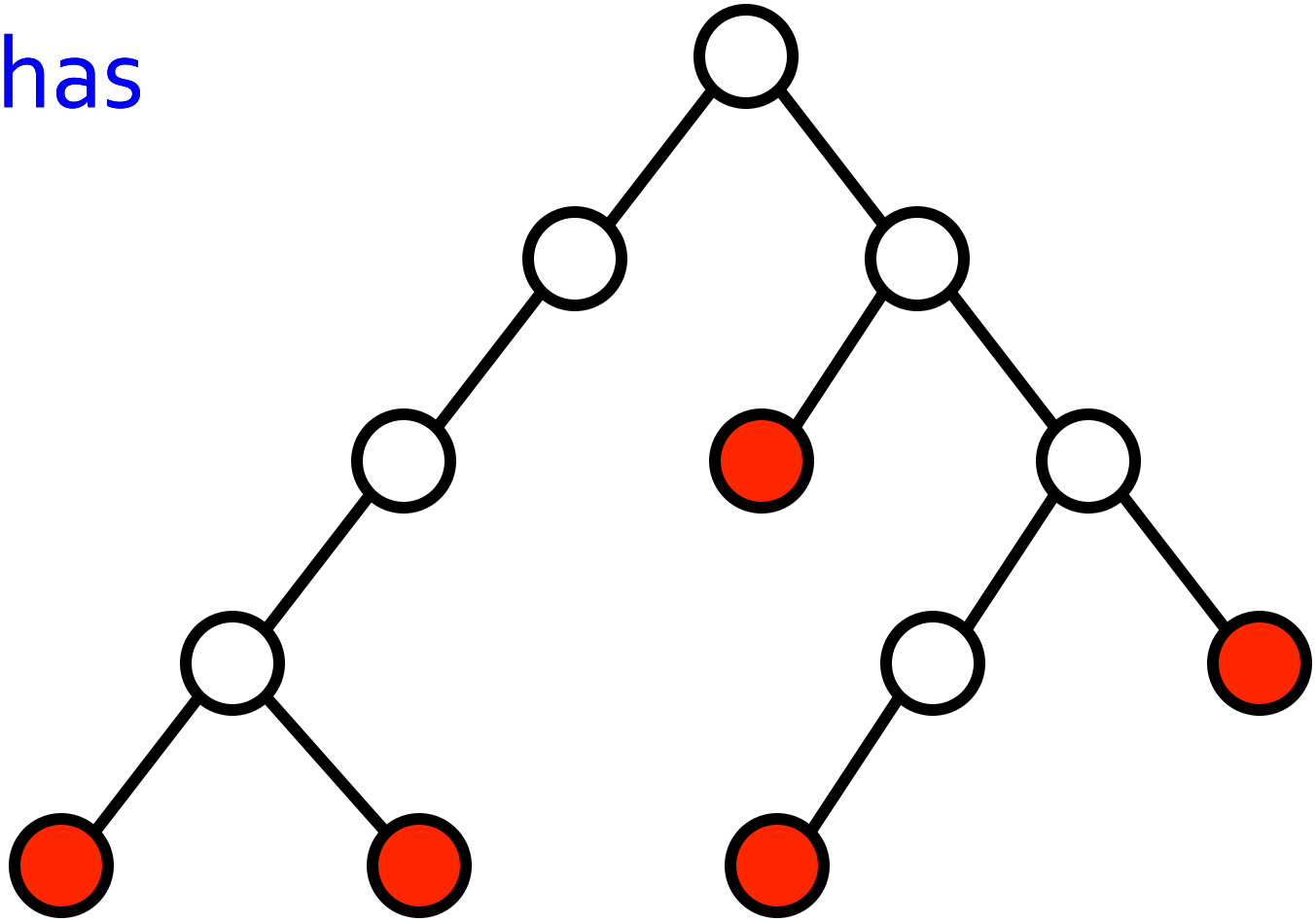
Raiz



Terminologia

UFMG COMPUTER SCIENCE

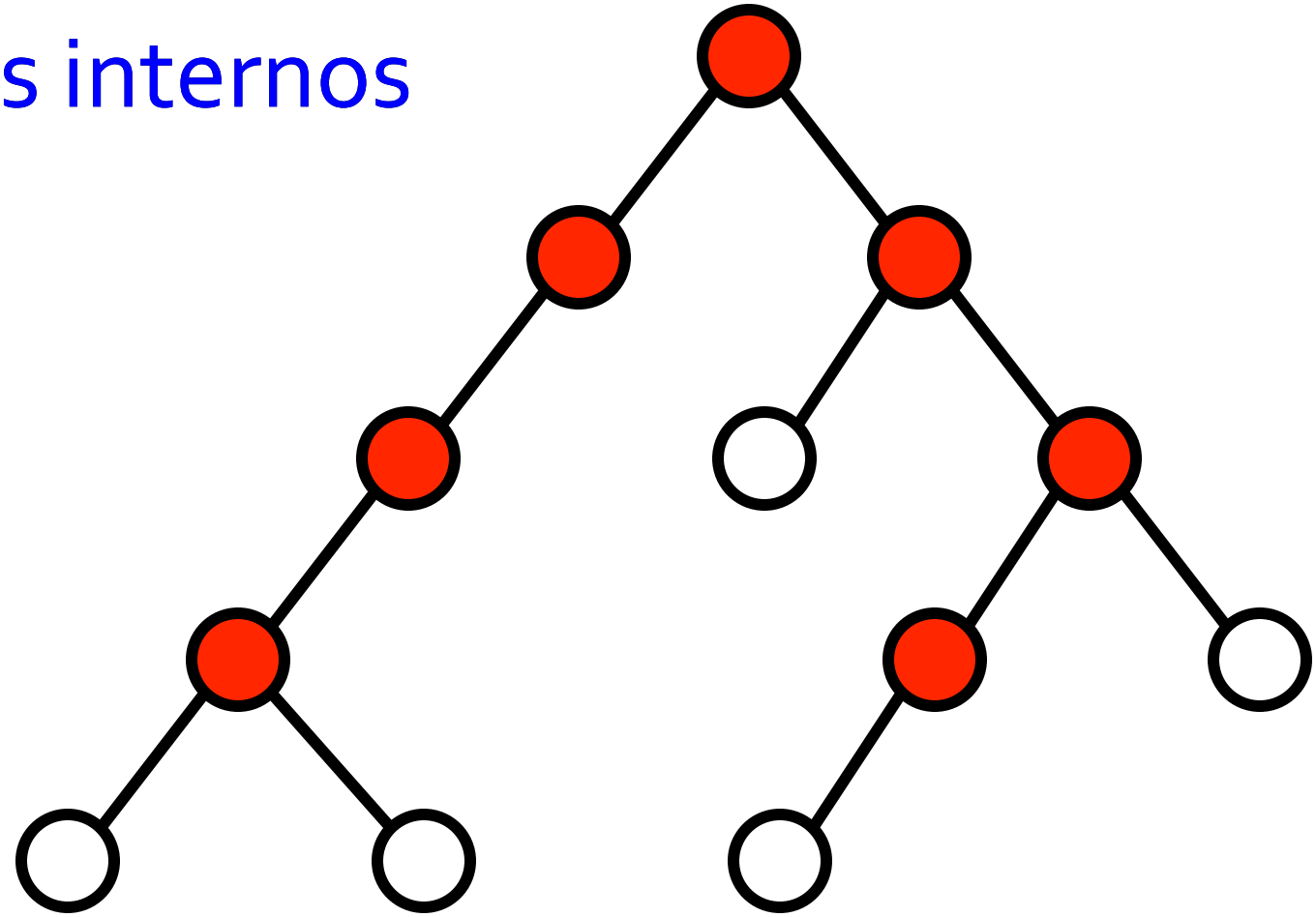
Folhas



Terminologia

UFMG COMPUTER SCIENCE

Nós internos

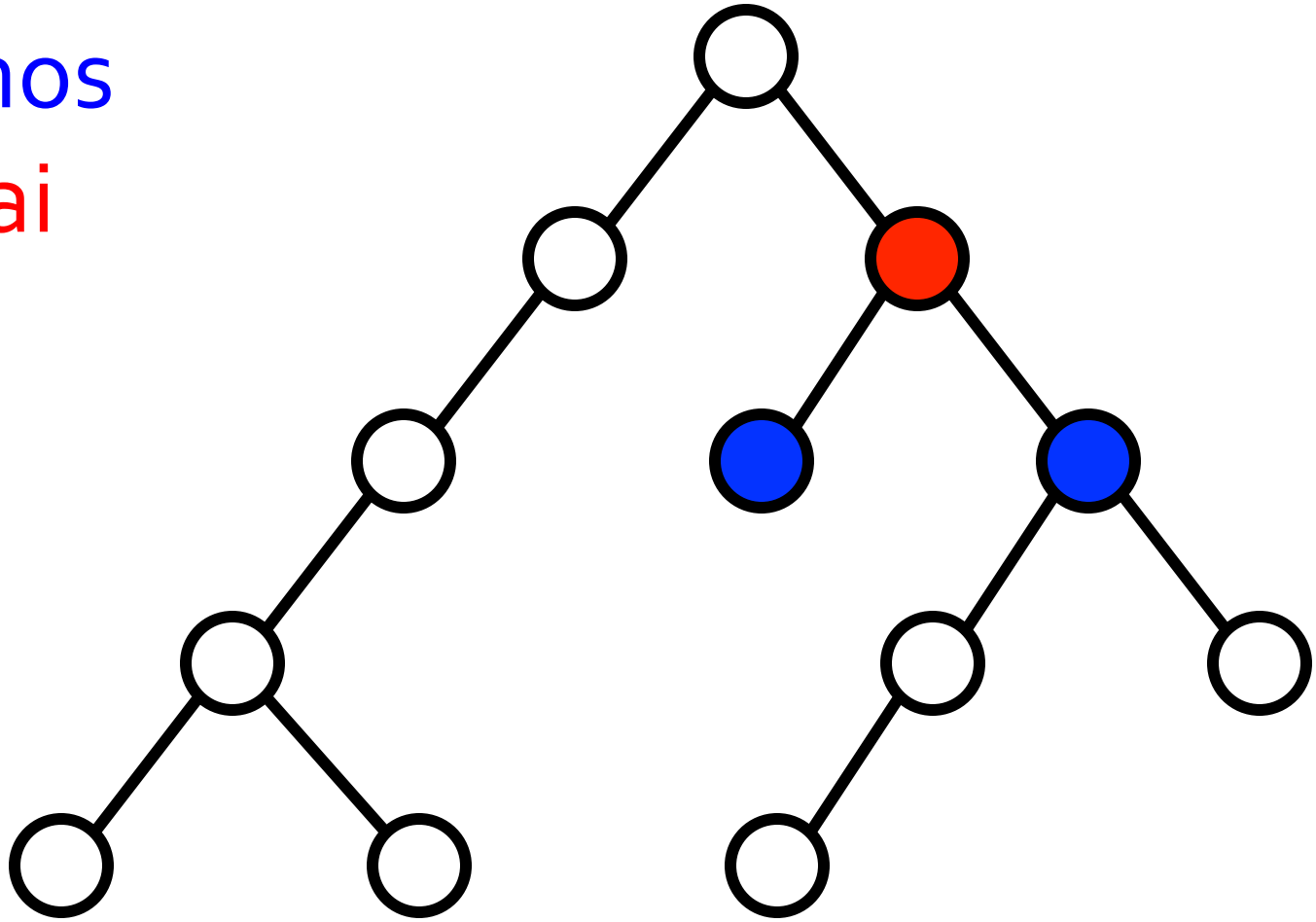


Terminologia

UFMG COMPUTER SCIENCE

Filhos

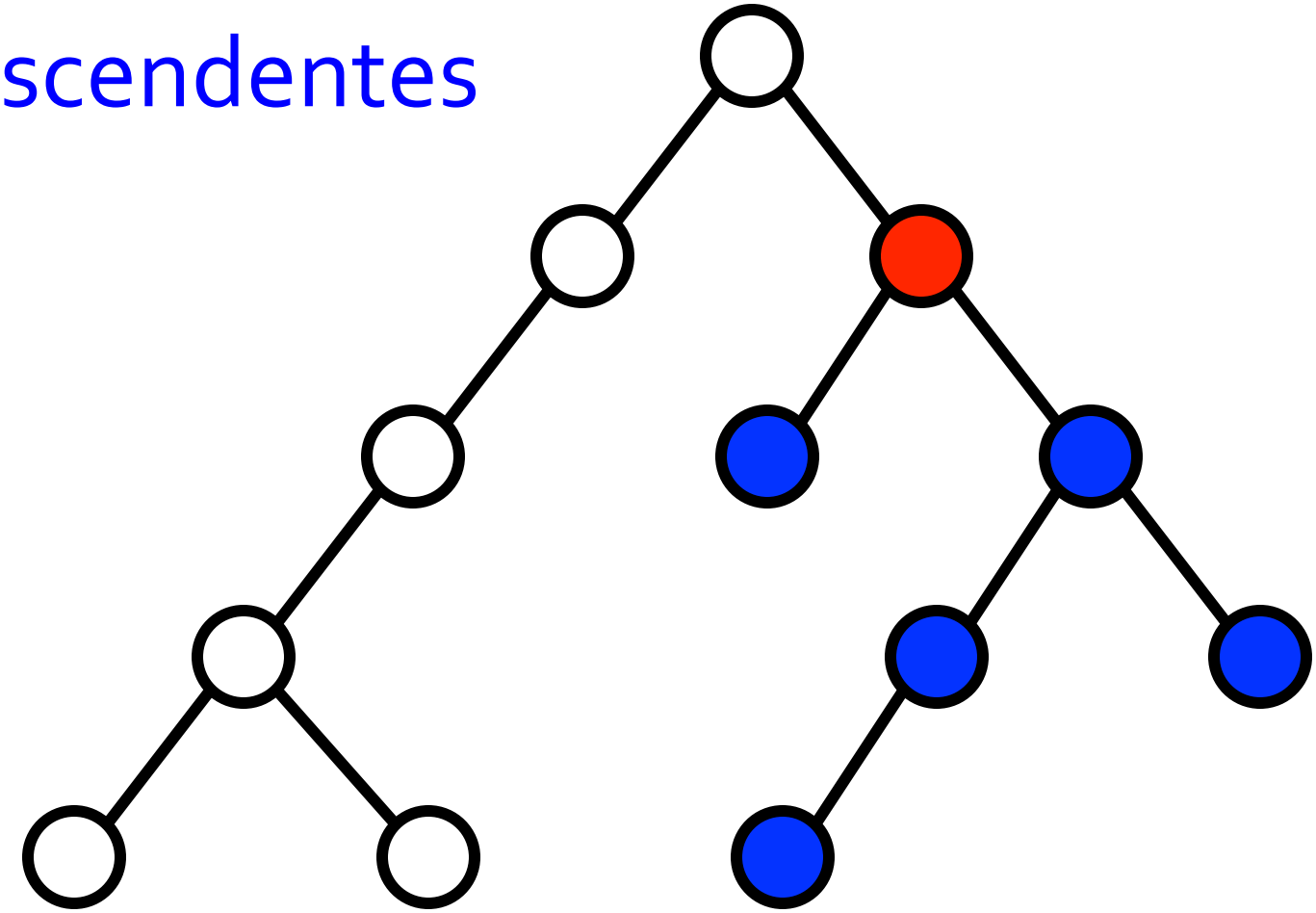
Pai



Terminologia

UFMG COMPUTER SCIENCE

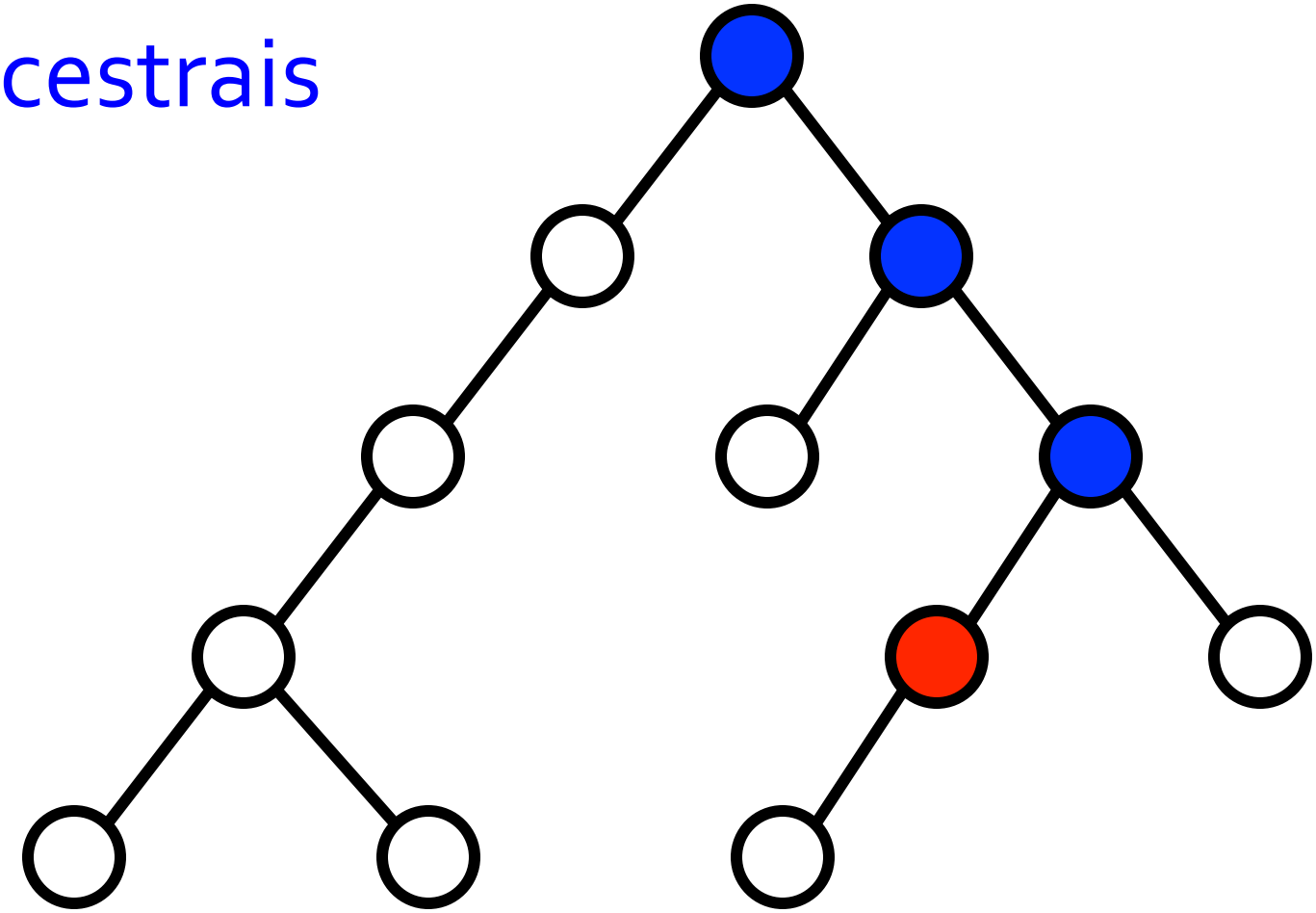
Descendentes



Terminologia

UFMG COMPUTER SCIENCE

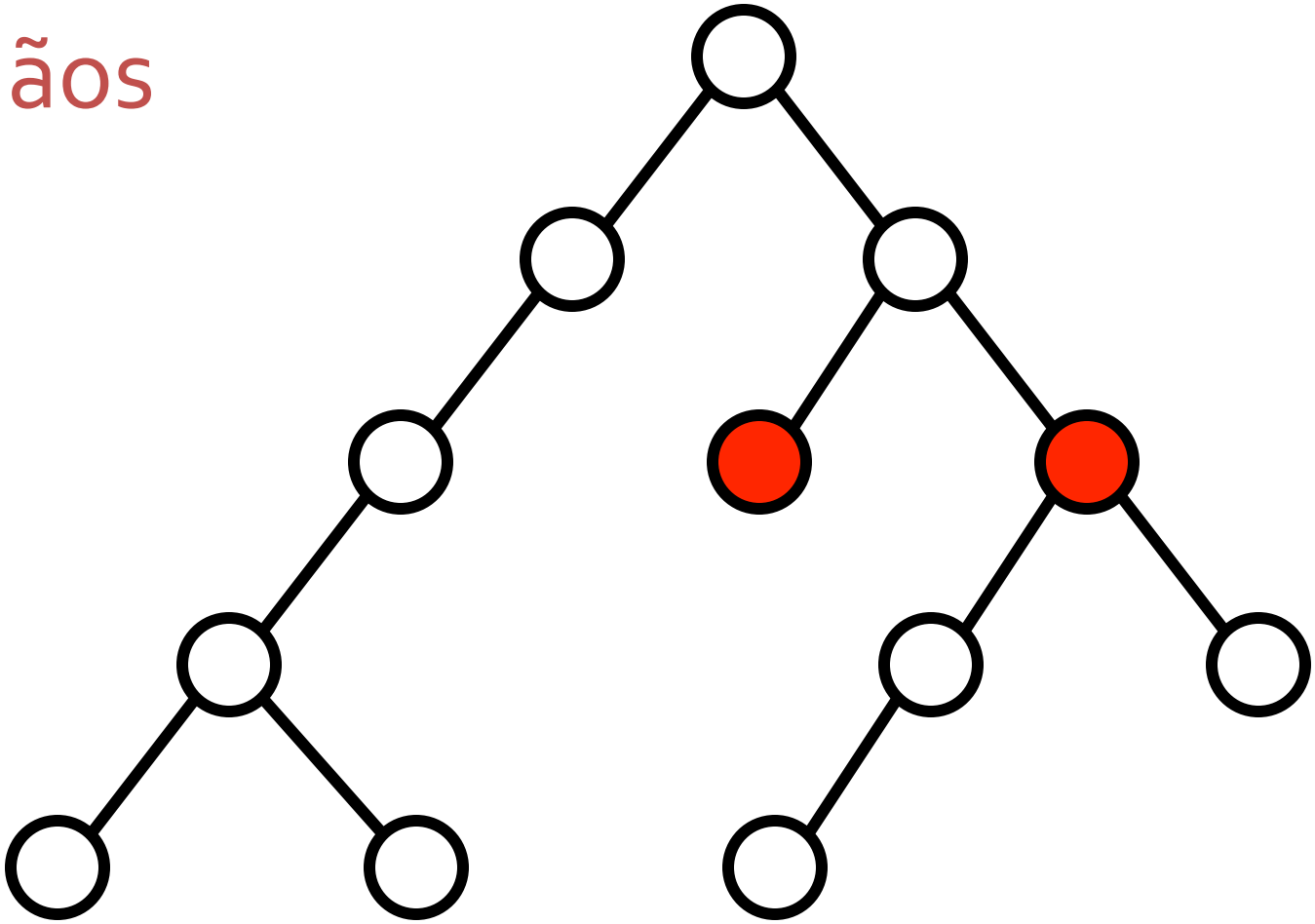
Ancestrais



Terminologia

UFMG COMPUTER SCIENCE

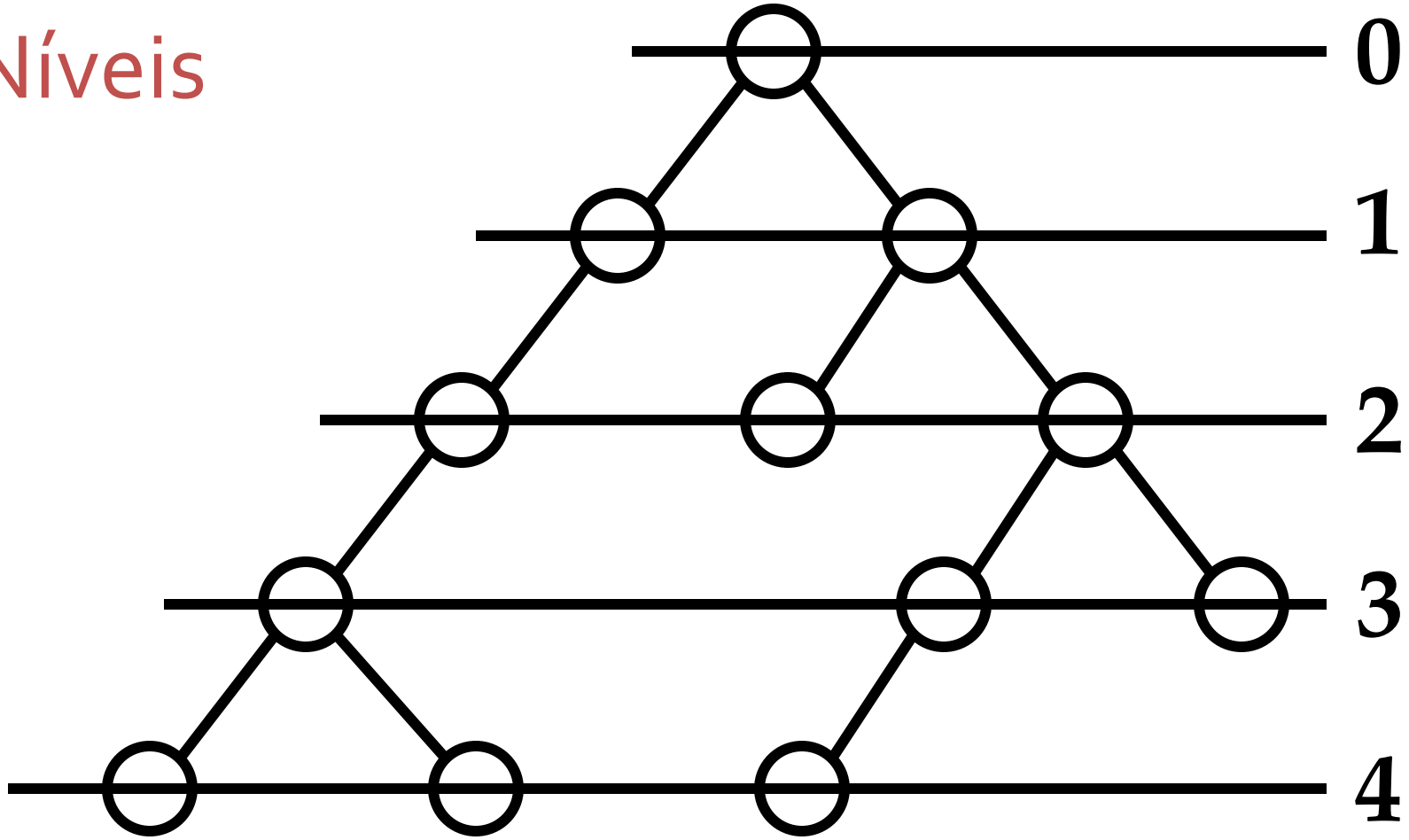
Irmãos



Terminologia

UFMG COMPUTER SCIENCE

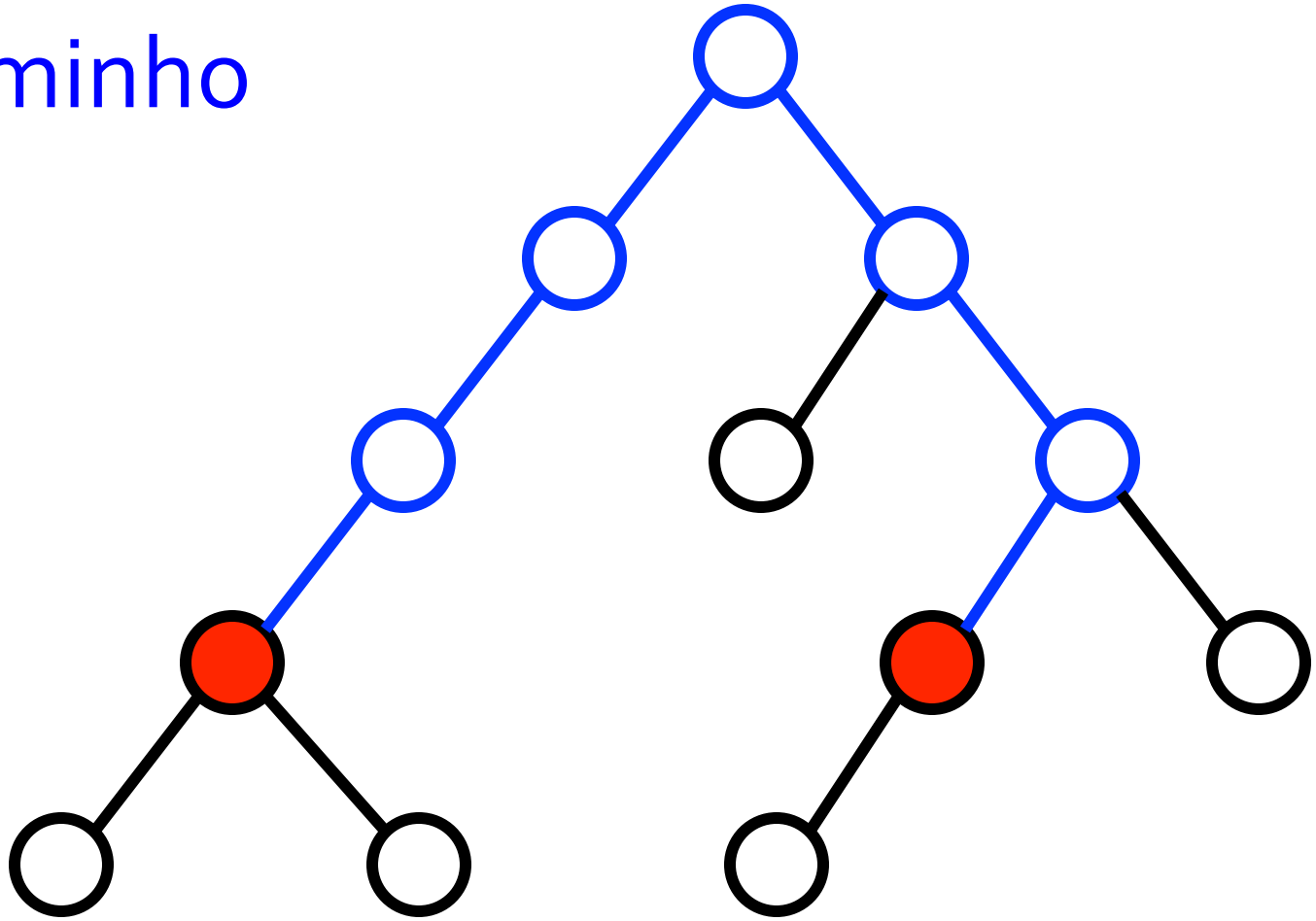
Níveis



Terminologia

UFMG COMPUTER SCIENCE

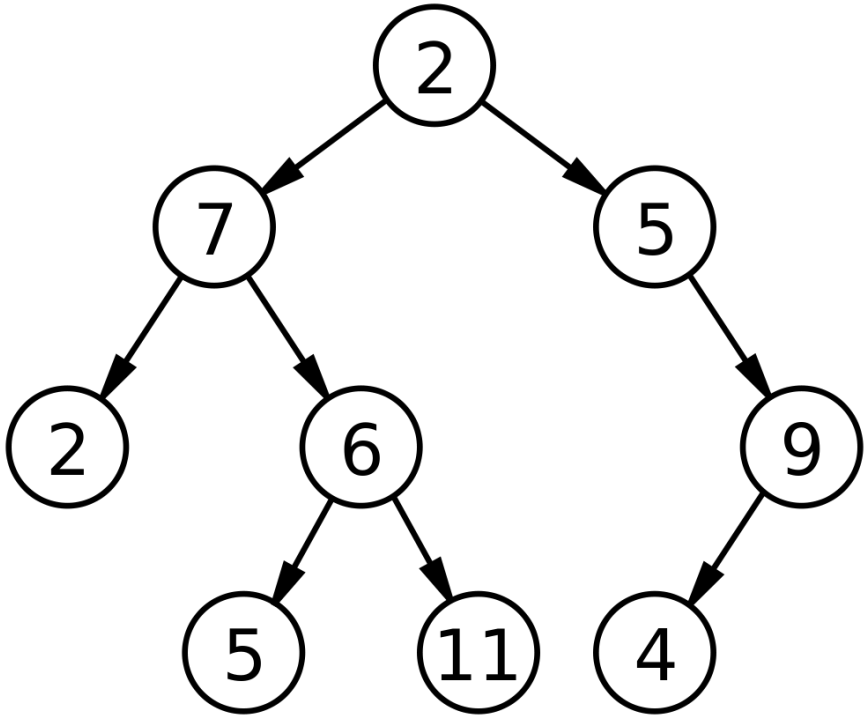
Caminho



TAD árvore binária

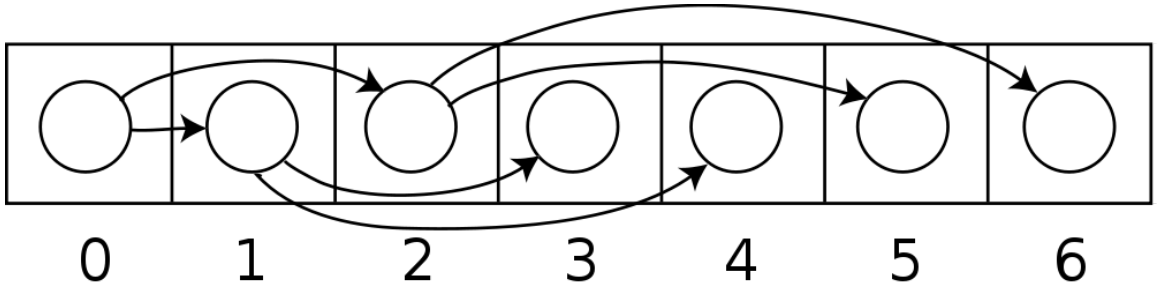
UFMG COMPUTER SCIENCE

- Cada nó tem ***no máximo*** dois filhos
  - (obs: a árvore ao lado ***não impõe*** nenhuma ordenação dos nodos)
- Operações
  - Inserção
  - Remoção
  - Caminhamento



Árvore binária: Impl. com Arranjos

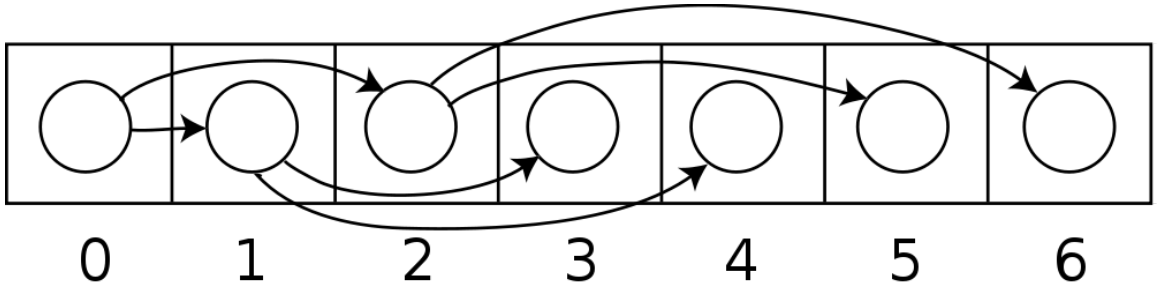
UFMG COMPUTER SCIENCE



- Filhos de *i*
  - Esquerda: $2i + 1$
  - Direita: $2i + 2$
- Pai de *i*: $\left\lfloor \frac{i-1}{2} \right\rfloor$

Árvore binária: Impl. com Arranjos

UFMG COMPUTER SCIENCE



- Vantagens
  - Representação compacta
  - Localidade de referência
- Desvantagens
  - Desperdício de espaço em árvores incompletas
  - Inserção e remoção podem exigir deslocamentos

Árvore binária:
Impl. com Apontadores

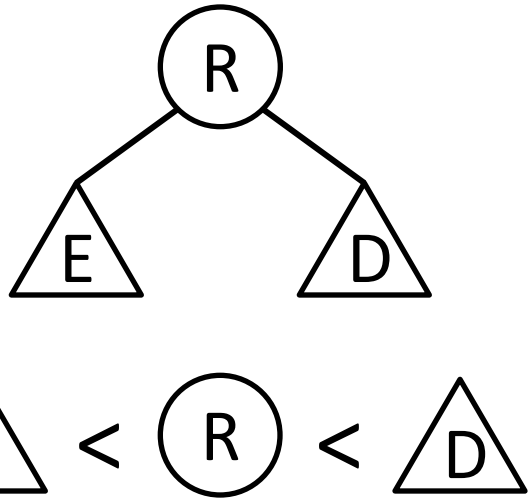
UFMG COMPUTER SCIENCE

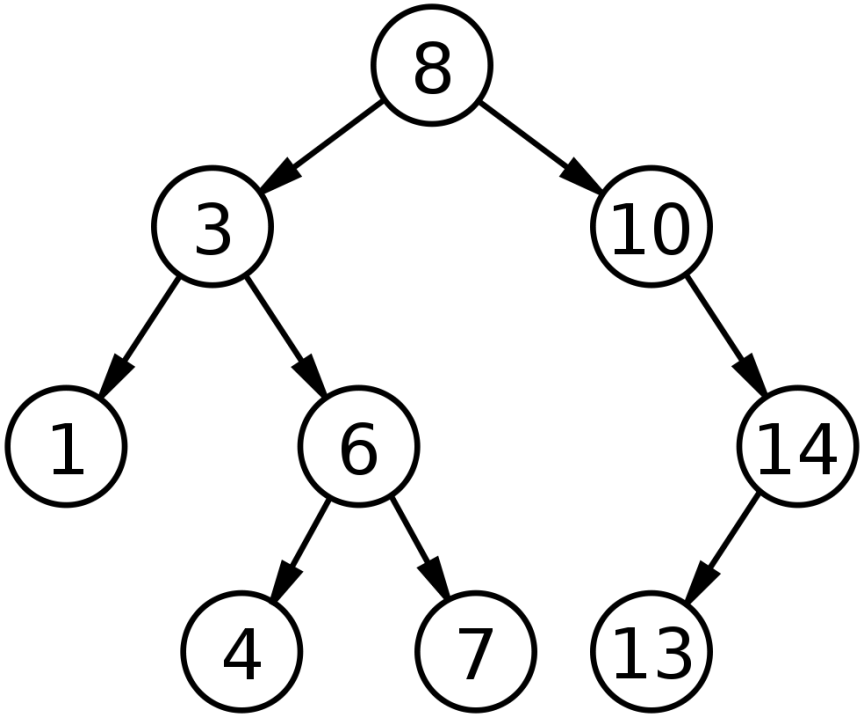
```
typedef int TChave;

typedef struct {
    TChave Chave;
    // outros componentes
} TItem;

typedef struct No {
