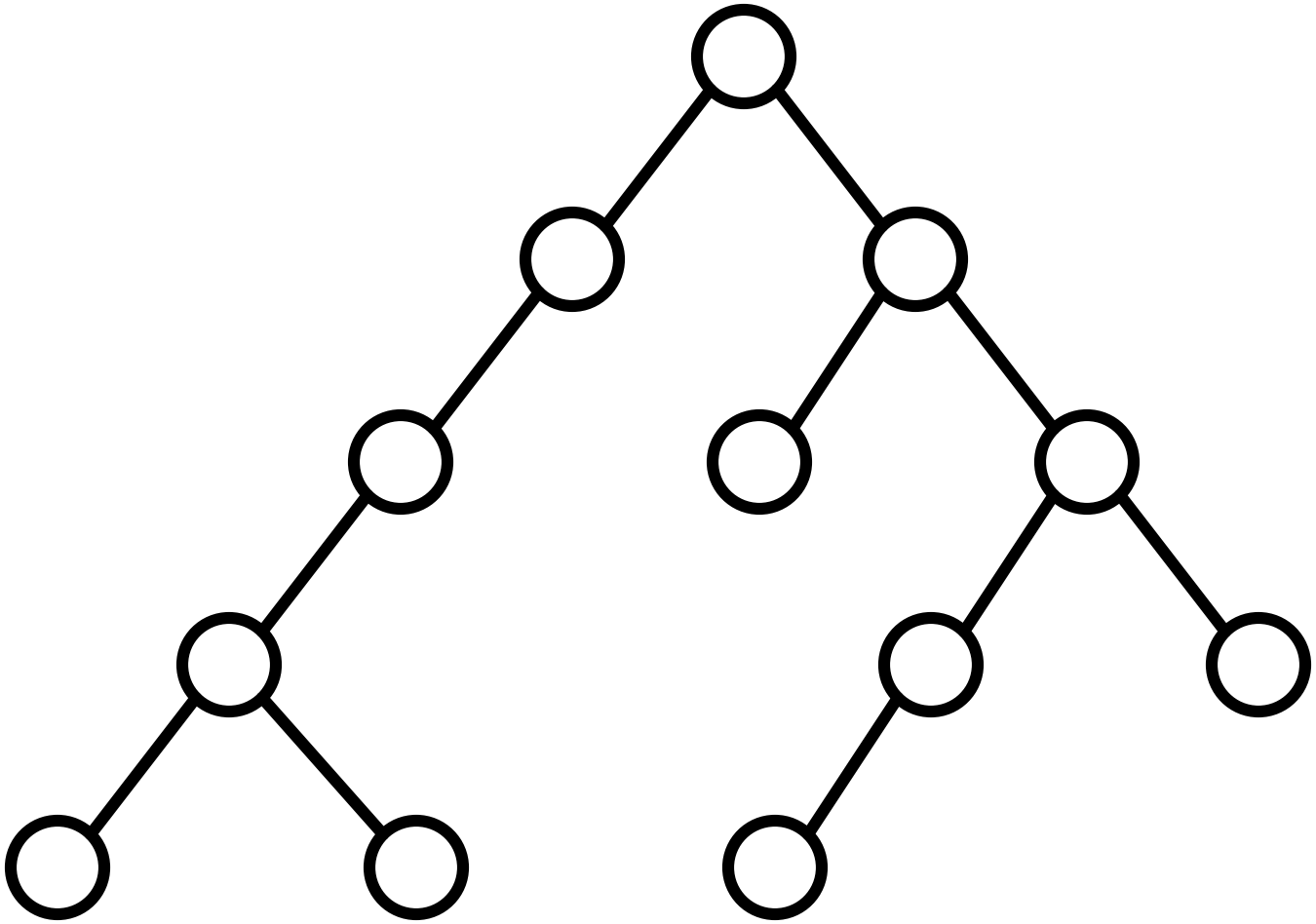
Árvores

Árvores

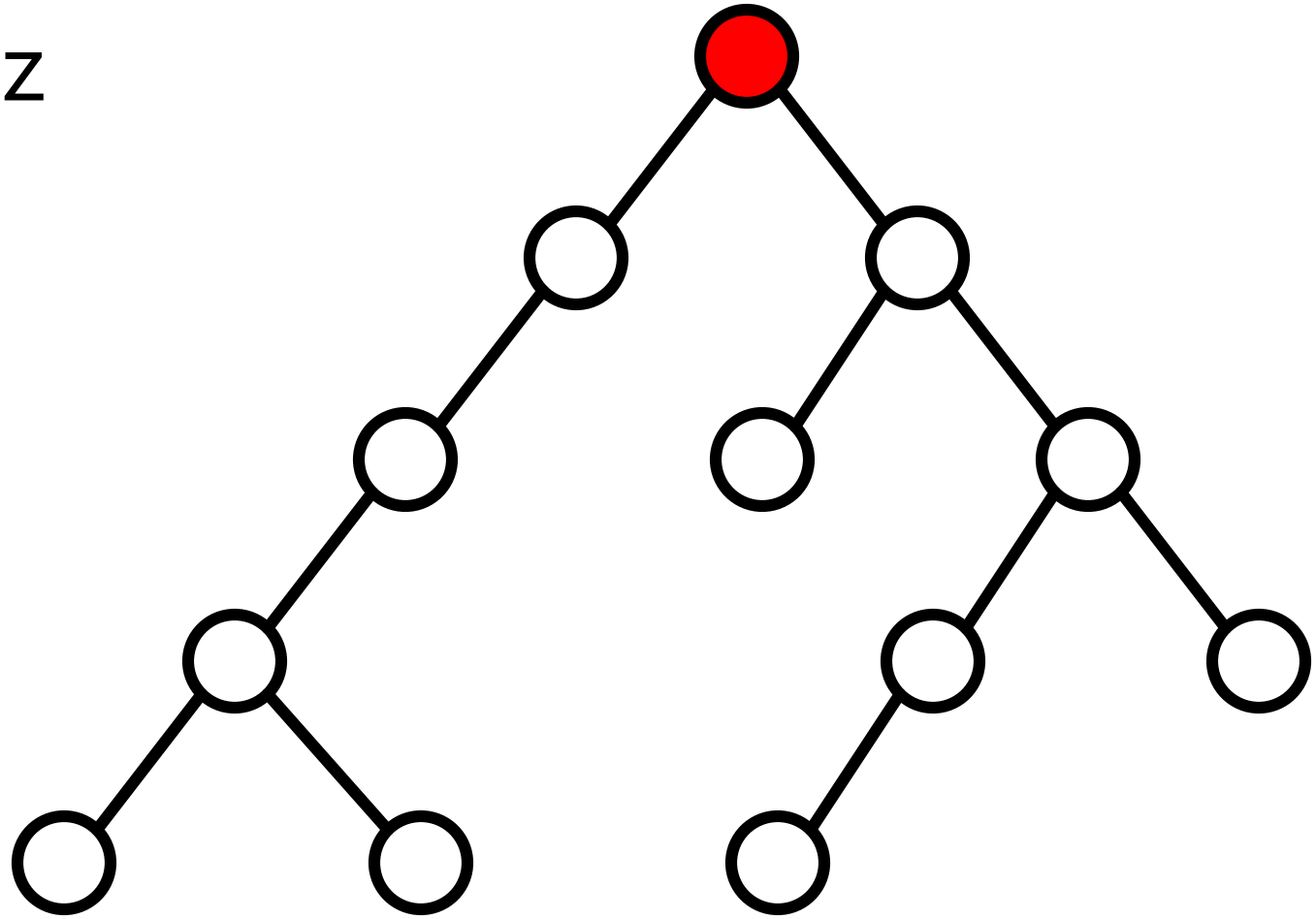
- Organizam dados de forma hierárquica
- Acontecem com frequência na natureza
- Fáceis de representar e manipular com computadores
- Úteis para várias tarefas