    TItem Item;
    Apontador Esq, Dir;
} TNo;

typedef struct No * Apontador;
typedef Apontador TArvore;
```

TAD árvore binária de busca (ABB)

UFMG COMPUTER SCIENCE

- Cada nó tem ***no máximo*** dois filhos
  - (obs: a árvore ao lado ***impõe*** uma ordenação particular aos nodos)





TAD árvore binária de busca (ABB)

UFMG COMPUTER SCIENCE

- Cada nó tem ***no máximo*** dois filhos
  - (obs: a árvore ao lado ***impõe*** uma ordenação particular aos nodos)
- Operações
  - Inserção
  - Remoção
  - Caminhamento
  - ***Pesquisa***
  - ***Ordenação***

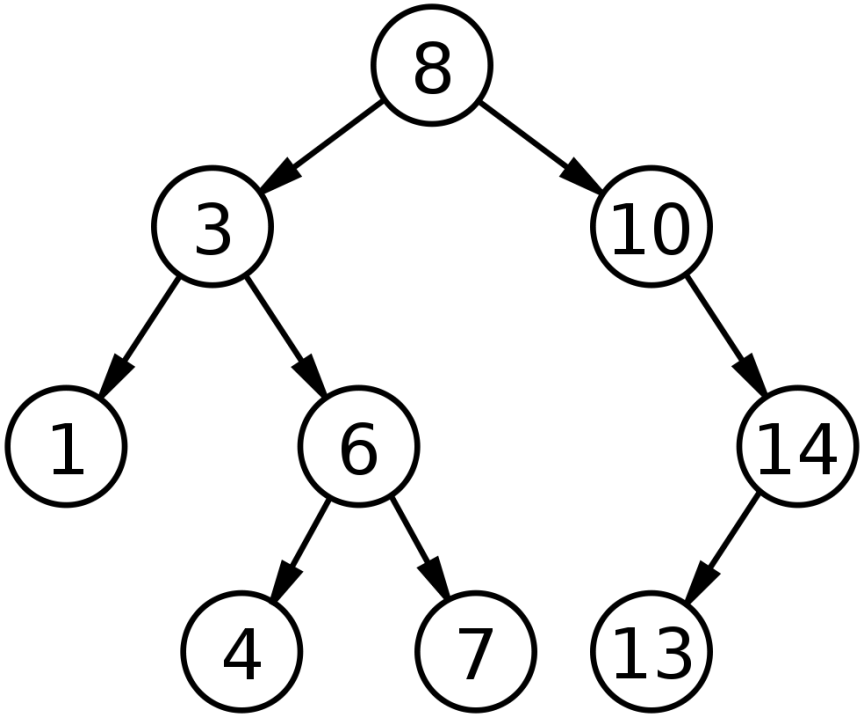


ABB: inserção

UFMG COMPUTER SCIENCE

- **`void Insere(TItem x, Apontador *p)`**
  - **Observação:** **`p`** passado por referência (endereço contido em **`*p`** vai ser alterado dentro de **`Insere`**)
  - **Ideia:** a partir do nodo apontado por **`*p`**, podemos encontrar o ponto de inserção para o registro **`x`**
    - **`*p`** é nulo?
      - Ponto de inserção!
    - **`x`** < registro atual?
      - Caminhamos para a esquerda
    - **`x`** > registro atual?
      - Caminhamos para a direita
    - **`x`** == registro atual?
      - Nodo já existe

ABB: inserção

UFMG COMPUTER SCIENCE

```
TArvore a; // Apontador
TItem x;
x.Chave = 8;
Insere(x, &a);
```

ABB: inserção

UFMG COMPUTER SCIENCE

```
Tarvore p; // Apontador
TItem x;
x.Chave = 8;
Insere(x, &p);
```

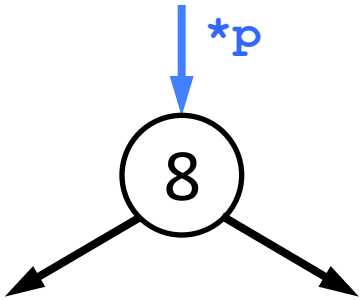


ABB: inserção

UFMG COMPUTER SCIENCE

```
Tarvore p; // Apontador
...        // várias inserções
TItem x;
x.Chave = 4;
Insere(x, &p);
```

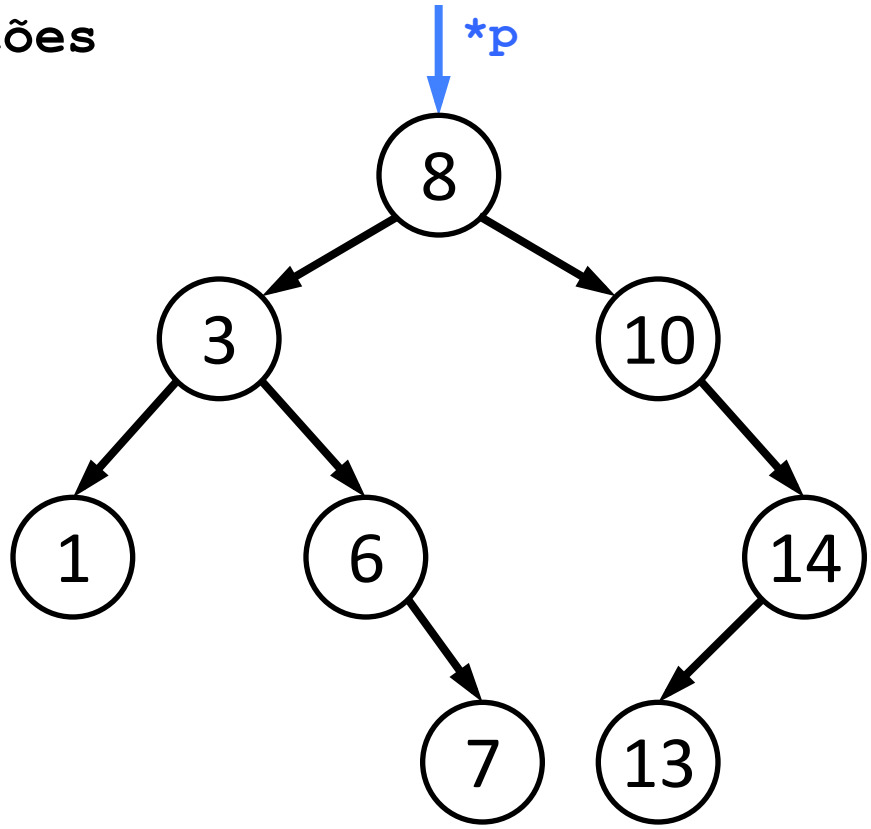


ABB: inserção

UFMG COMPUTER SCIENCE

```
Tarvore p; // Apontador
...        // várias inserções
TItem x;
x.Chave = 4;
Insere(x, &p);
```

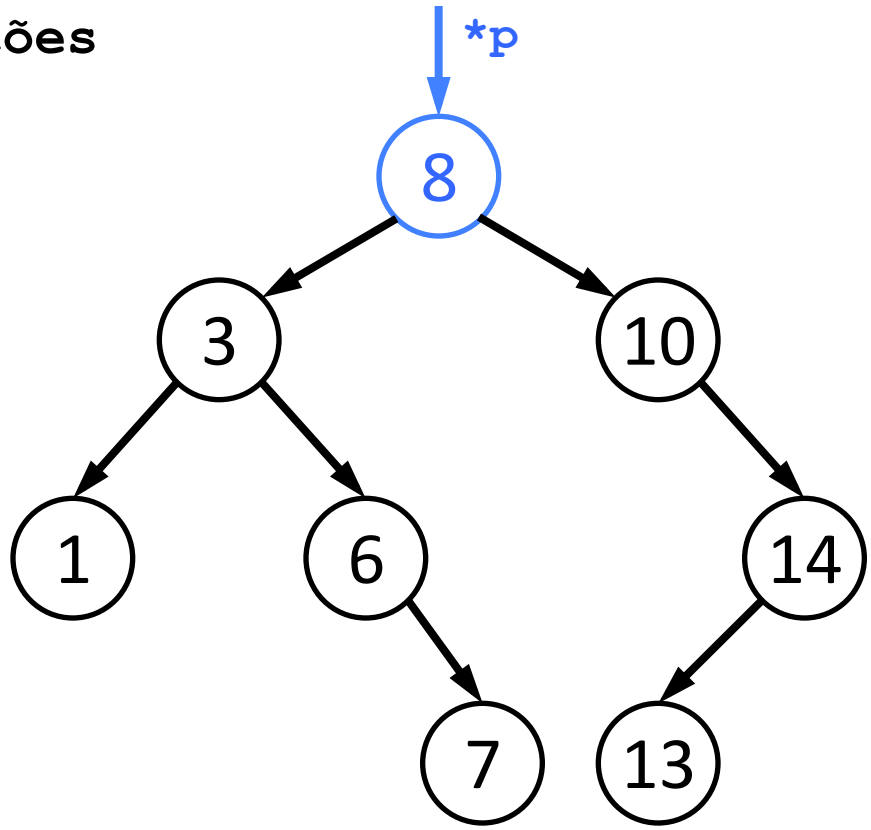


ABB: inserção

UFMG COMPUTER SCIENCE

```
Tarvore p; // Apontador
...        // várias inserções
TItem x;
x.Chave = 4;
Insere(x, &p);
```

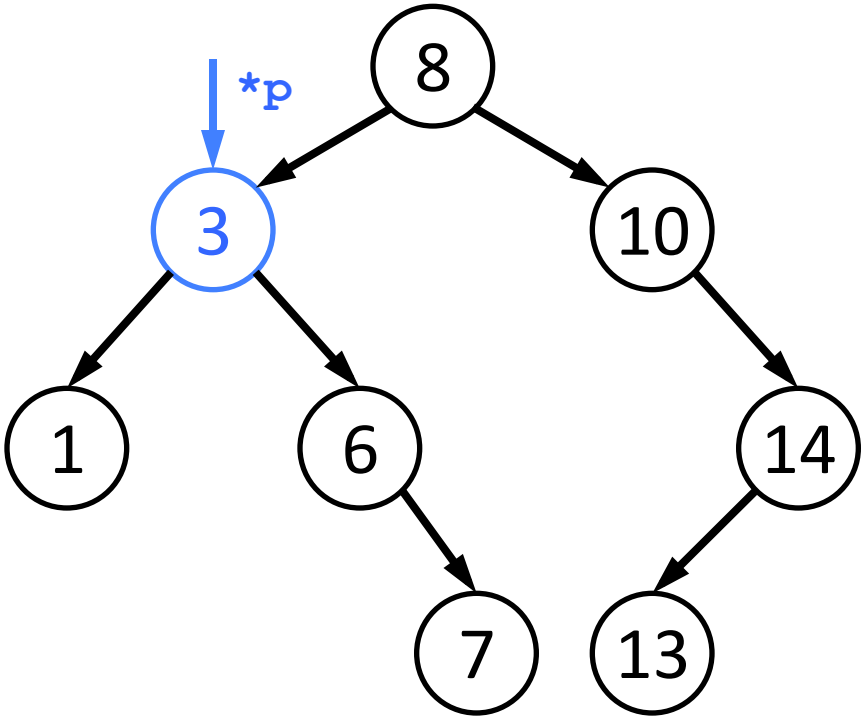


ABB: inserção

UFMG COMPUTER SCIENCE

```
Tarvore p; // Apontador
...        // várias inserções
TItem x;
x.Chave = 4;
Insere(x, &o);
```

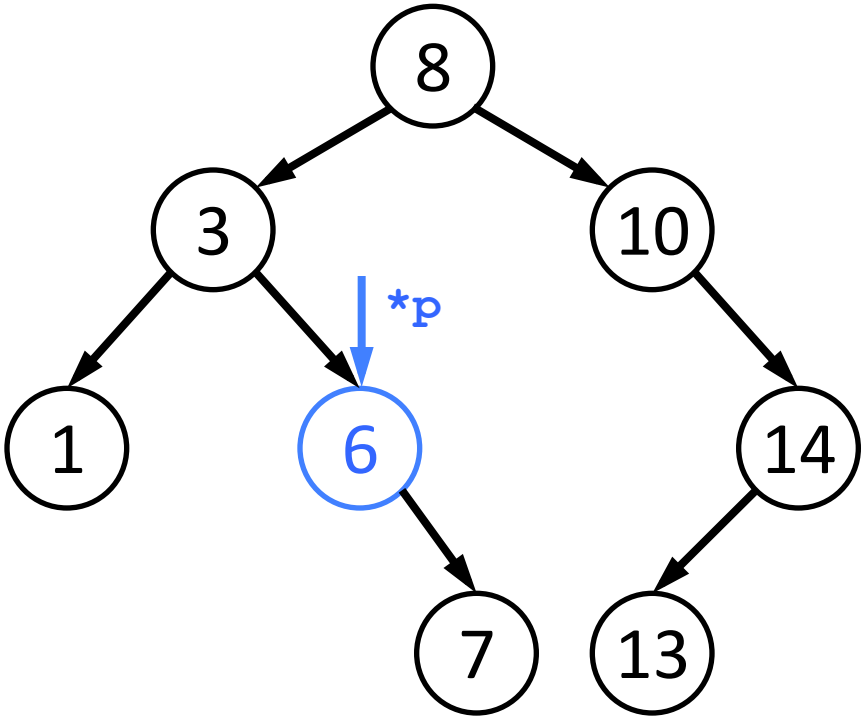


ABB: inserção

UFMG COMPUTER SCIENCE

```
Tarvore p; // Apontador
...        // várias inserções
TItem x;
x.Chave = 4;
Insere(x, &p);
```

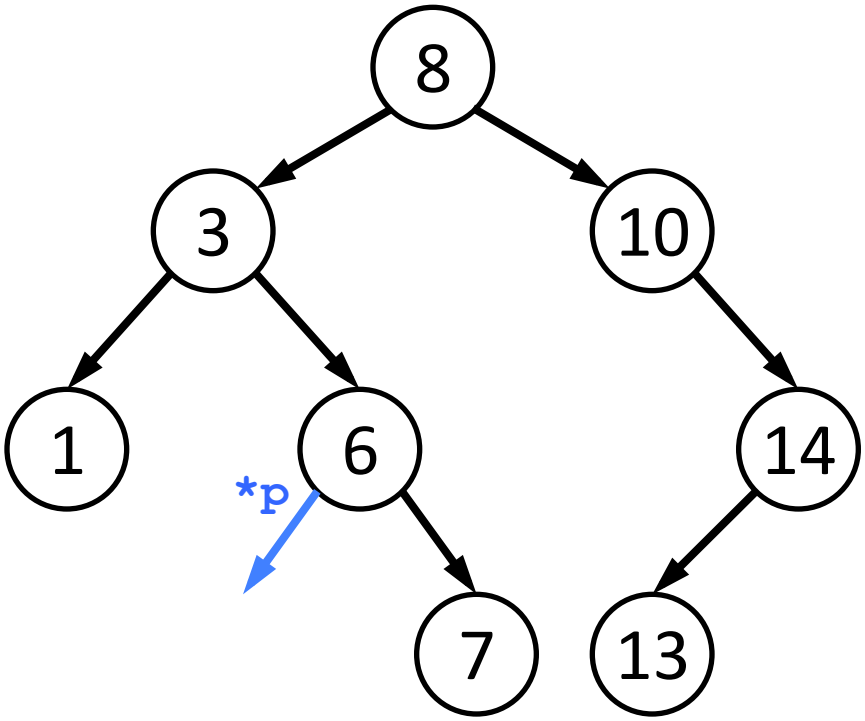


ABB: inserção

UFMG COMPUTER SCIENCE

```
Tarvore p; // Apontador
...        // várias inserções
TItem x;
x.Chave = 4;
Insere(x, &p);
```

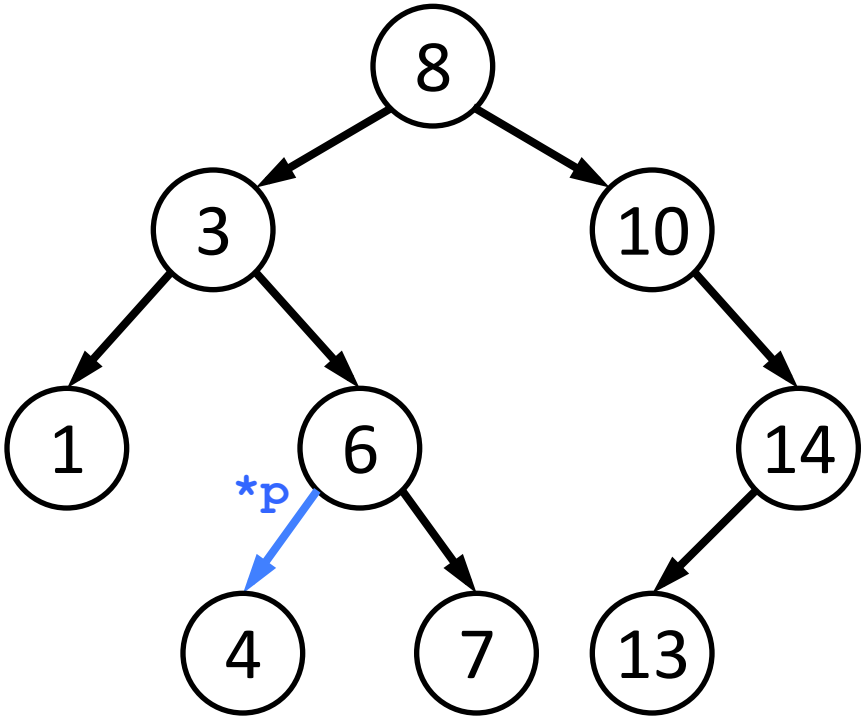


ABB: inserção

UFMG COMPUTER SCIENCE

```
void Insere(TItem x, Apontador *p) {
    if (*p == NULL) {
        *p = malloc(sizeof(TNo));
        (*p)->Item = x;
        (*p)->Esq = NULL;
        (*p)->Dir = NULL;
    }
    else if (x.Chave < (*p)->Item.Chave)
        Insere(x, &(*p)->Esq);
    else if (x.Chave > (*p)->Item.Chave)
        Insere(x, &(*p)->Dir);
    else
        printf("ERRO: Item já existe\n");
}
```

ABB: remoção

UFMG COMPUTER SCIENCE

- **`void Retira(TItem x, Apontador *p)`**
  - **Observação:** **`p`** passado por referência (endereço contido em **`*p`** vai ser alterado dentro de **`Retira`**)
  - **Ideia:** a partir do nodo apontado por **`*p`**, podemos encontrar o ponto de remoção para o registro **`x`**
    - **`*p`** é nulo?
      - Nada a fazer
    - **`x`** < registro atual?
      - Caminhamos para a esquerda
    - **`x`** > registro atual?
      - Caminhamos para a direita
    - **`x`** == registro atual?
      - Encontramos nodo a ser removido

ABB: remoção

UFMG COMPUTER SCIENCE

- Como remover um nodo?
  - Tem zero filhos (i.e., nodo folha)?
    - Simplesmente remova
  - Tem um filho?
    - Substitua pelo filho
  - Tem dois filhos?
    - Copie item do antecessor na sub-árvore esquerda (antecessor central) e remova o antecessor, ou
    - Copie item do sucessor na sub-árvore direita (sucessor central) e remova sucessor

ABB: remoção (caso 1)

UFMG COMPUTER SCIENCE

```
Tarvore p; // Apontador
...        // várias inserções
TItem x;
x.Chave = 4;
Retira(x, &p);
```

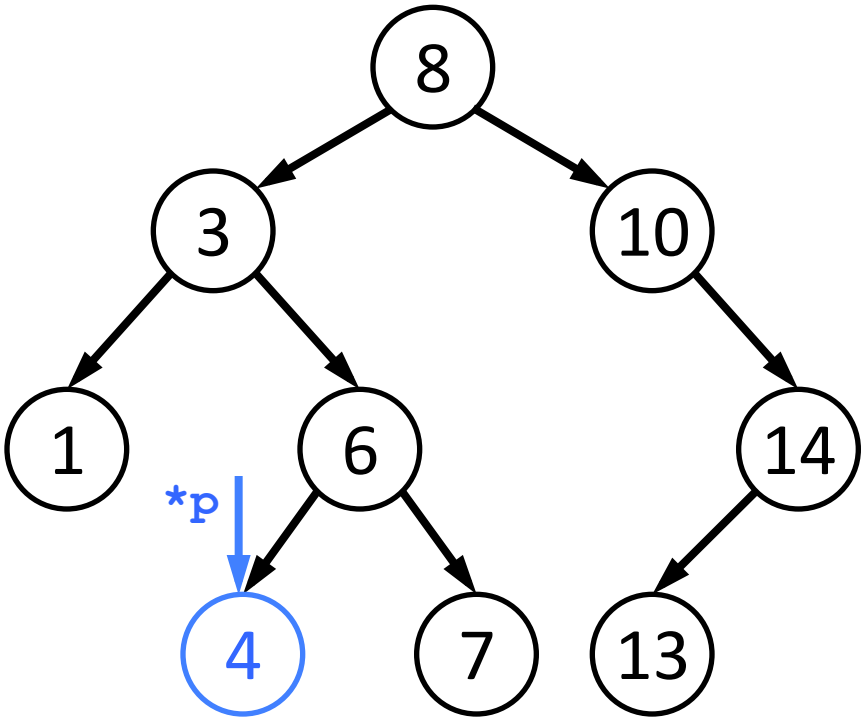


ABB: remoção (caso 1)

UFMG COMPUTER SCIENCE

```
Tarvore p; // Apontador
...        // várias inserções
TItem x;
x.Chave = 4;
Retira(x, &p);
```

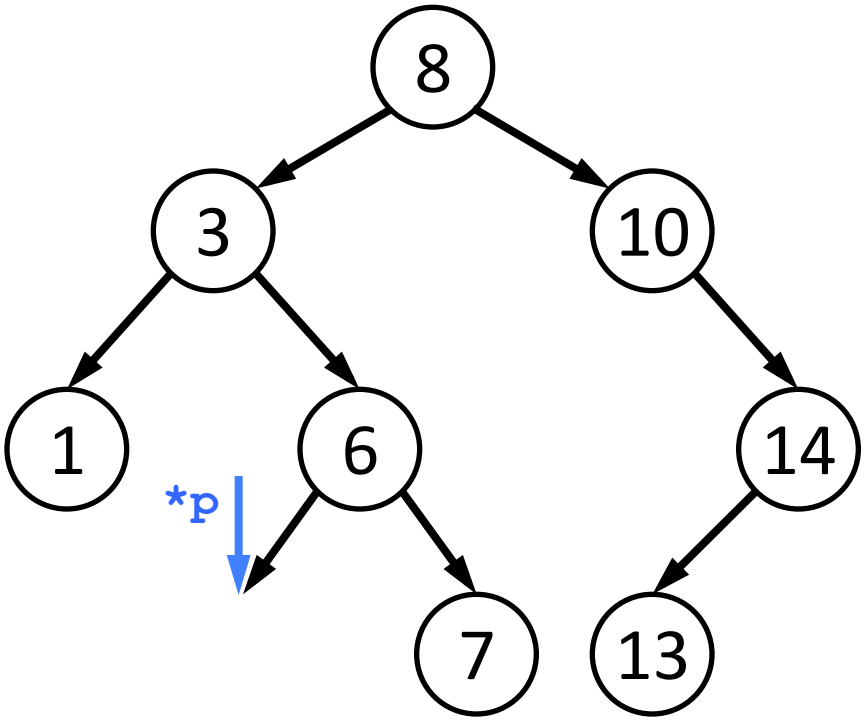


ABB: remoção (caso 2)

UFMG COMPUTER SCIENCE

```
TArvore p; // Apontador
...        // várias inserções
TItem x;
x.Chave = 14;
Retira(x, &p);
```

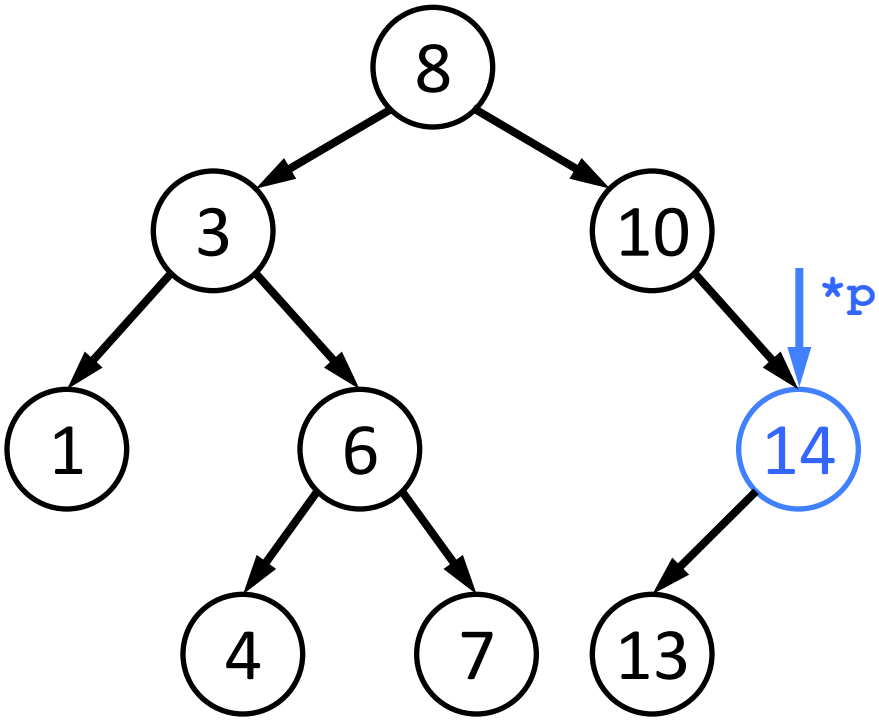


ABB: remoção (caso 2)

UFMG COMPUTER SCIENCE

```
TArvore P; // Apontador
...        // várias inserções
TItem x;
x.Chave = 14;
Retira(x, &p);
```

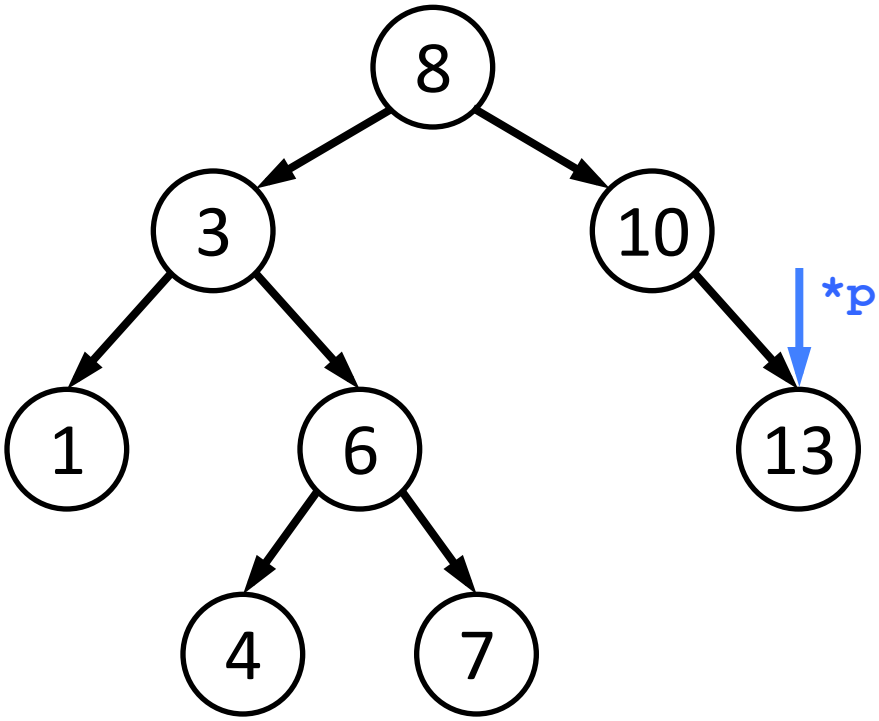


ABB: remoção (caso 3)