Terminologia



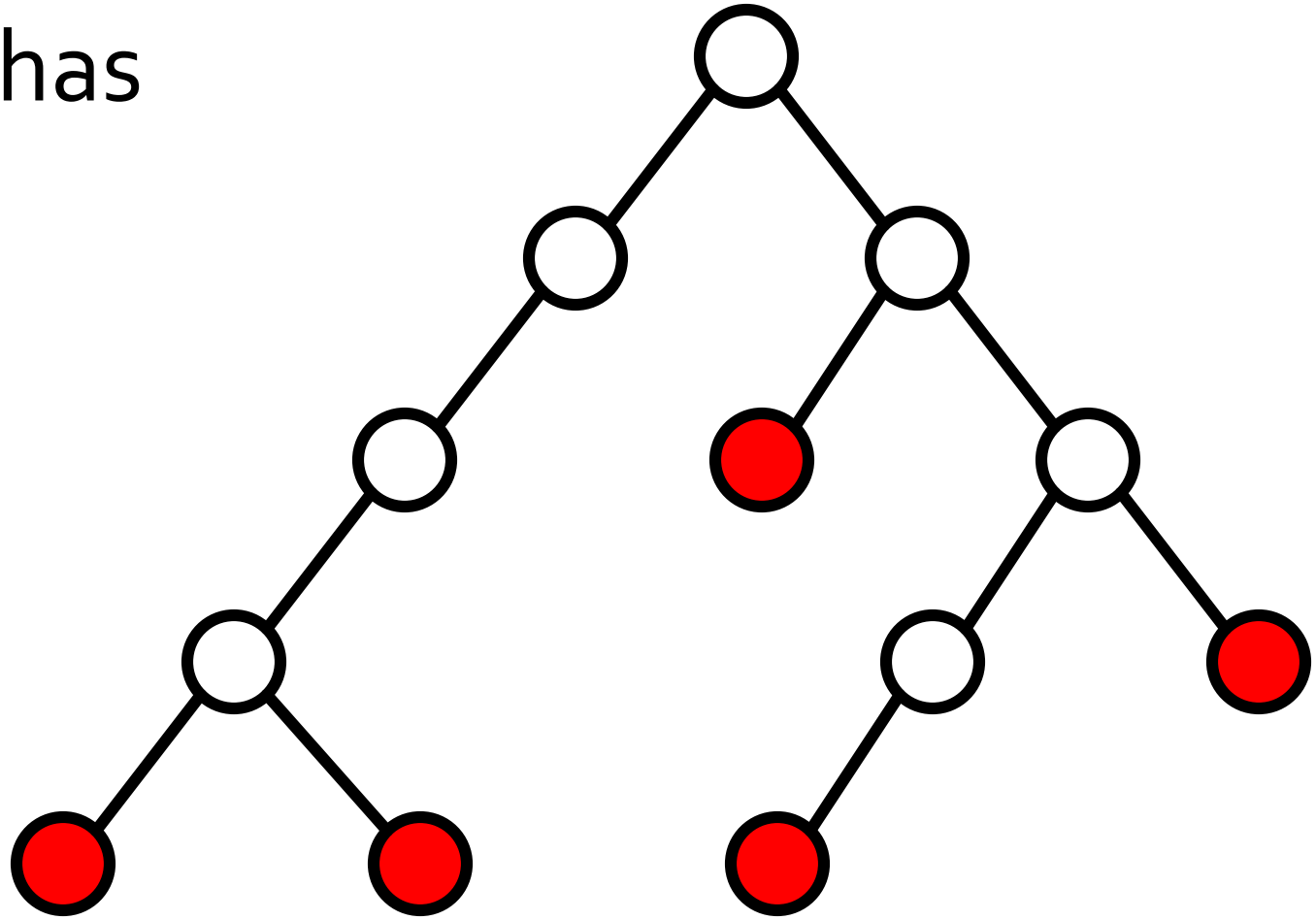
Terminologia