UFMG COMPUTER SCIENCE

```
TArvore p; // Apontador
...        // várias inserções
TItem x;
x.Chave = 8;
Retira(x, &p);
```

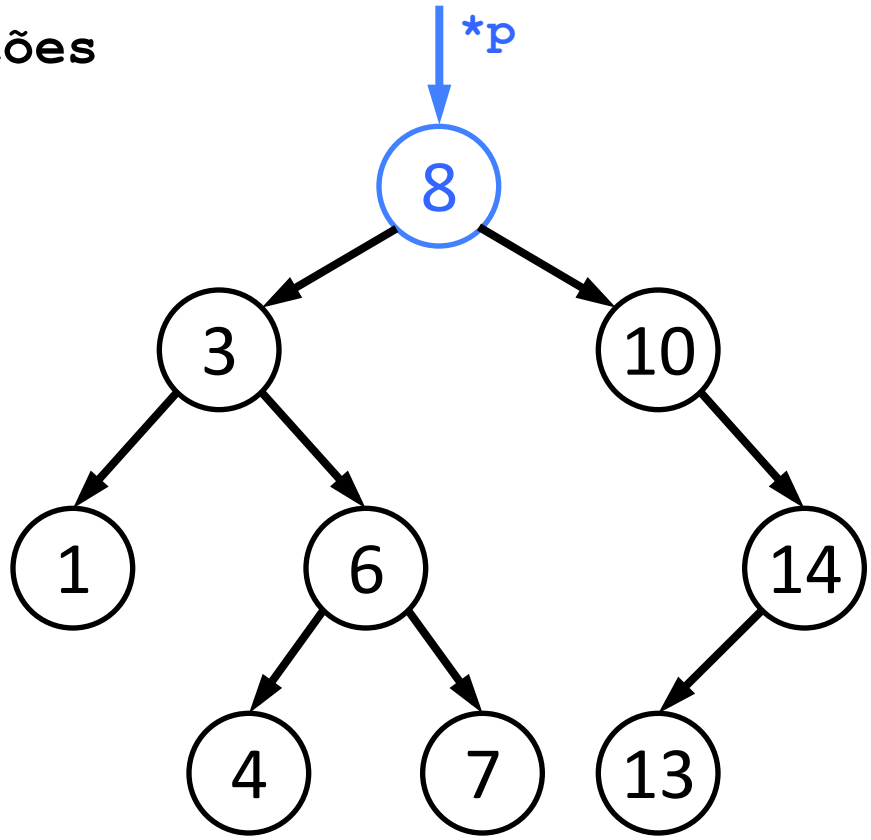


ABB: remoção (caso 3)

UFMG COMPUTER SCIENCE

```
TArvore p; // Apontador
...        // várias inserções
TItem x;
x.Chave = 8;
Retira(x, &p);
```

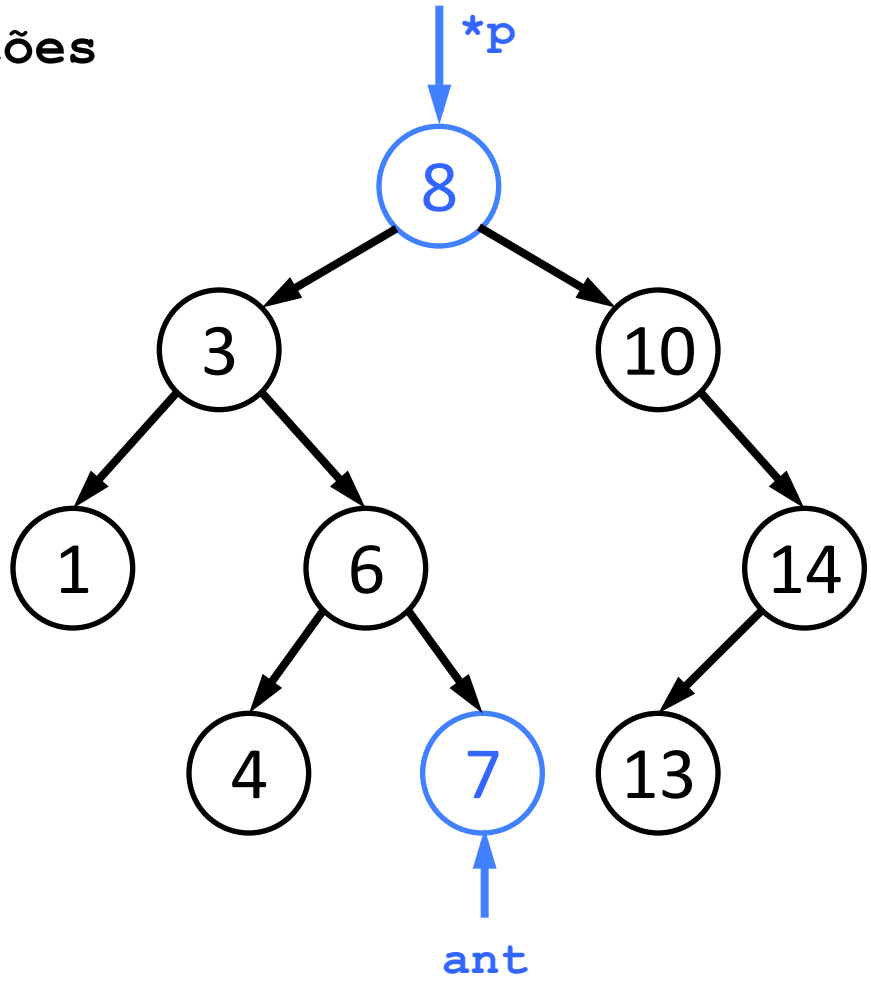


ABB: remoção (caso 3)

UFMG COMPUTER SCIENCE

```
TArvore p; // Apontador
...        // várias inserções
TItem x;
x.Chave = 8;
Retira(x, &p);
```

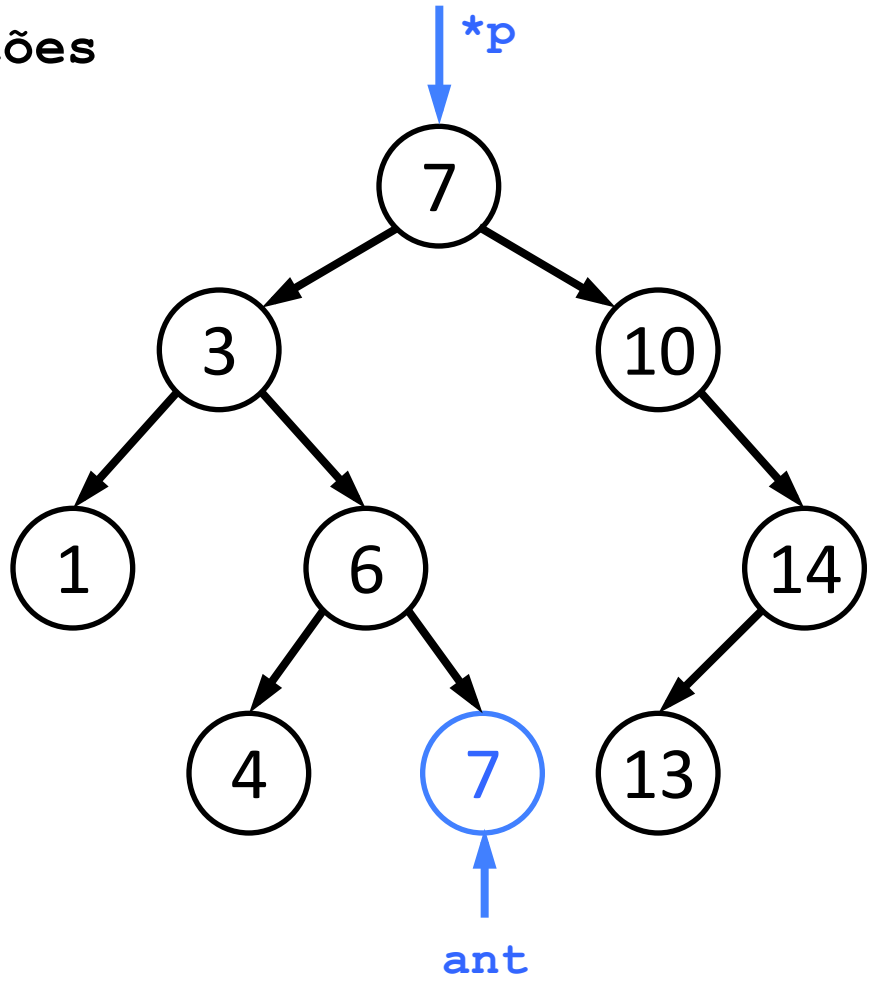


ABB: remoção (caso 3)

UFMG COMPUTER SCIENCE

```
TArvore p; // Apontador
...        // várias inserções
TItem x;
x.Chave = 8;
Retira(x, &p);
```

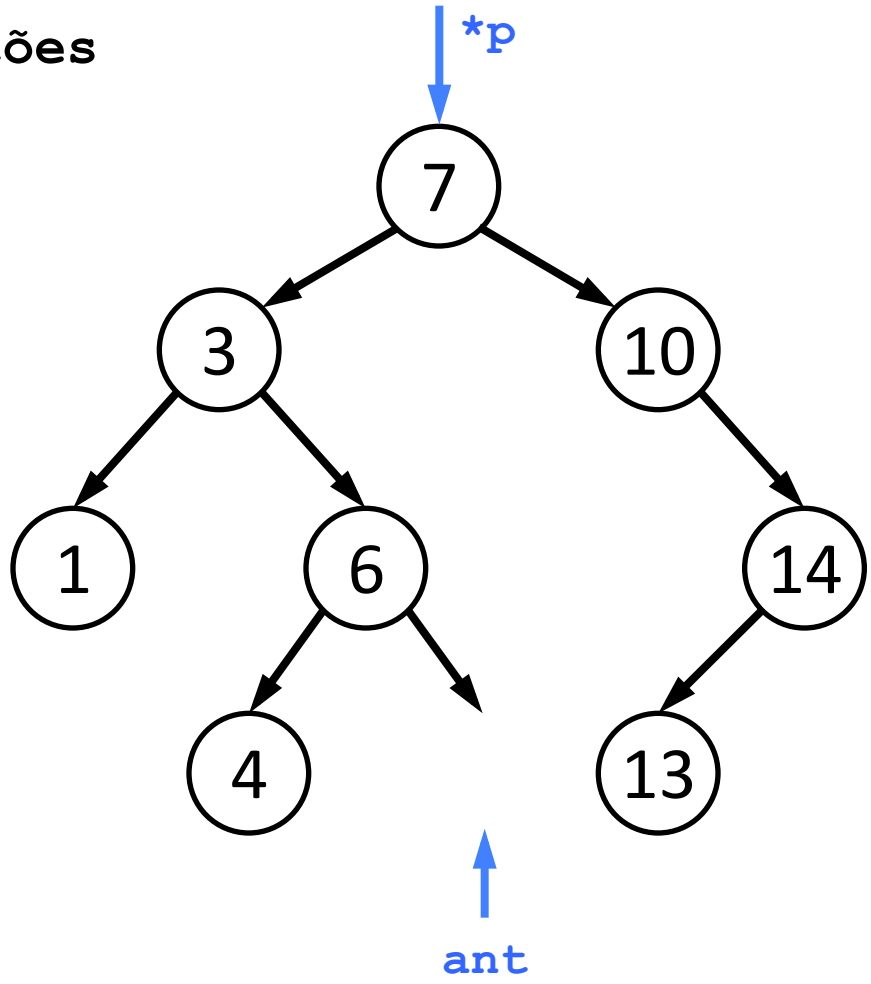


ABB: remoção (caso 3)

UFMG COMPUTER SCIENCE

```
TArvore p; // Apontador
...        // várias inserções
TItem x;
x.Chave = 8;
Retira(x, &p);
```

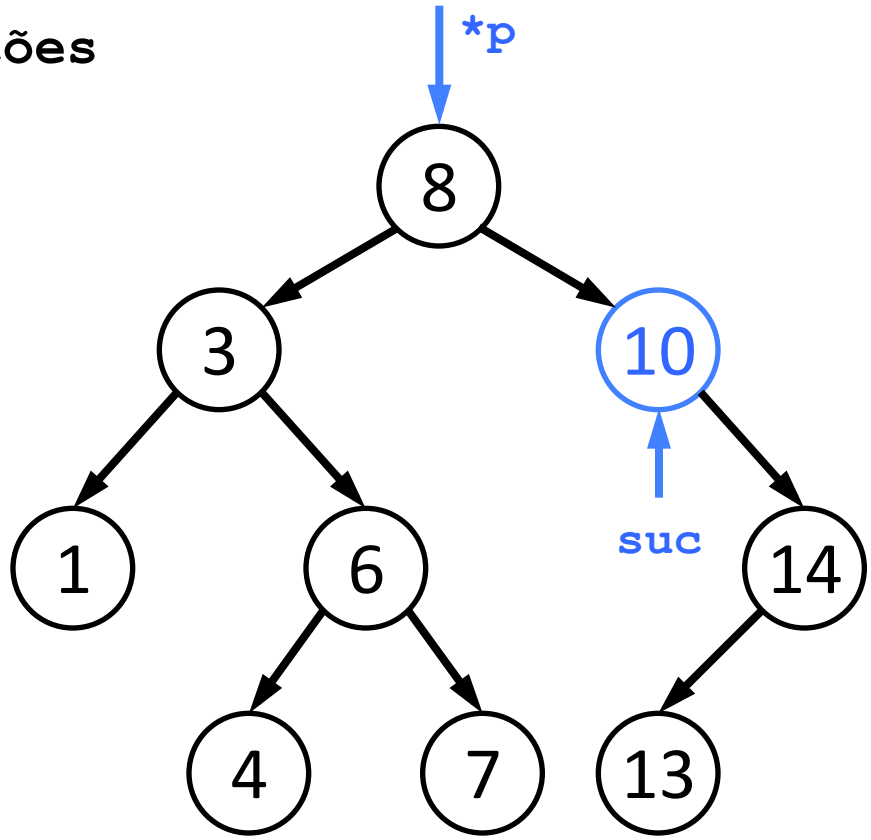


ABB: remoção (caso 3)

UFMG COMPUTER SCIENCE

```
TArvore p; // Apontador
...        // várias inserções
TItem x;
x.Chave = 8;
Retira(x, &p);
```

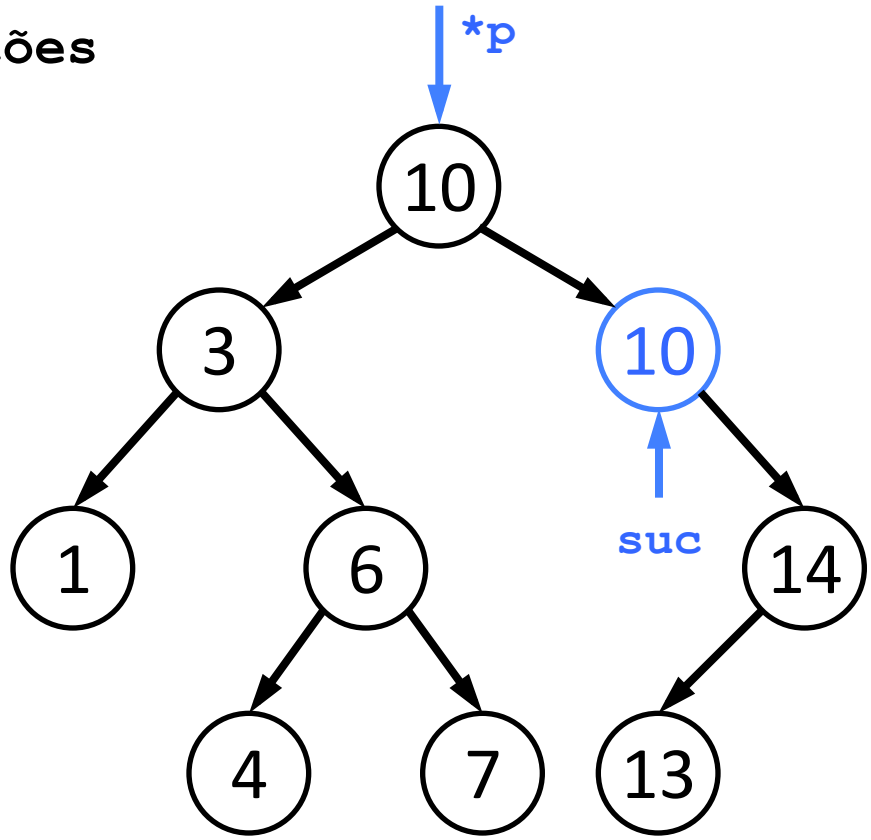


ABB: remoção (caso 3)

UFMG COMPUTER SCIENCE

```
TArvore p; // Apontador
...        // várias inserções
TItem x;
x.Chave = 8;
Retira(x, &p);
```

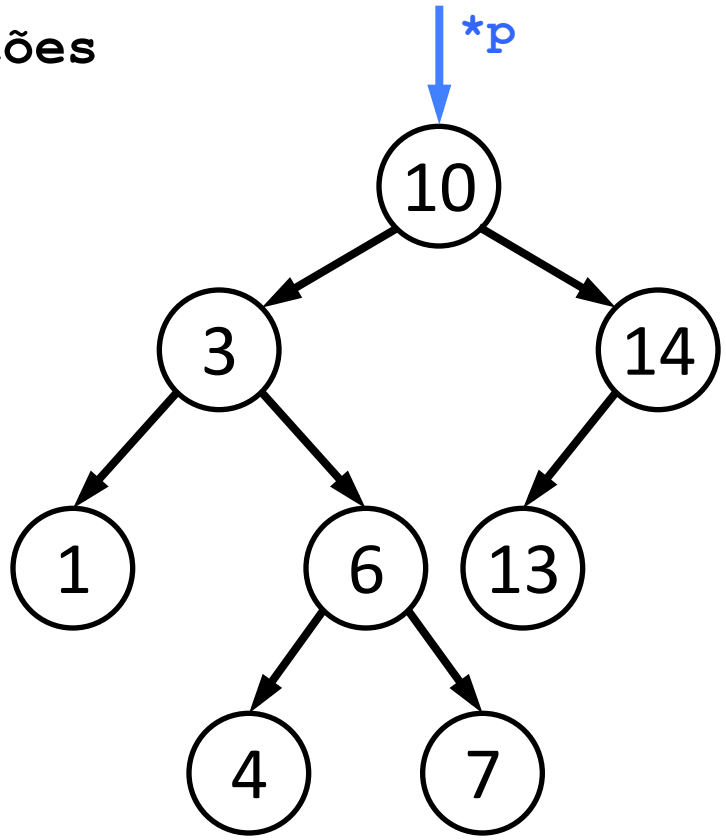


ABB: remoção

UFMG COMPUTER SCIENCE

```
void Retira(TItem x, Apontador *p) {
    Apontador Aux;
    if (*p == NULL)
        printf("ERRO: Item não encontrado\n");
    else if (x.Chave < (*p)->Item.Chave)
        Retira(x, &(*p)->Esq);
    else if (x.Chave > (*p)->Item.Chave)
        Retira(x, &(*p)->Dir);
    ...
```

ABB: remoção

UFMG COMPUTER SCIENCE

```
    ...
    else if ((*p)->Dir == NULL) {
        Aux = *p;
        *p = (*p)->Esq;
        free(Aux);
    } else if ((*p)->Esq == NULL) {
        Aux = *p;
        *p = (*p)->Dir;
        free(Aux);
    } else
        if (rand() % 100 + 1 < 50)
            Antecessor(*p, &(*p)->Esq);
        else
            Sucessor(*p, &(*p)->Dir);
}
```

Caso 1 + Caso 2

Caso 3

ABB: remoção

UFMG COMPUTER SCIENCE

```
void Antecessor(Apontador q, Apontador *r) {
    if ((*r)->Dir != NULL) {
        Antecessor(q, &(*r)->Dir);
        return;
    }
    q->Item = (*r)->Item;
    q = *r;
    *r = (*r)->Esq;
    free(q);
}
```

ABB: remoção

UFMG COMPUTER SCIENCE

```
void Sucessor(Apontador q, Apontador *r) {
    if ((*r)->Esq != NULL) {
        Sucessor(q, &(*r)->Esq);
        return;
    }
    q->Item = (*r)->Item;
    q = *r;
    *r = (*r)->Dir;
    free(q);
}
```

Caminhamento

UFMG COMPUTER SCIENCE

- Diversas formas de percorrer ou caminhar em uma árvore listando seus nós, as principais:
  - Pré-ordem (pré-fixada)
  - Central (infixada)
  - Pós-ordem (pós-fixada)
- Para todas elas:
  - Se T é uma árvore nula, então a lista é nula
  - Se T é uma árvore de um único nó então a lista contém apenas este nó

Caminhamento pré-ordem

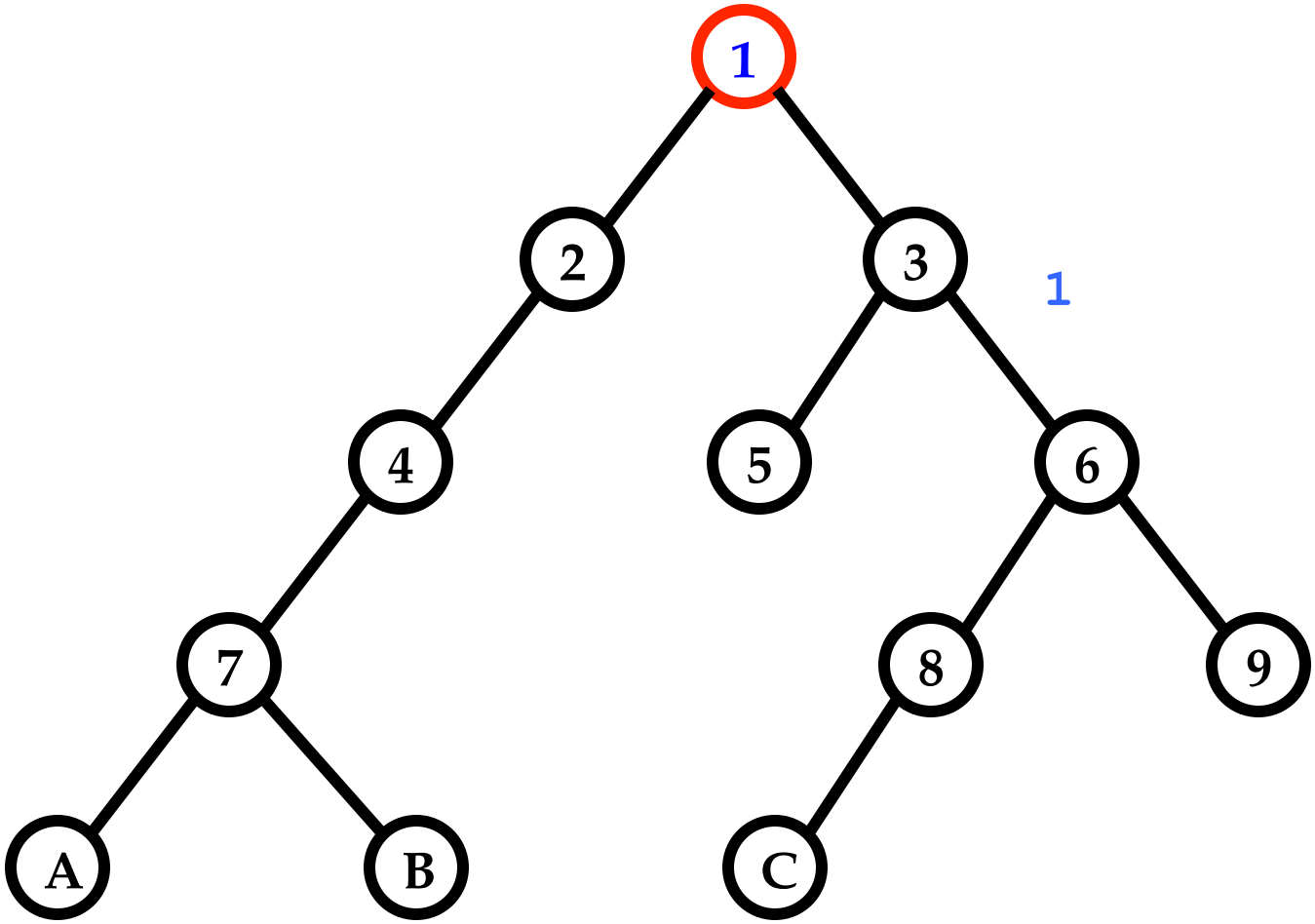
UFMG COMPUTER SCIENCE

- Idea:
  - **Lista raiz**
  - Visita sub-árvore esquerda em pré-ordem
  - Visita sub-árvore direita em pré-ordem

```
void PreOrdem(Apontador p) {
     if (p == null)
          return;
     printf("%d\n", p->Chave);
     PreOrdem(p->Esq);
     PreOrdem(p->Dir);
}
```

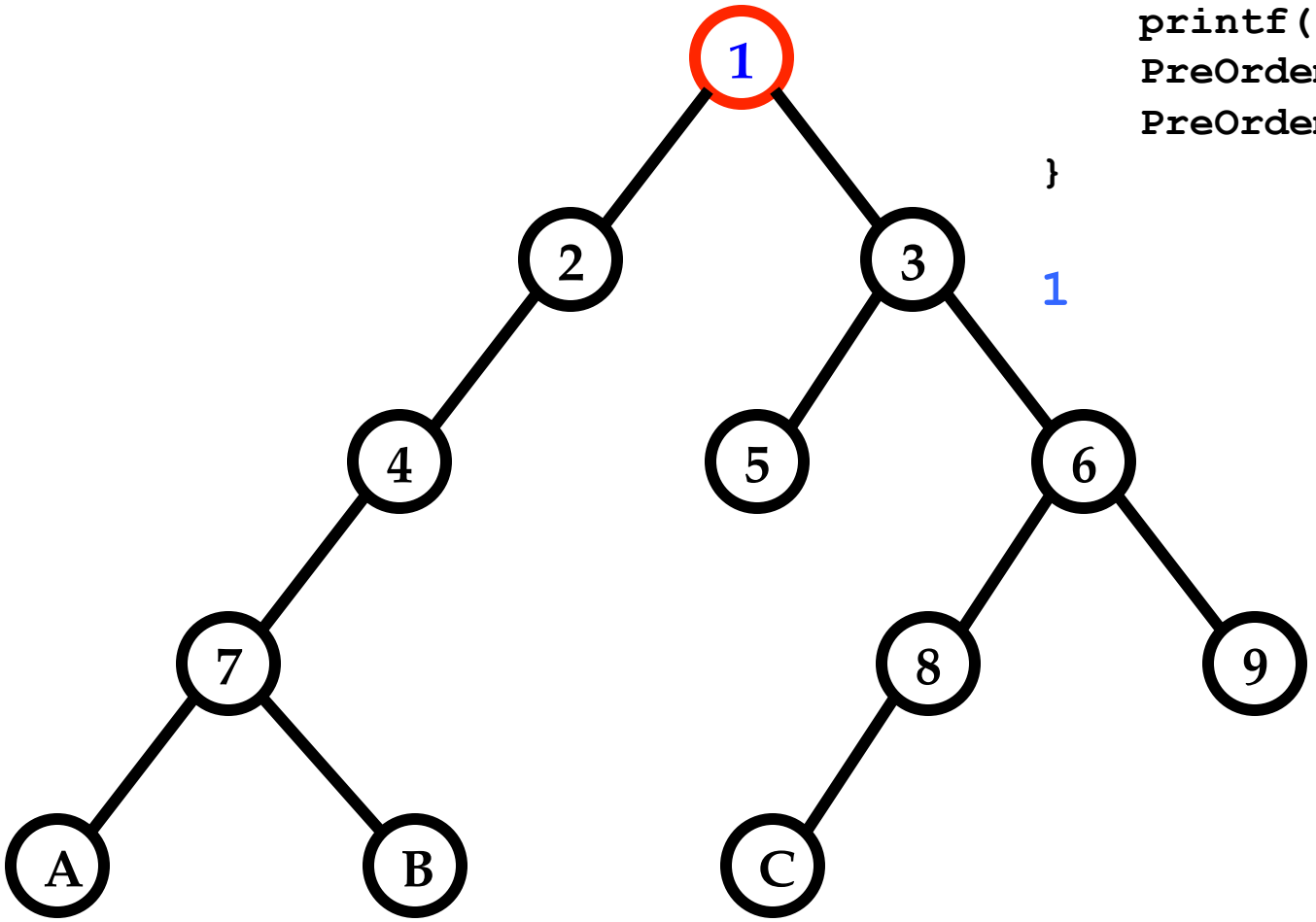
Caminhamento pré-ordem

UFMG COMPUTER SCIENCE



Caminhamento pré-ordem

UFMG COMPUTER SCIENCE




```
void PreOrdem(Apontador p) {
     if (p == null)
          return;
     printf("%d\n", p->Chave);
     PreOrdem(p->Esq);
     PreOrdem(p->Dir);
}
```

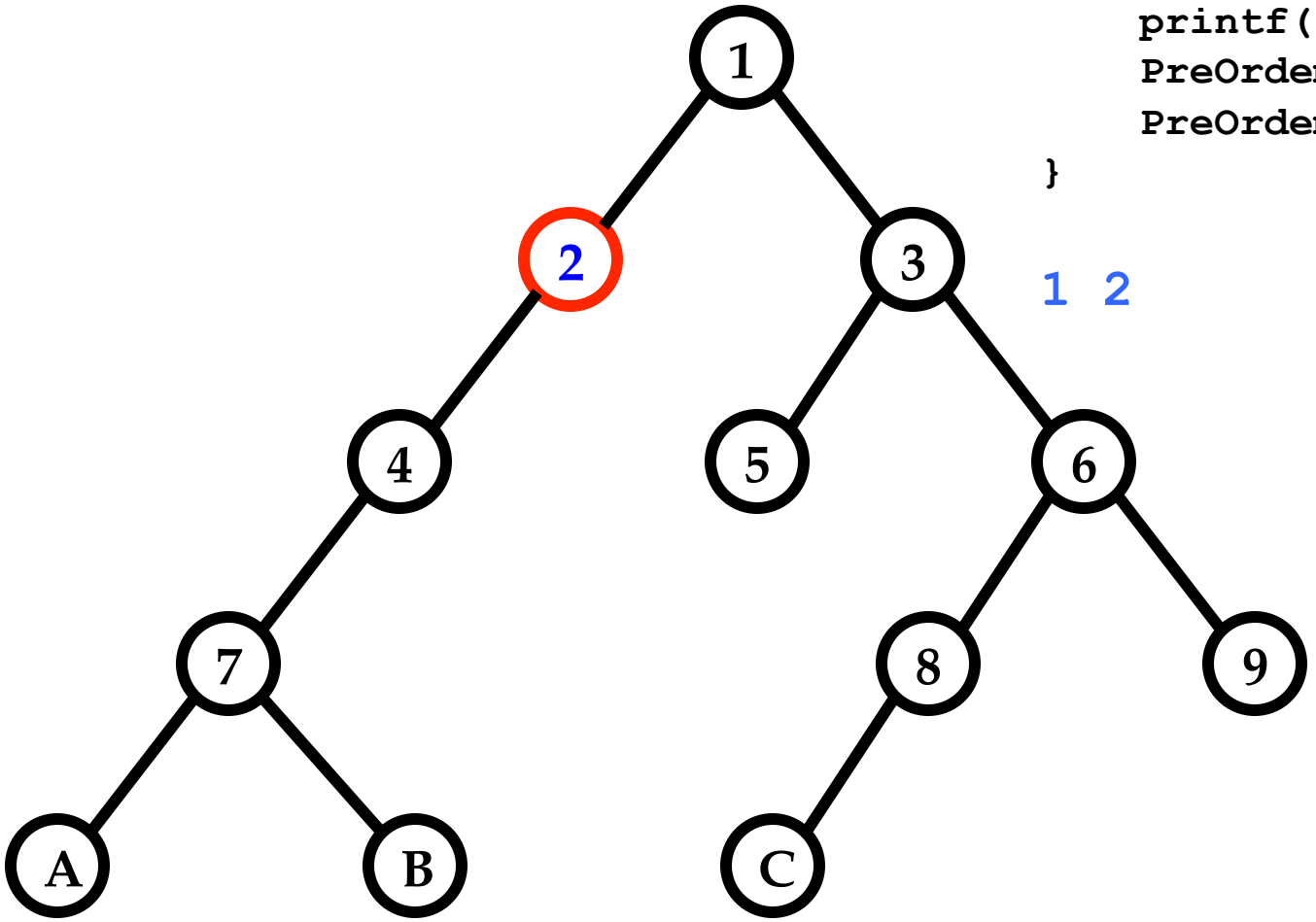
1

Caminhamento pré-ordem

UFMG COMPUTER SCIENCE

```
void PreOrdem(Apontador p) {
     if (p == null)
          return;
     printf("%d\n", p->Chave);
     PreOrdem(p->Esq);
     PreOrdem(p->Dir);
}
```

1 2

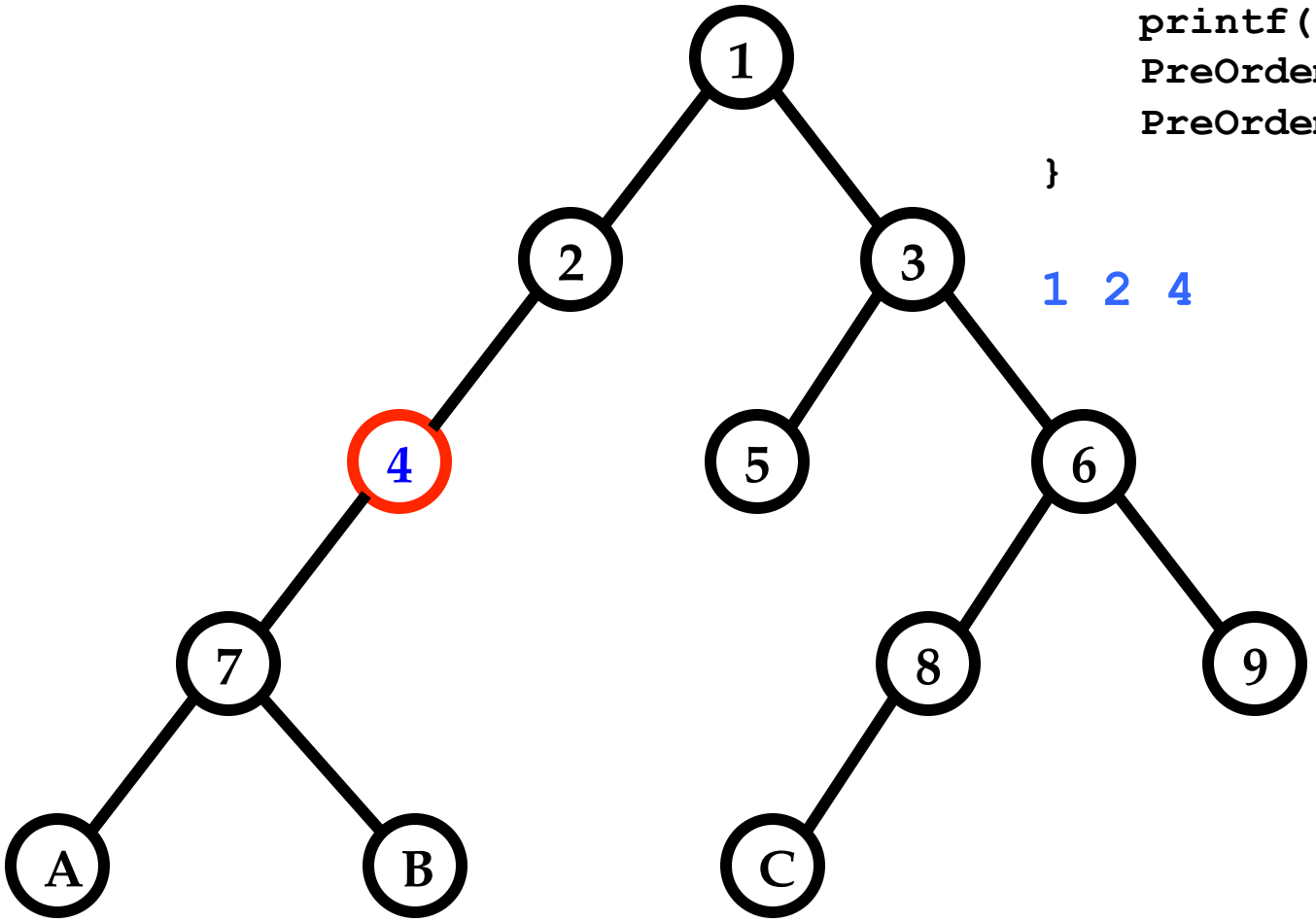


Caminhamento pré-ordem

UFMG COMPUTER SCIENCE

```
void PreOrdem(Apontador p) {
     if (p == null)
          return;
     printf("%d\n", p->Chave);
     PreOrdem(p->Esq);
     PreOrdem(p->Dir);
}
```

1 2 4

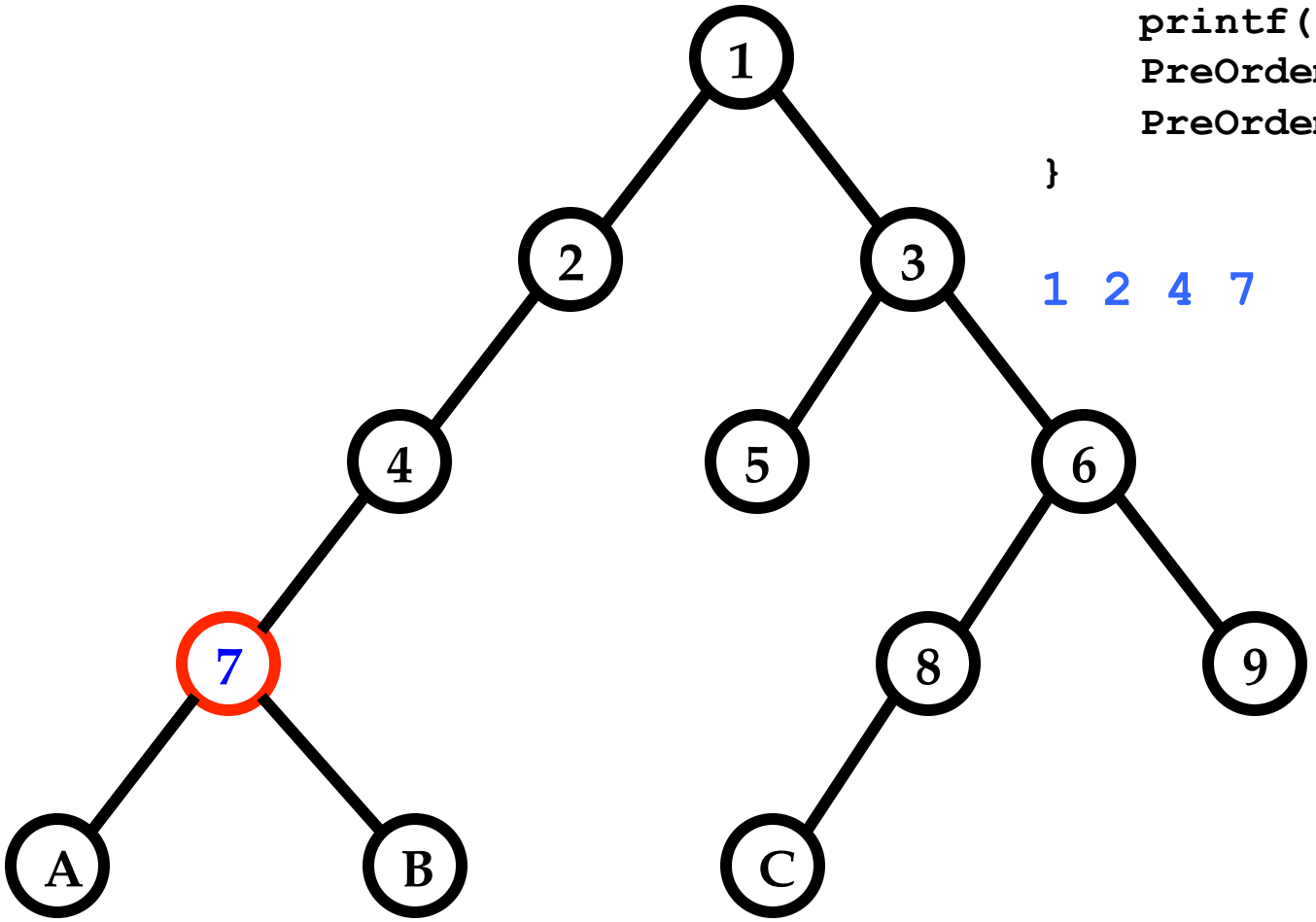


Caminhamento pré-ordem

UFMG COMPUTER SCIENCE

```
void PreOrdem(Apontador p) {
     if (p == null)
          return;
     printf("%d\n", p->Chave);
     PreOrdem(p->Esq);
     PreOrdem(p->Dir);
}
```

1 2 4 7

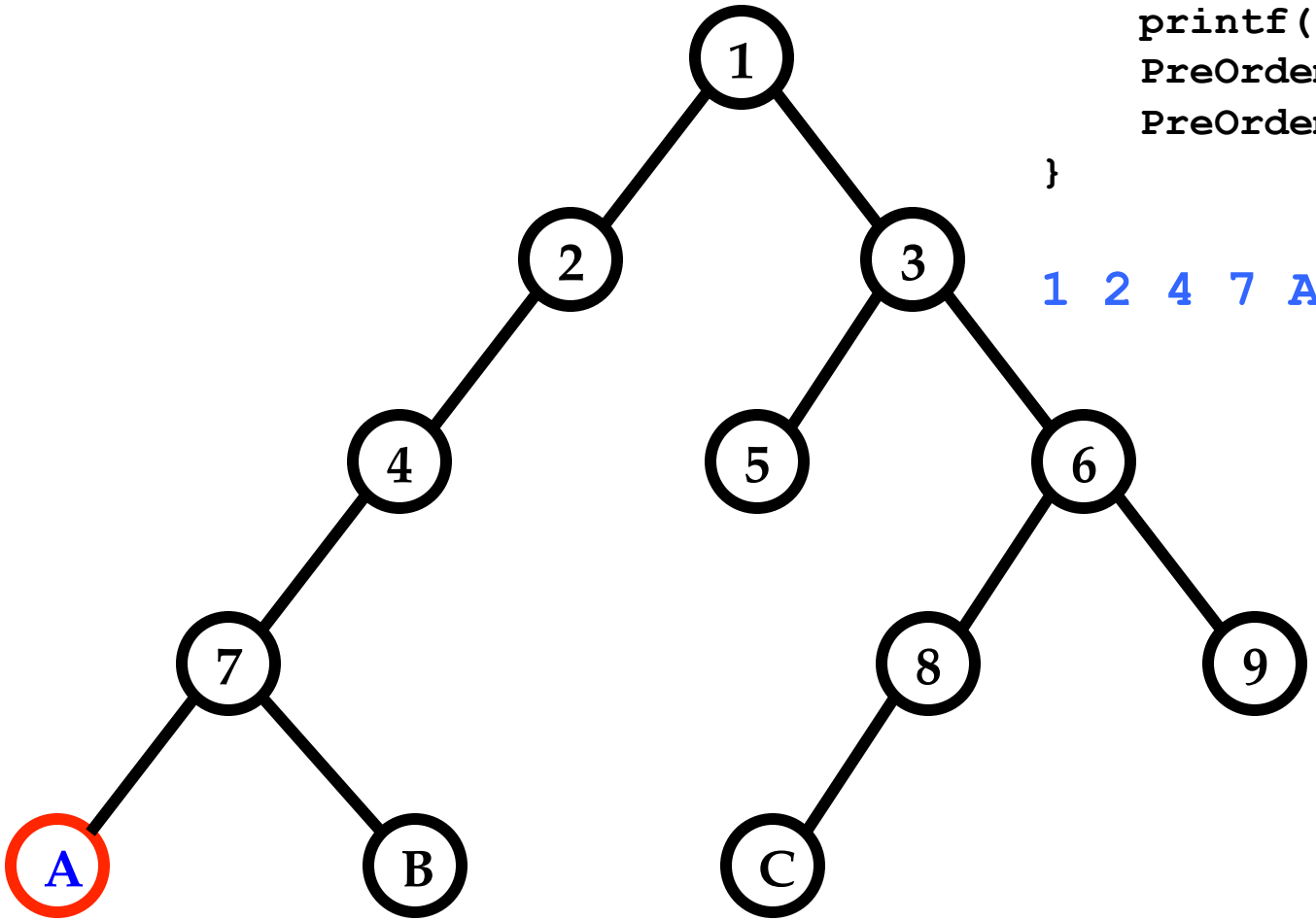


Caminhamento pré-ordem

UFMG COMPUTER SCIENCE

```
void PreOrdem(Apontador p) {
     if (p == null)
          return;
     printf("%d\n", p->Chave);
     PreOrdem(p->Esq);
     PreOrdem(p->Dir);
}
```