Raiz



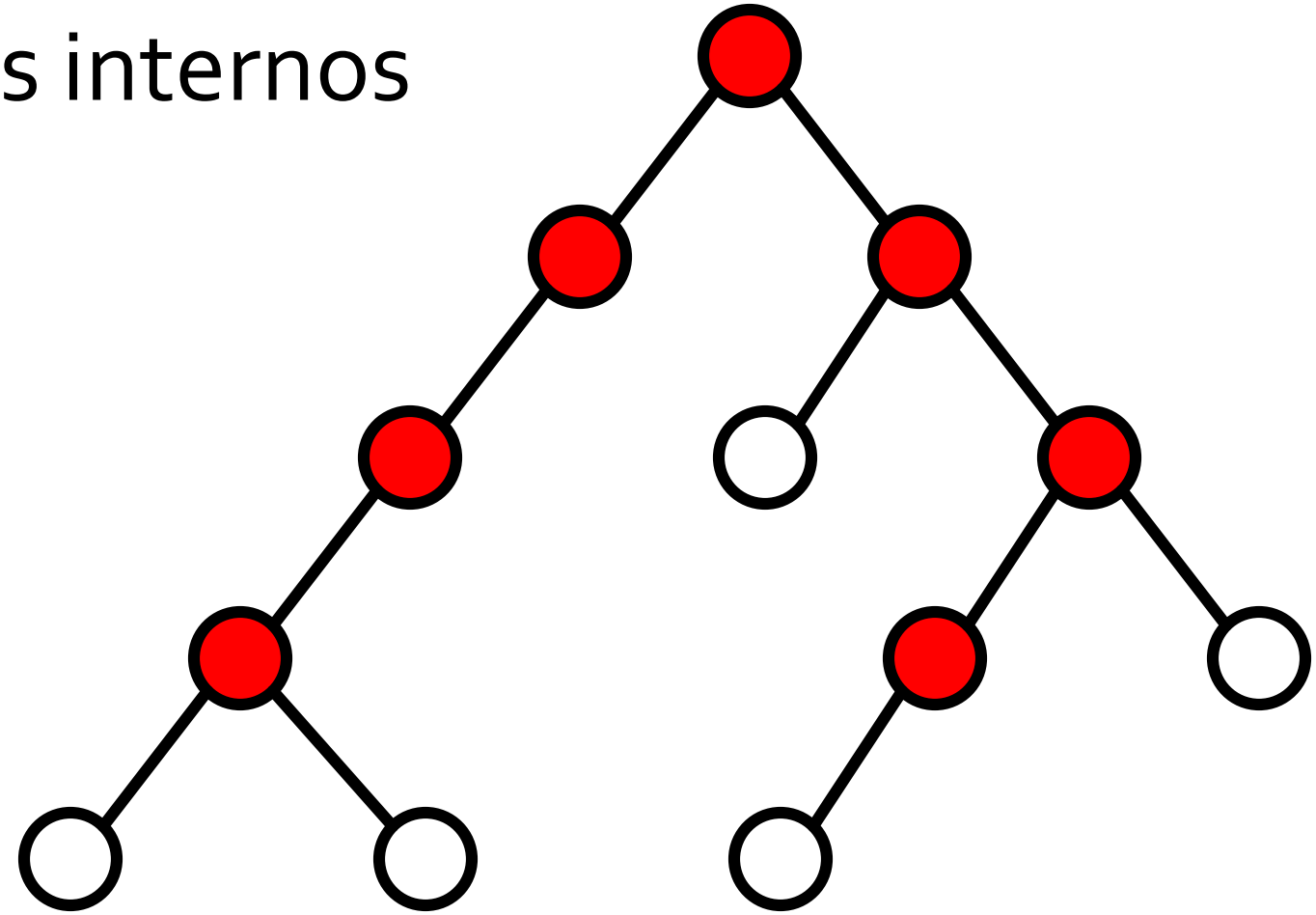
Terminologia

Folhas