1 2 4 7 A

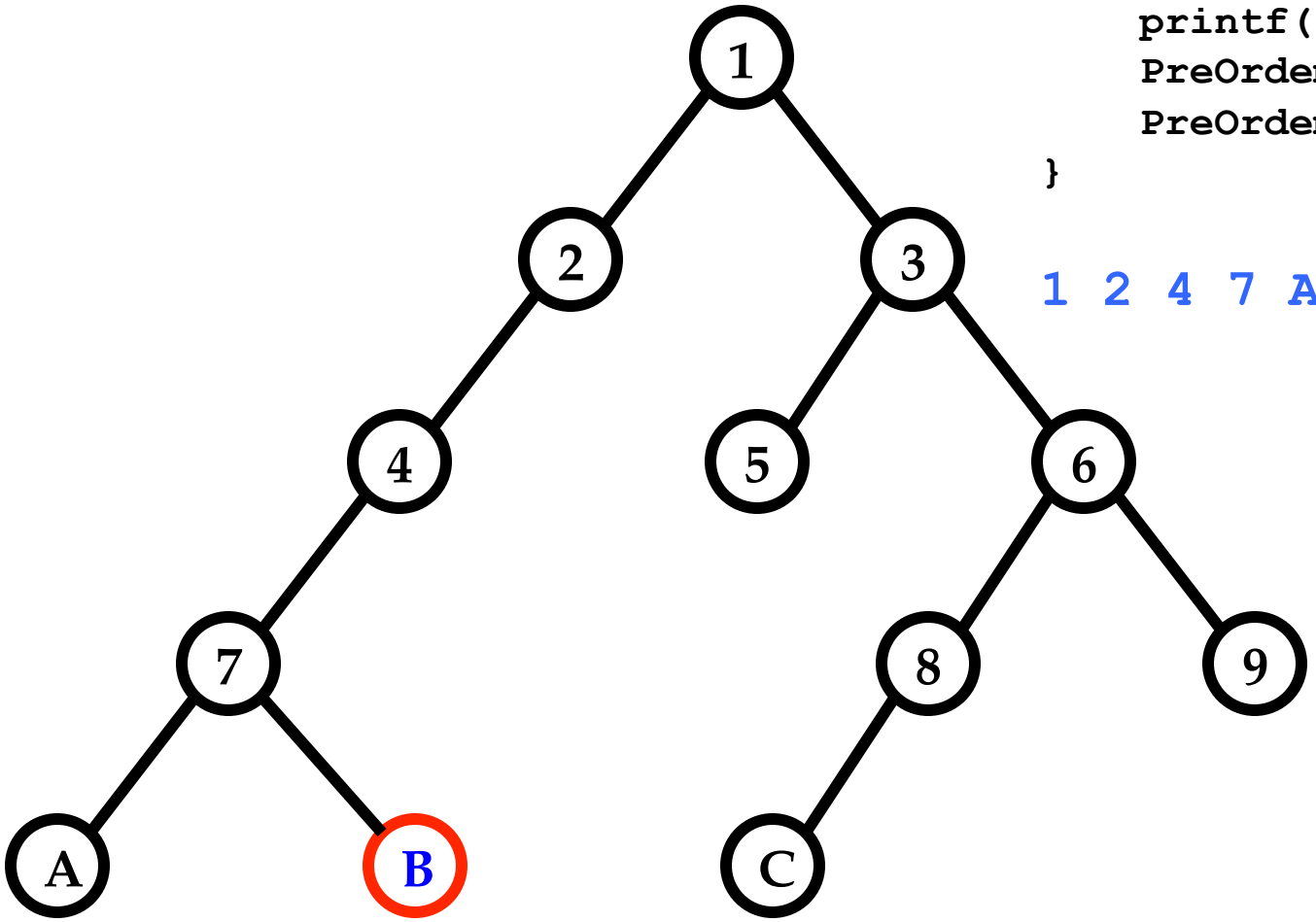


Caminhamento pré-ordem

UFMG COMPUTER SCIENCE

```
void PreOrdem(Apontador p) {
     if (p == null)
          return;
     printf("%d\n", p->Chave);
     PreOrdem(p->Esq);
     PreOrdem(p->Dir);
}
```

1 2 4 7 A B

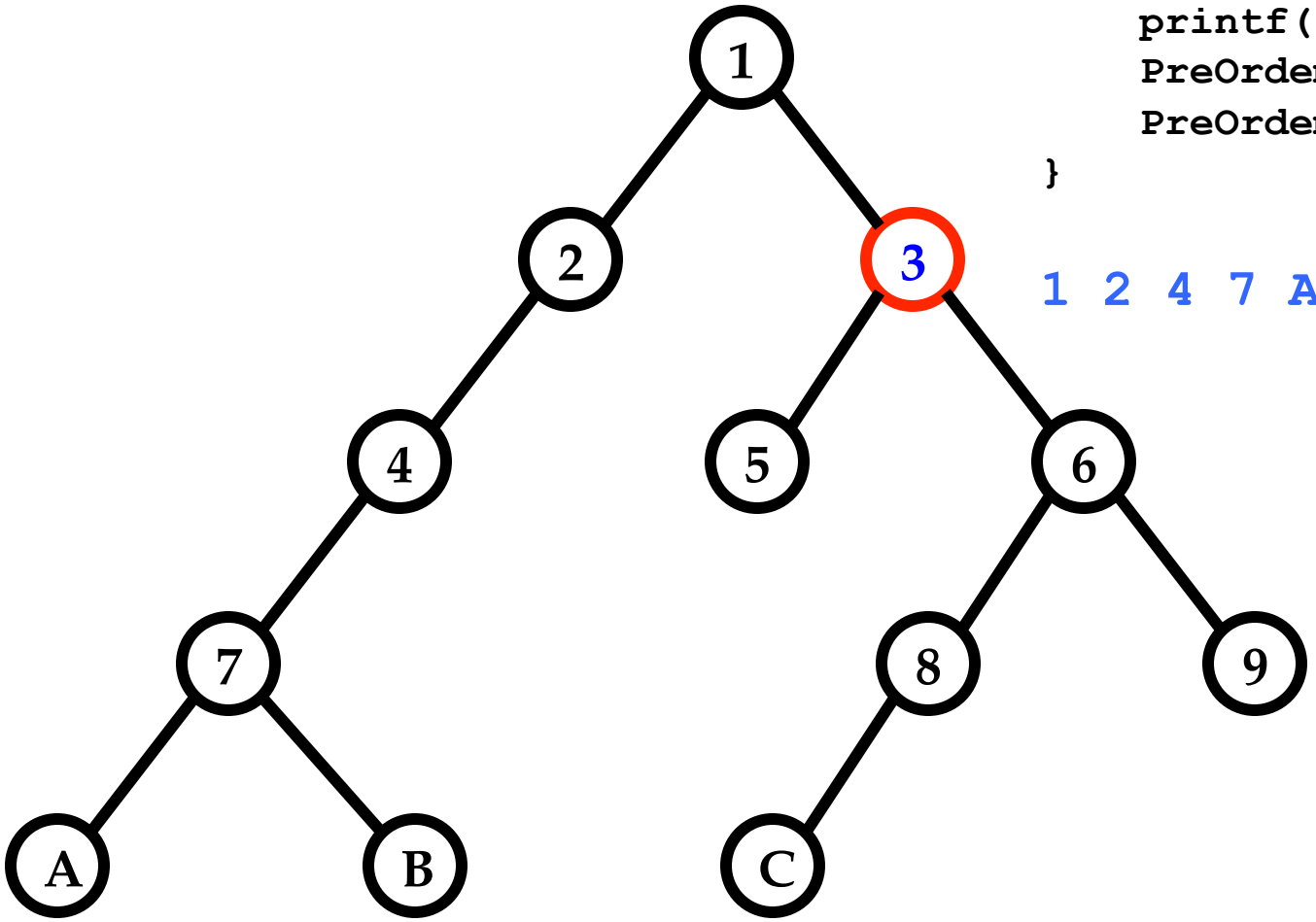


Caminhamento pré-ordem

UFMG COMPUTER SCIENCE

```
void PreOrdem(Apontador p) {
     if (p == null)
          return;
     printf("%d\n", p->Chave);
     PreOrdem(p->Esq);
     PreOrdem(p->Dir);
}
```

1 2 4 7 A B 3

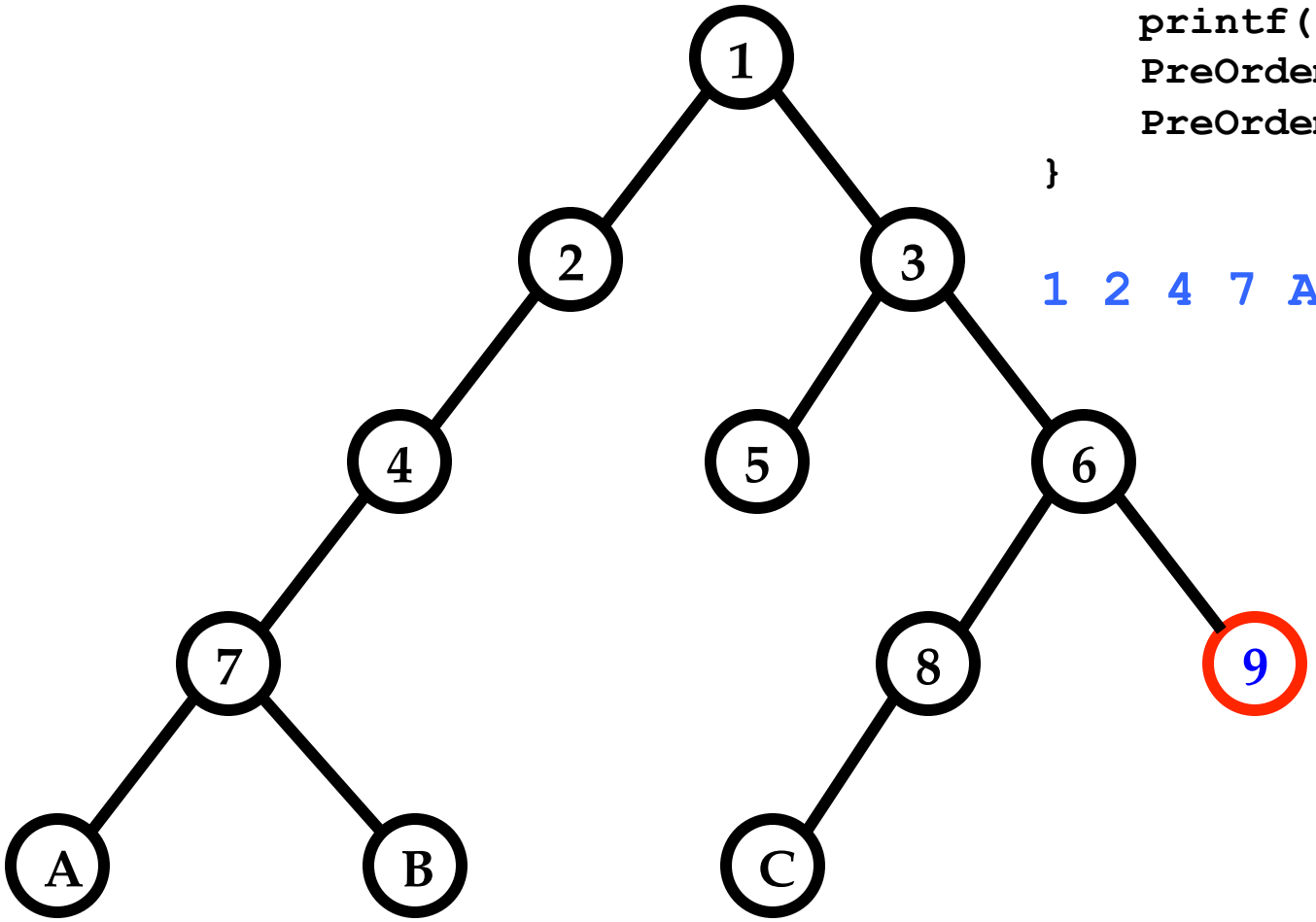


Caminhamento pré-ordem

UFMG COMPUTER SCIENCE

```
void PreOrdem(Apontador p) {
     if (p == null)
          return;
     printf("%d\n", p->Chave);
     PreOrdem(p->Esq);
     PreOrdem(p->Dir);
}
```

1 2 4 7 A B 3 5 6 8 C 9



Caminhamento central

UFMG COMPUTER SCIENCE

- Idea:
  - Visita sub-árvore esquerda em pré-ordem
  - **Lista raiz**
  - Visita sub-árvore direita em pré-ordem

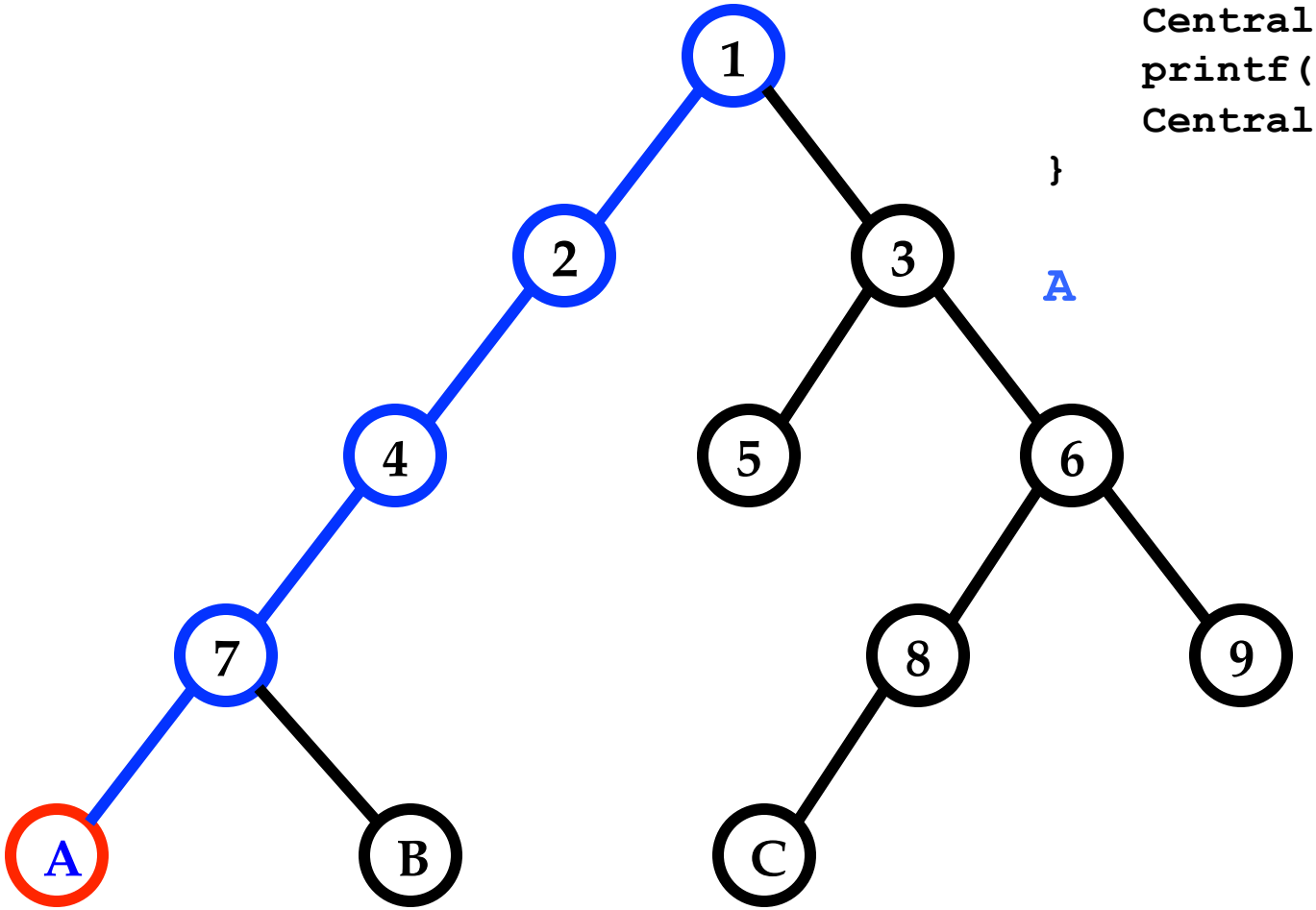
```
void Central(Apontador p) {
     if (p == null)
          return;
     Central(p->Esq);
     printf("%d\n", p->Chave);
     Central(p->Dir);
}
```

Caminhamento central

UFMG COMPUTER SCIENCE

```
void Central(Apontador p) {
     if (p == null)
          return;
     Central(p->Esq);
     printf("%d\n", p->Chave);
     Central(p->Dir);
}
```

A

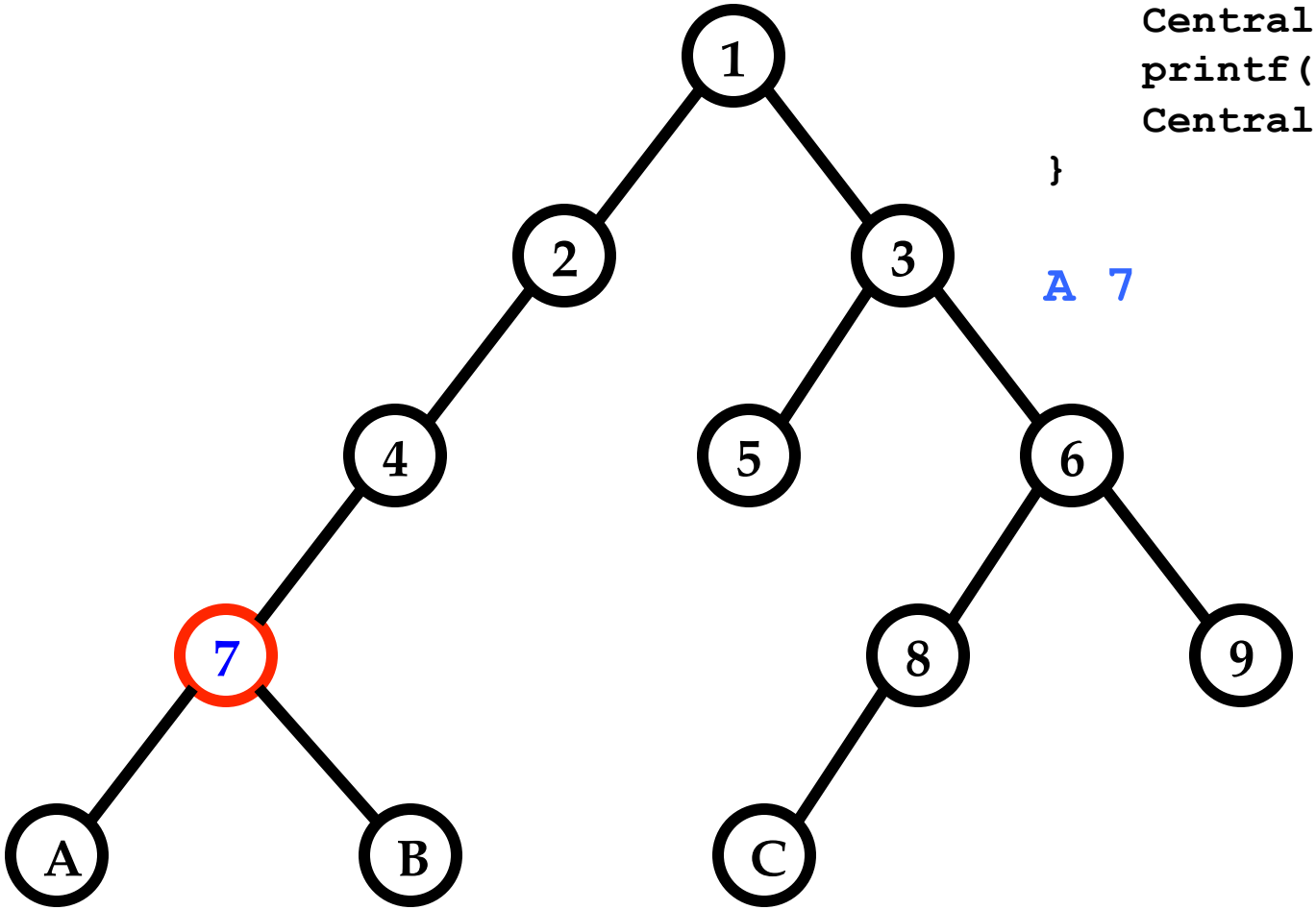


Caminhamento central

UFMG COMPUTER SCIENCE

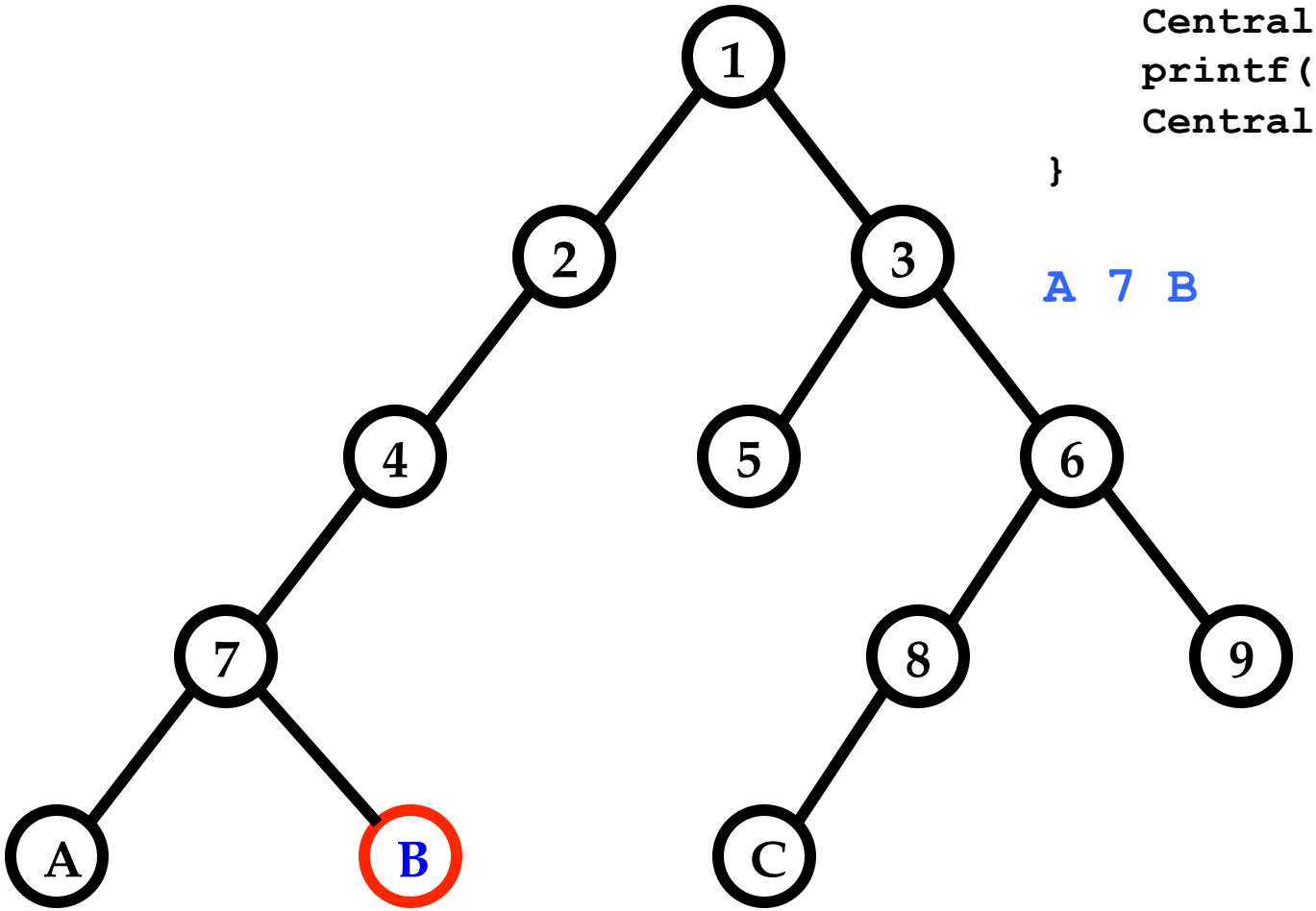
```
void Central(Apontador p) {
     if (p == null)
          return;
     Central(p->Esq);
     printf("%d\n", p->Chave);
     Central(p->Dir);
}
```

A 7



Caminhamento central

UFMG COMPUTER SCIENCE

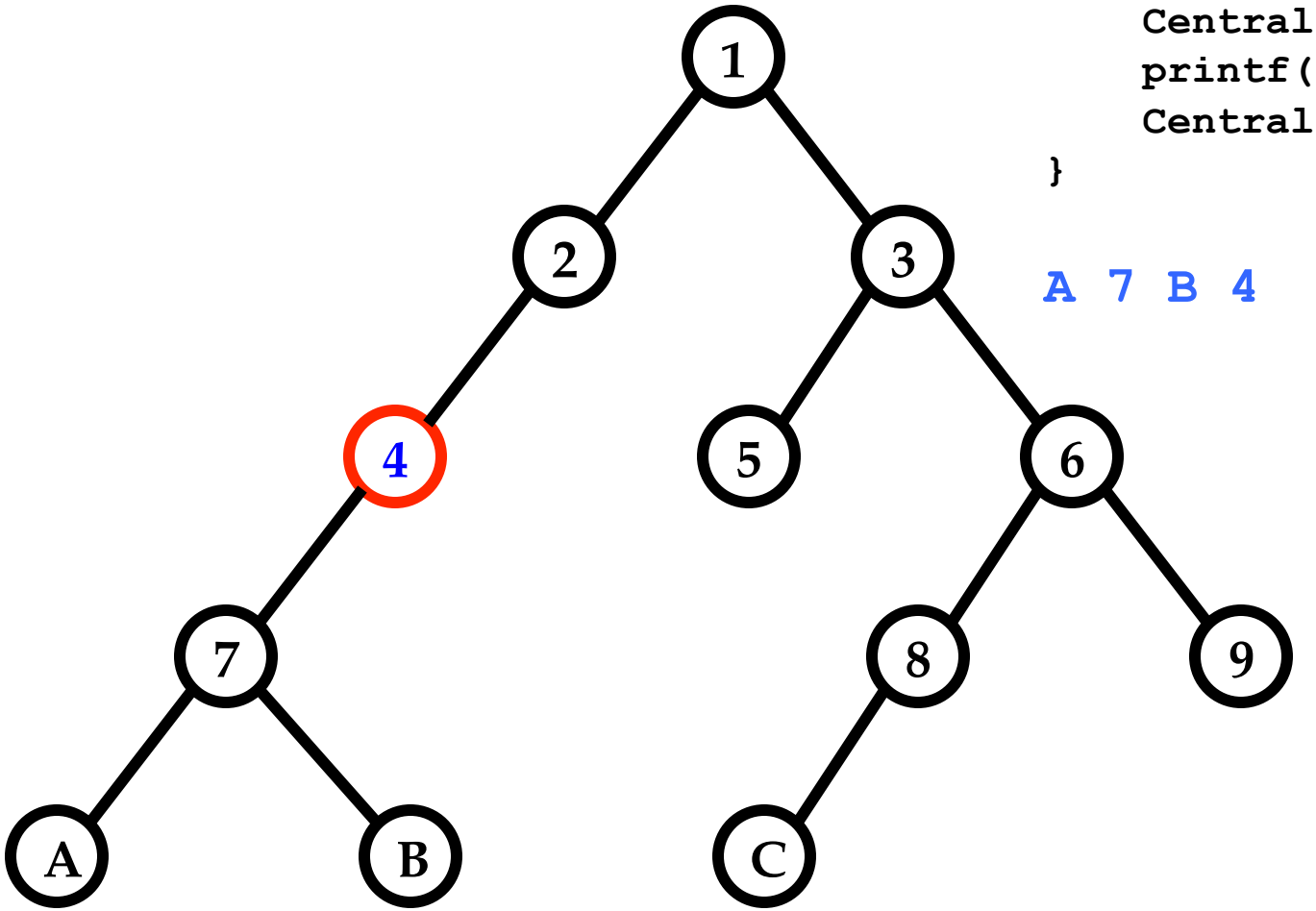



```
void Central(Apontador p) {
     if (p == null)
          return;
     Central(p->Esq);
     printf("%d\n", p->Chave);
     Central(p->Dir);
}
```

A 7 B

Caminhamento central

UFMG COMPUTER SCIENCE




```
void Central(Apontador p) {
     if (p == null)
          return;
     Central(p->Esq);
     printf("%d\n", p->Chave);
     Central(p->Dir);
}
```

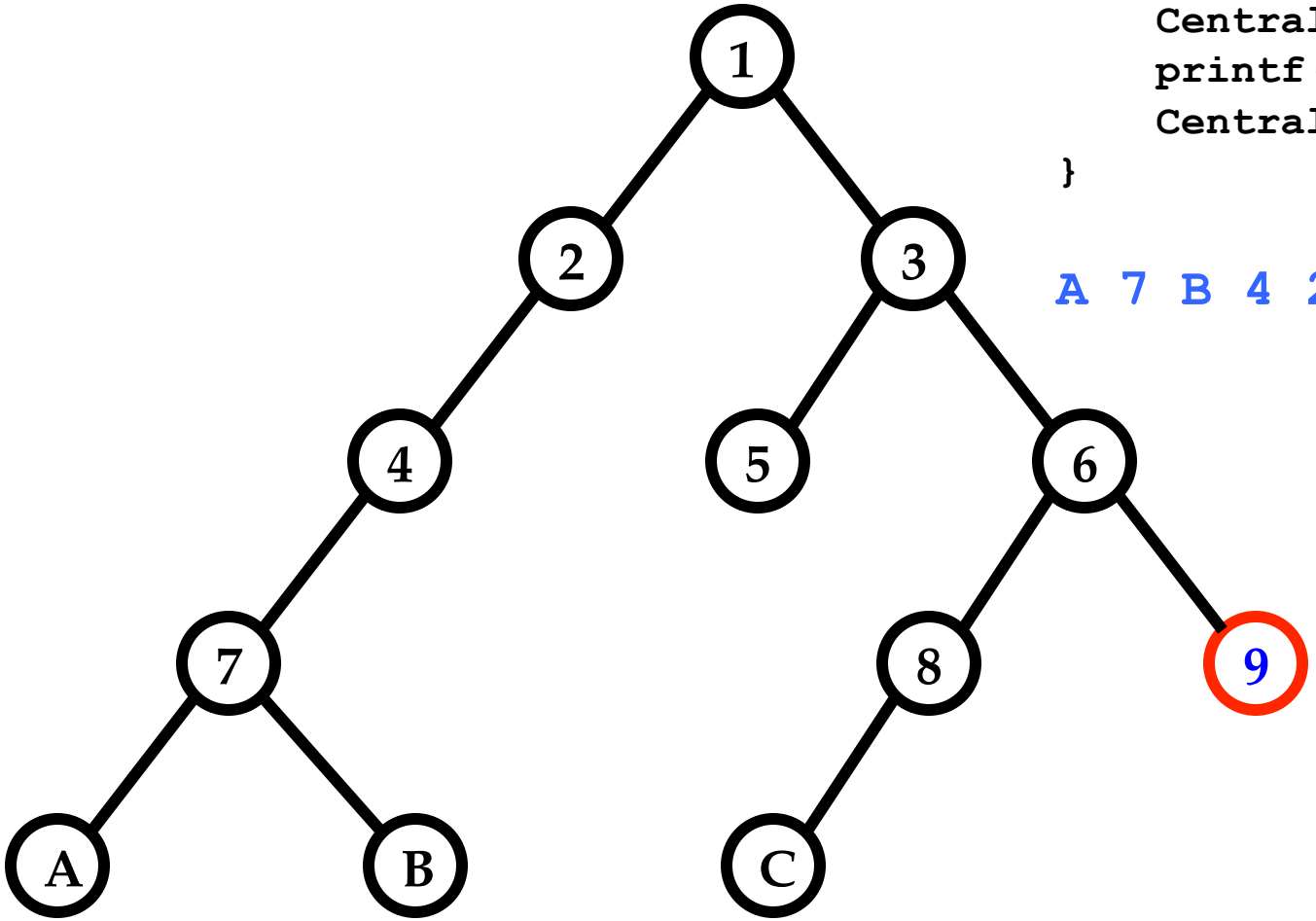
A 7 B 4

Caminhamento central

UFMG COMPUTER SCIENCE

```
void Central(Apontador p) {
     if (p == null)
          return;
     Central(p->Esq);
     printf("%d\n", p->Chave);
     Central(p->Dir);
}
```

A 7 B 4 2 1 5 3 C 8 6 9



Caminhamento pós-ordem

UFMG COMPUTER SCIENCE

- Idea:
  - Visita sub-árvore esquerda em pré-ordem
  - Visita sub-árvore direita em pré-ordem
  - **Lista raiz**

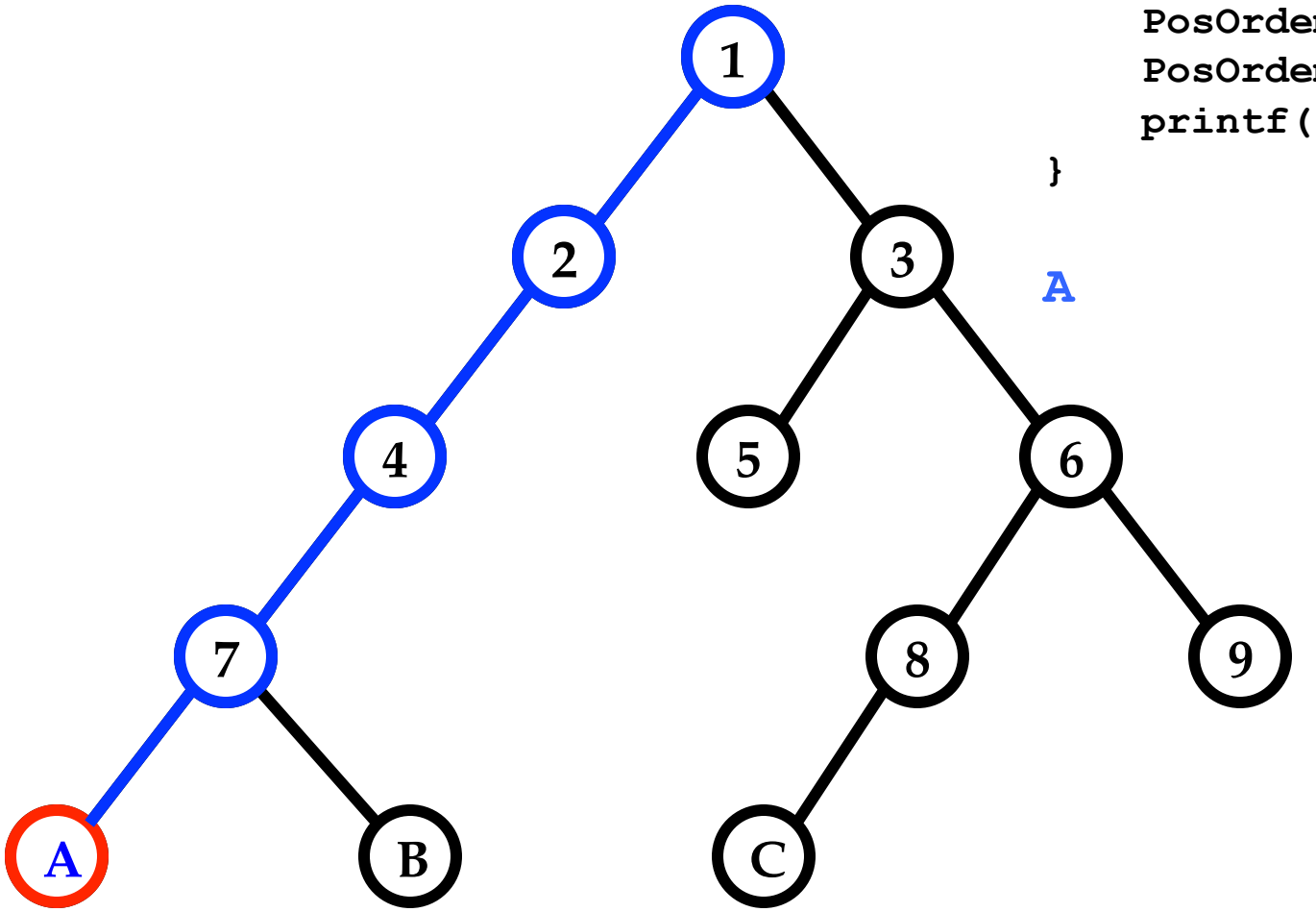
```
void PosOrdem(Apontador p) {
    if (p == null)
        return;
    PosOrdem(p->Esq);
    PosOrdem(p->Dir);
    printf("%d\n", p->Chave);
}
```

Caminhamento pós-ordem

UFMG COMPUTER SCIENCE

```
void PosOrdem(Apontador p) {
     if (p == null)
          return;
     PosOrdem(p->Esq);
     PosOrdem(p->Dir);
     printf("%d\n", p->Chave);
}
```

A

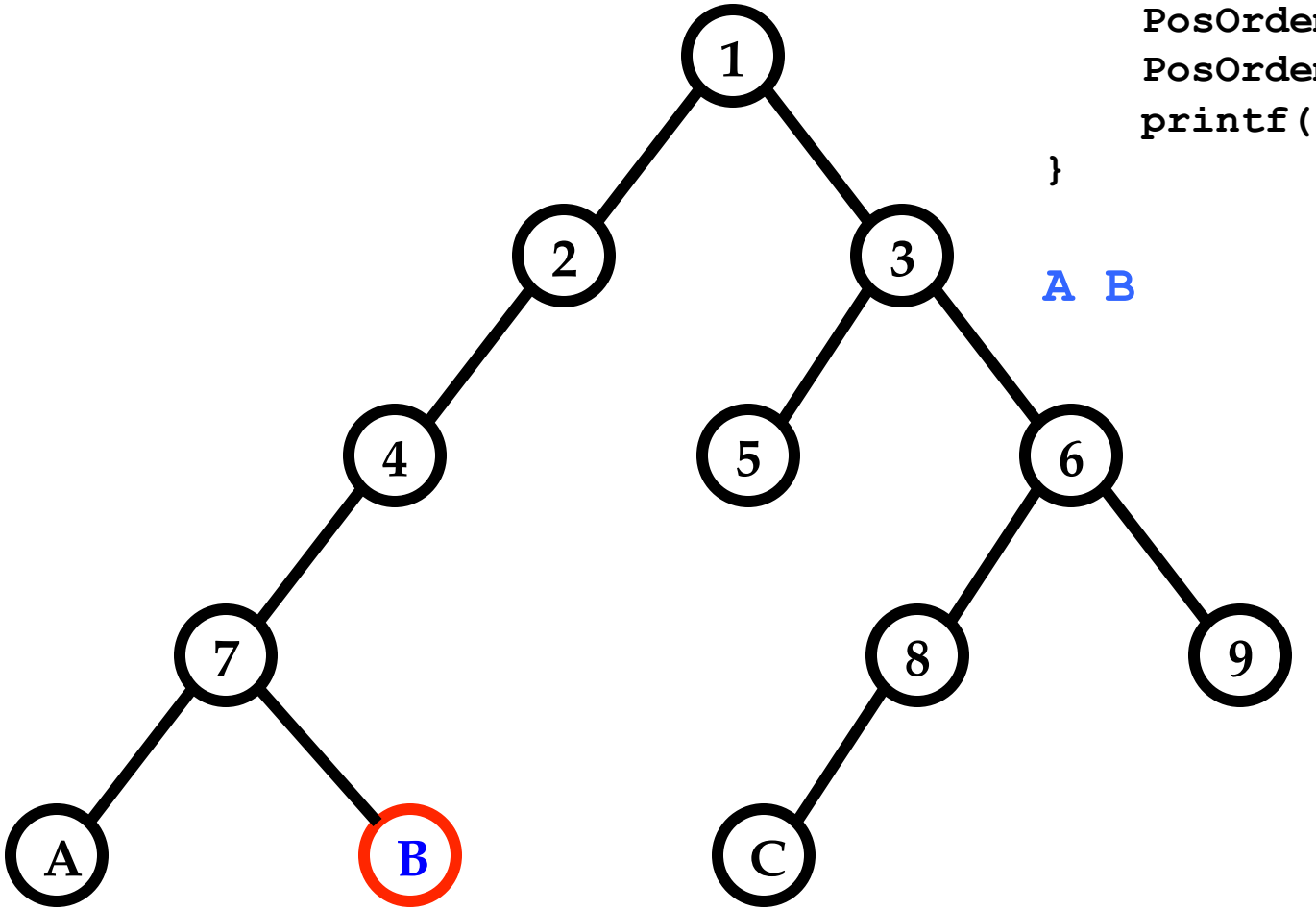


Caminhamento pós-ordem

UFMG COMPUTER SCIENCE

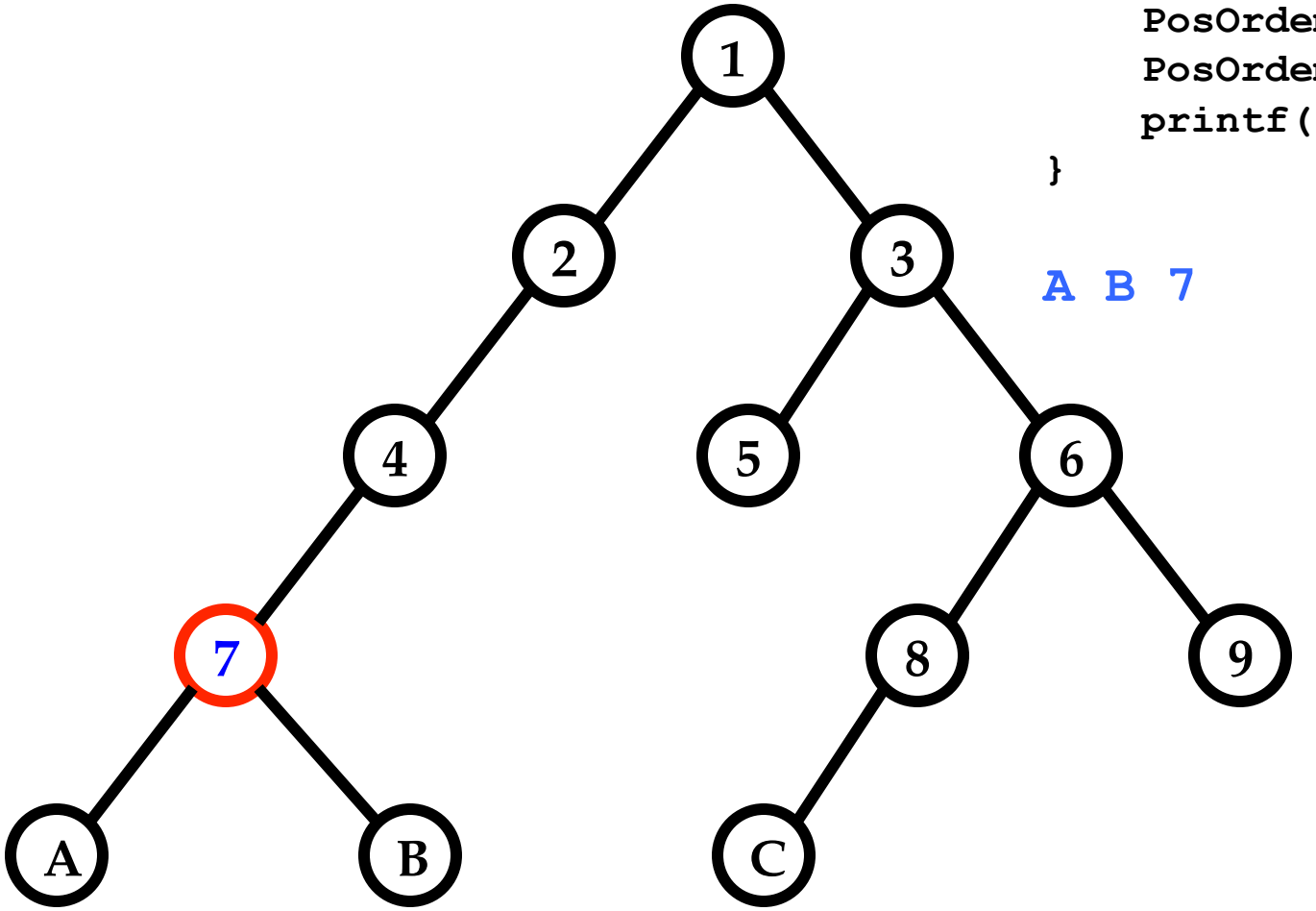
```
void PosOrdem(Apontador p) {
     if (p == null)
          return;
     PosOrdem(p->Esq);
     PosOrdem(p->Dir);
     printf("%d\n", p->Chave);
}
```