Terminologia

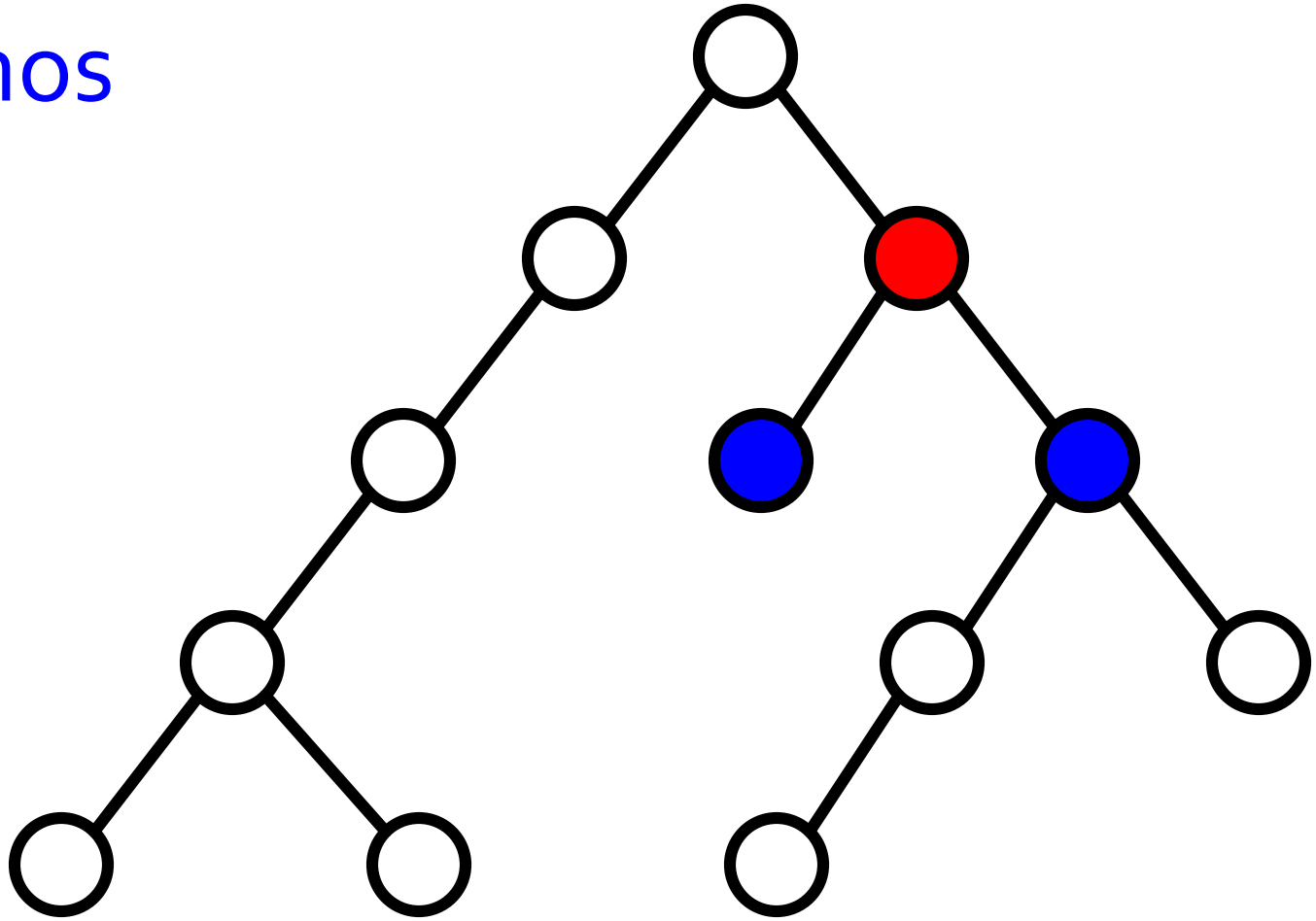
Nós internos