A B



Caminhamento pós-ordem

UFMG COMPUTER SCIENCE




```
void PosOrdem(Apontador p) {
     if (p == null)
          return;
     PosOrdem(p->Esq);
     PosOrdem(p->Dir);
     printf("%d\n", p->Chave);
}
```

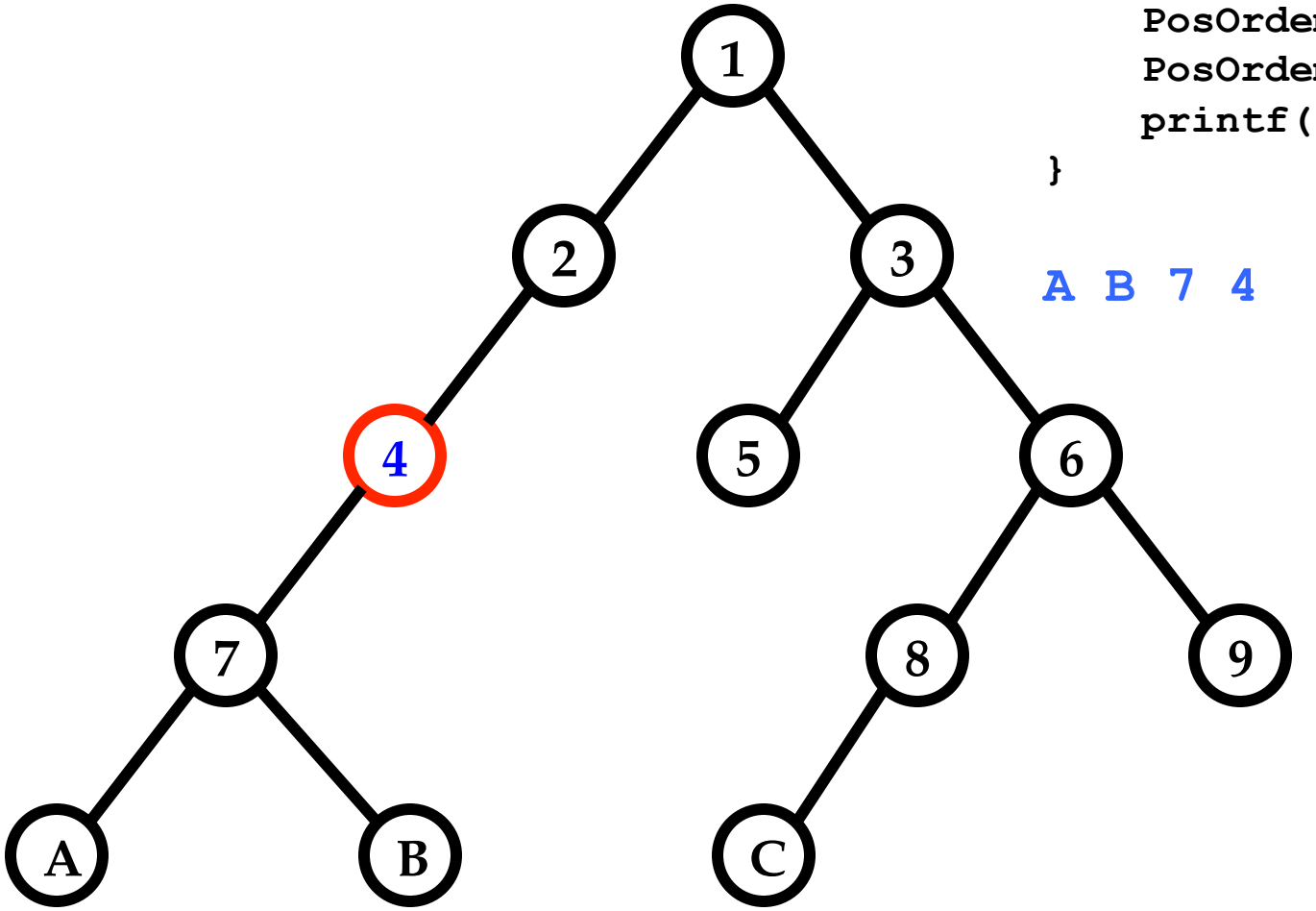
A B 7

Caminhamento pós-ordem

UFMG COMPUTER SCIENCE

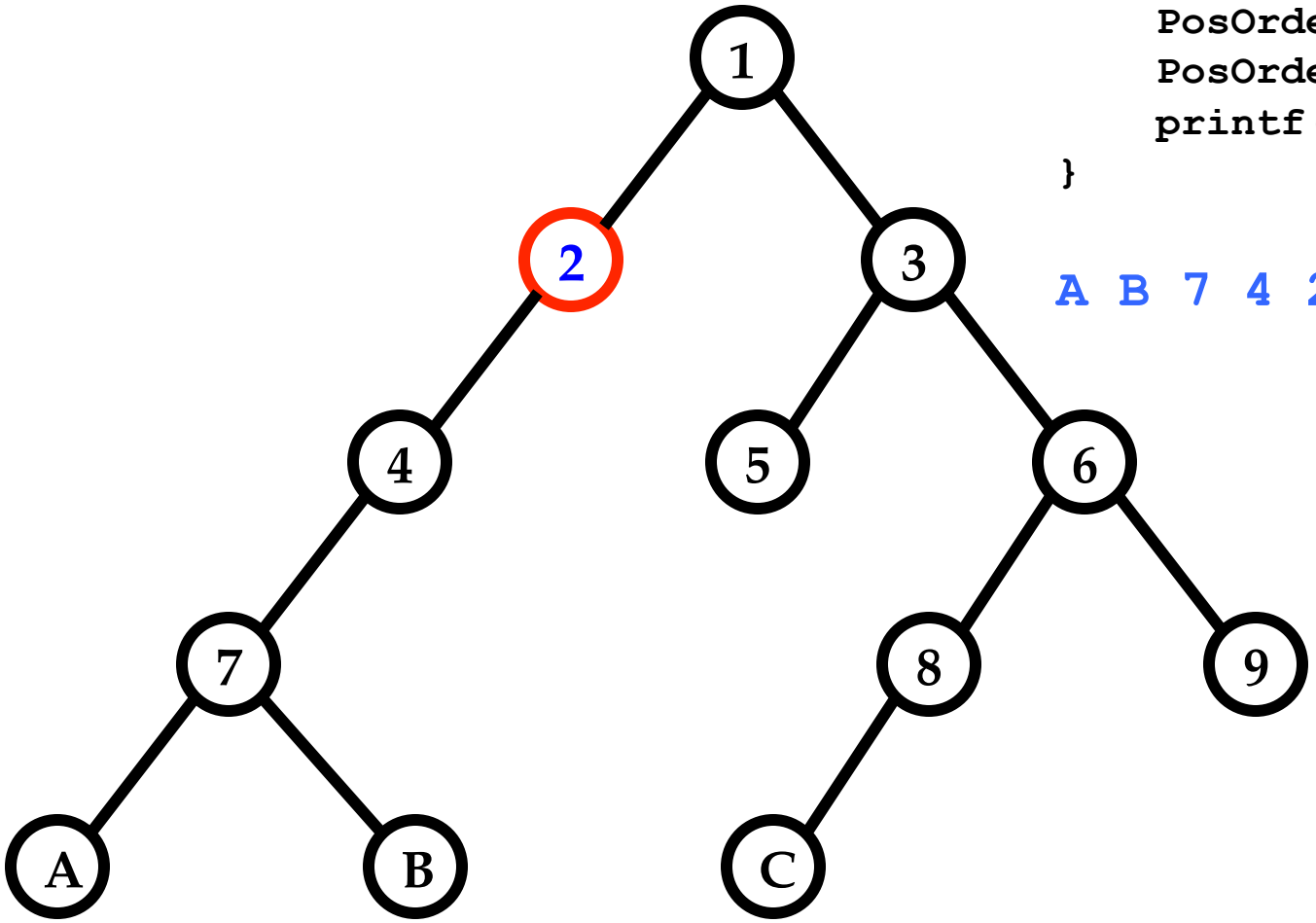
```
void PosOrdem(Apontador p) {
     if (p == null)
          return;
     PosOrdem(p->Esq);
     PosOrdem(p->Dir);
     printf("%d\n", p->Chave);
}
```

A B 7 4



Caminhamento pós-ordem

UFMG COMPUTER SCIENCE

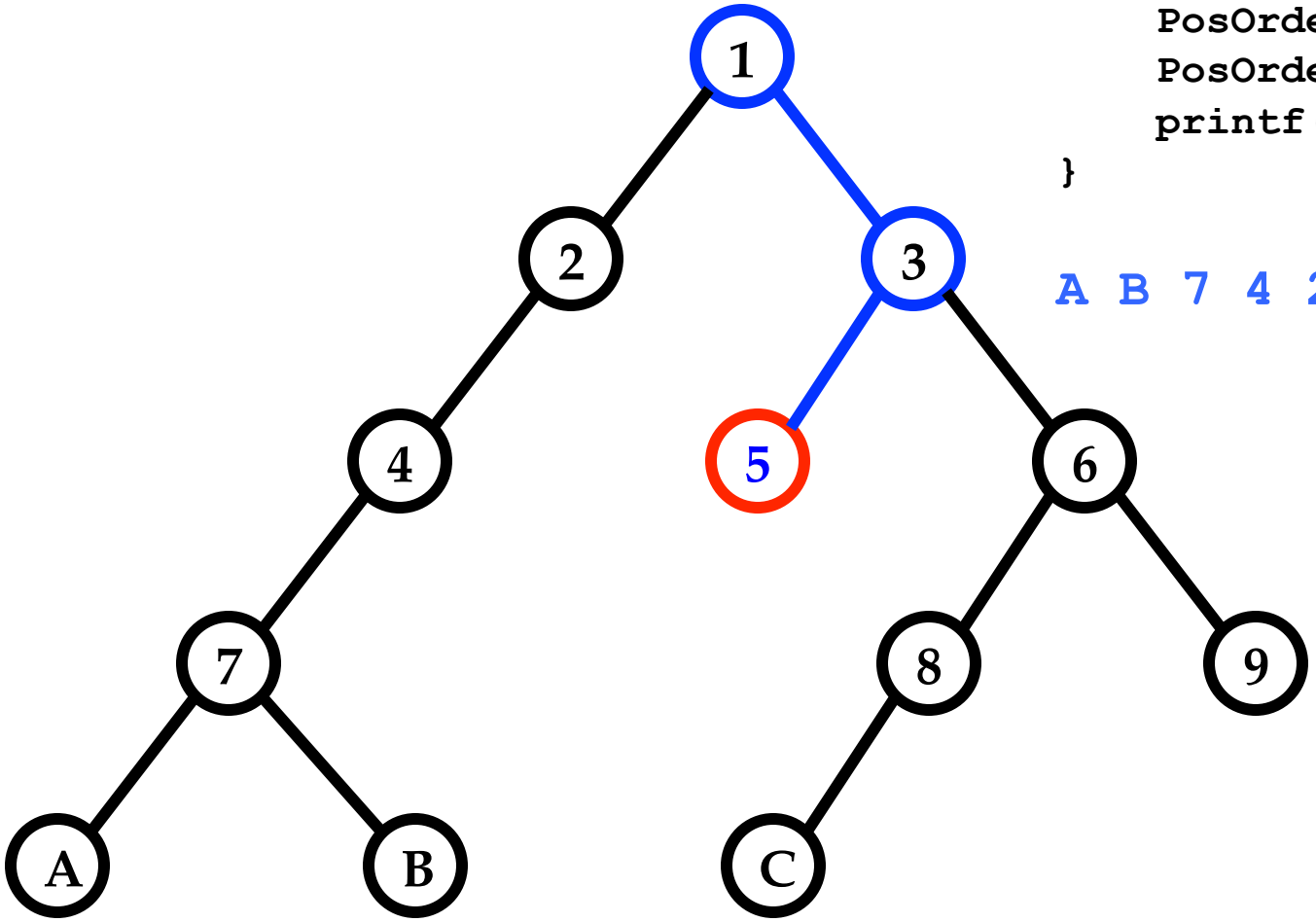



```
void PosOrdem(Apontador p) {
     if (p == null)
          return;
     PosOrdem(p->Esq);
     PosOrdem(p->Dir);
     printf("%d\n", p->Chave);
}
```

A B 7 4 2

Caminhamento pós-ordem

UFMG COMPUTER SCIENCE

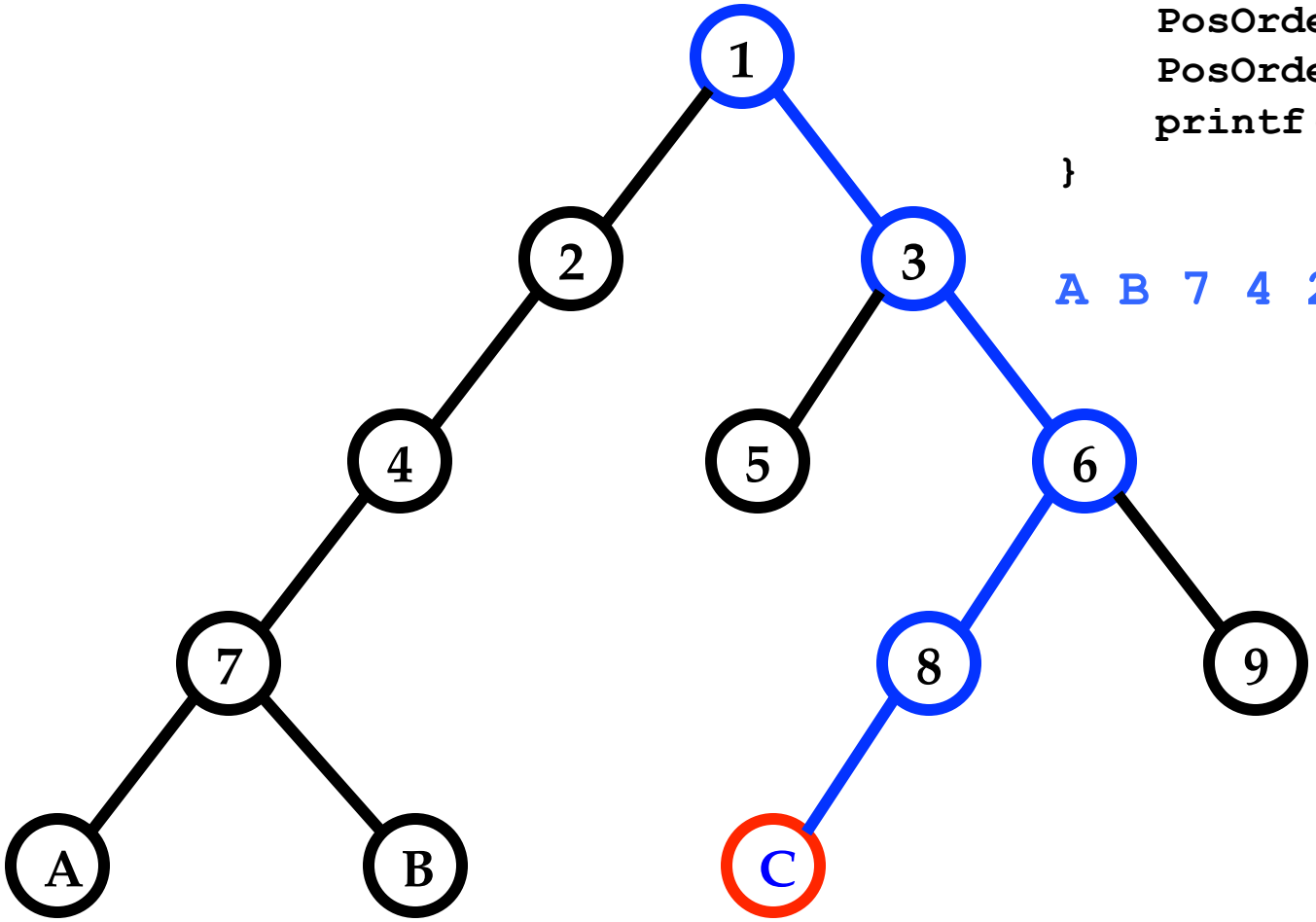



```
void PosOrdem(Apontador p) {
     if (p == null)
          return;
     PosOrdem(p->Esq);
     PosOrdem(p->Dir);
     printf("%d\n", p->Chave);
}
```

A B 7 4 2 5

Caminhamento pós-ordem

UFMG COMPUTER SCIENCE

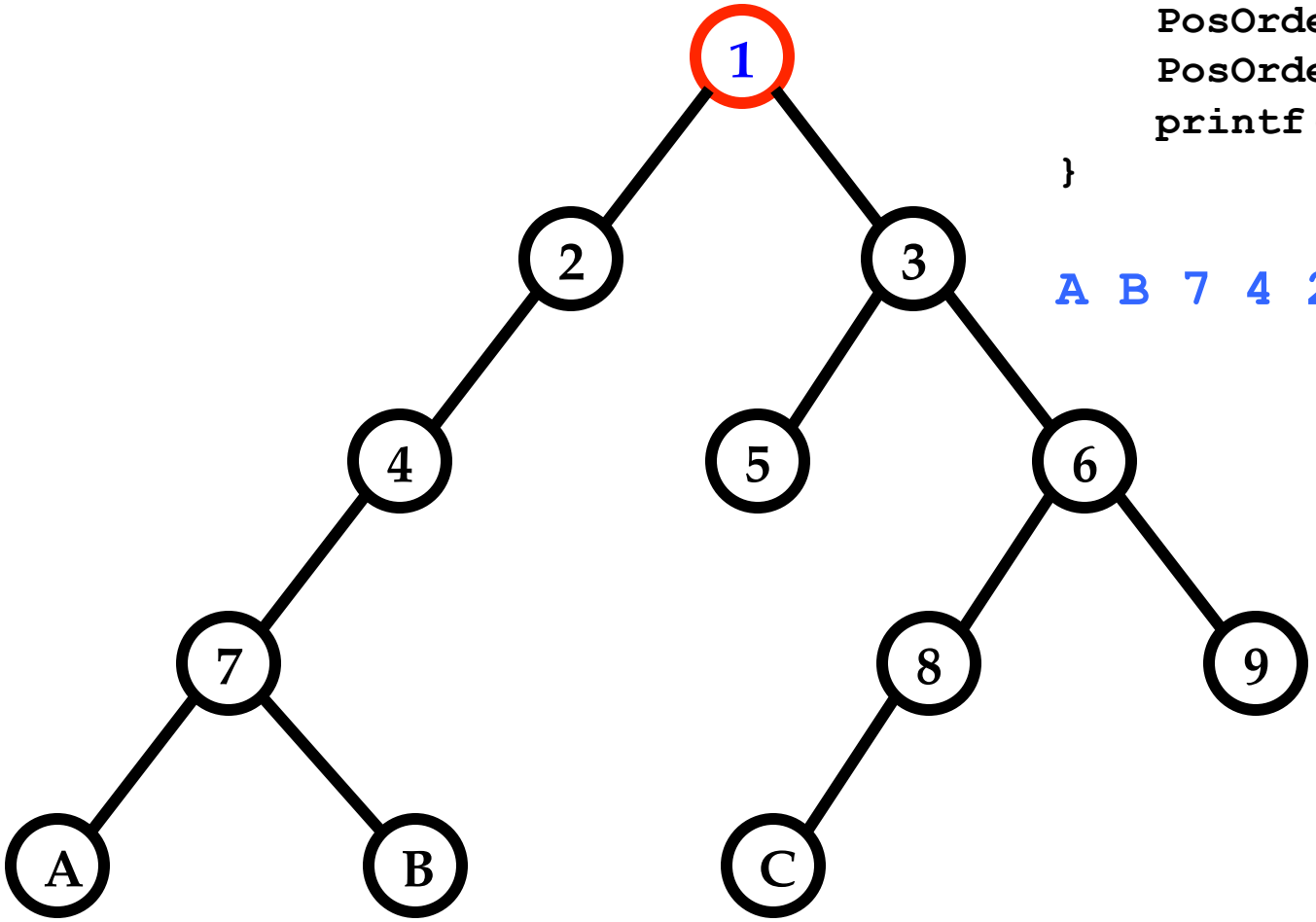



```
void PosOrdem(Apontador p) {
     if (p == null)
          return;
     PosOrdem(p->Esq);
     PosOrdem(p->Dir);
     printf("%d\n", p->Chave);
}
```

A B 7 4 2 5 C

Caminhamento pós-ordem

UFMG COMPUTER SCIENCE




```
void PosOrdem(Apontador p) {
     if (p == null)
          return;
     PosOrdem(p->Esq);
     PosOrdem(p->Dir);
     printf("%d\n", p->Chave);
}
```

A B 7 4 2 5 C 8 9 6 3 1