Terminologia

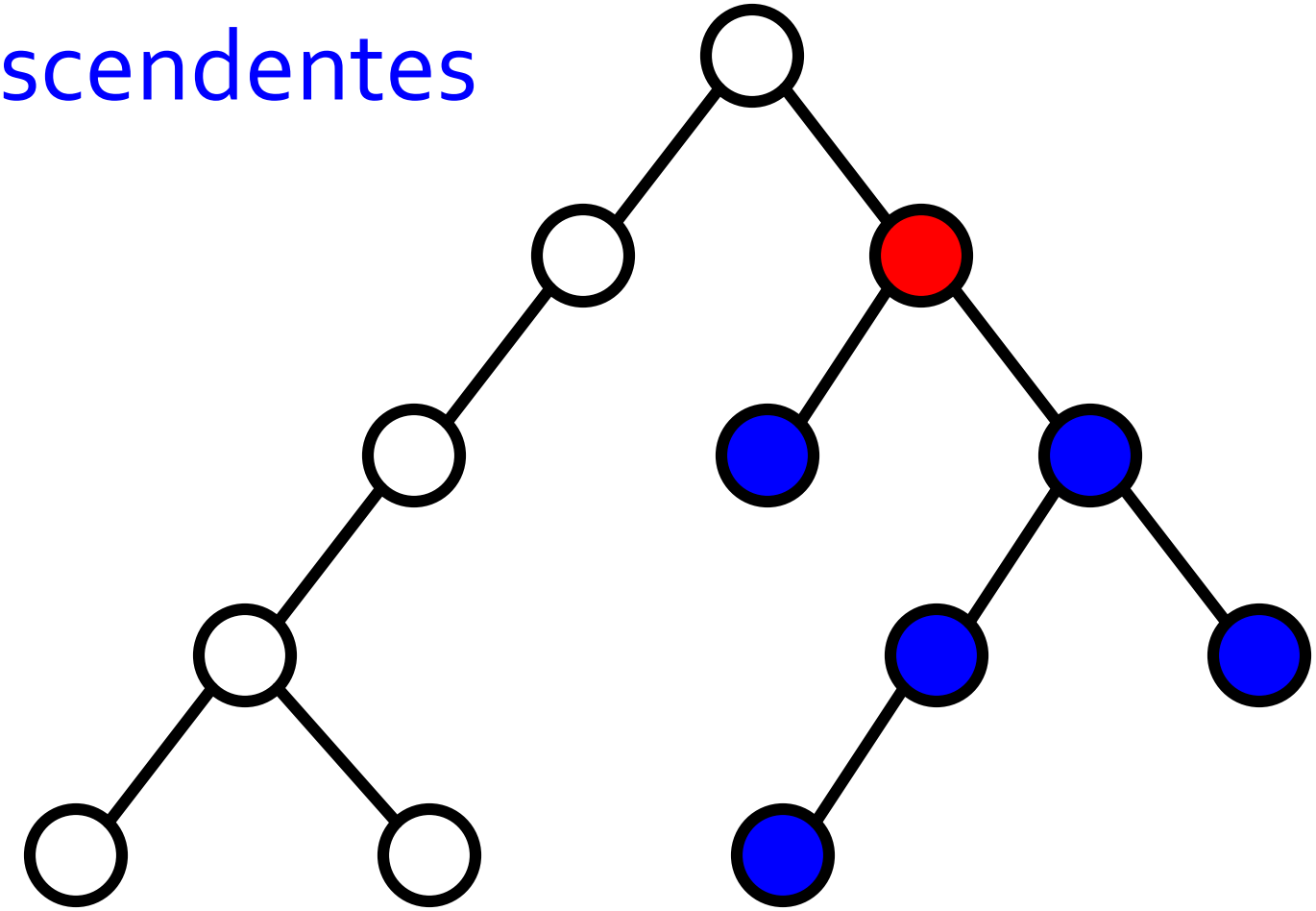
Filhos

Pai



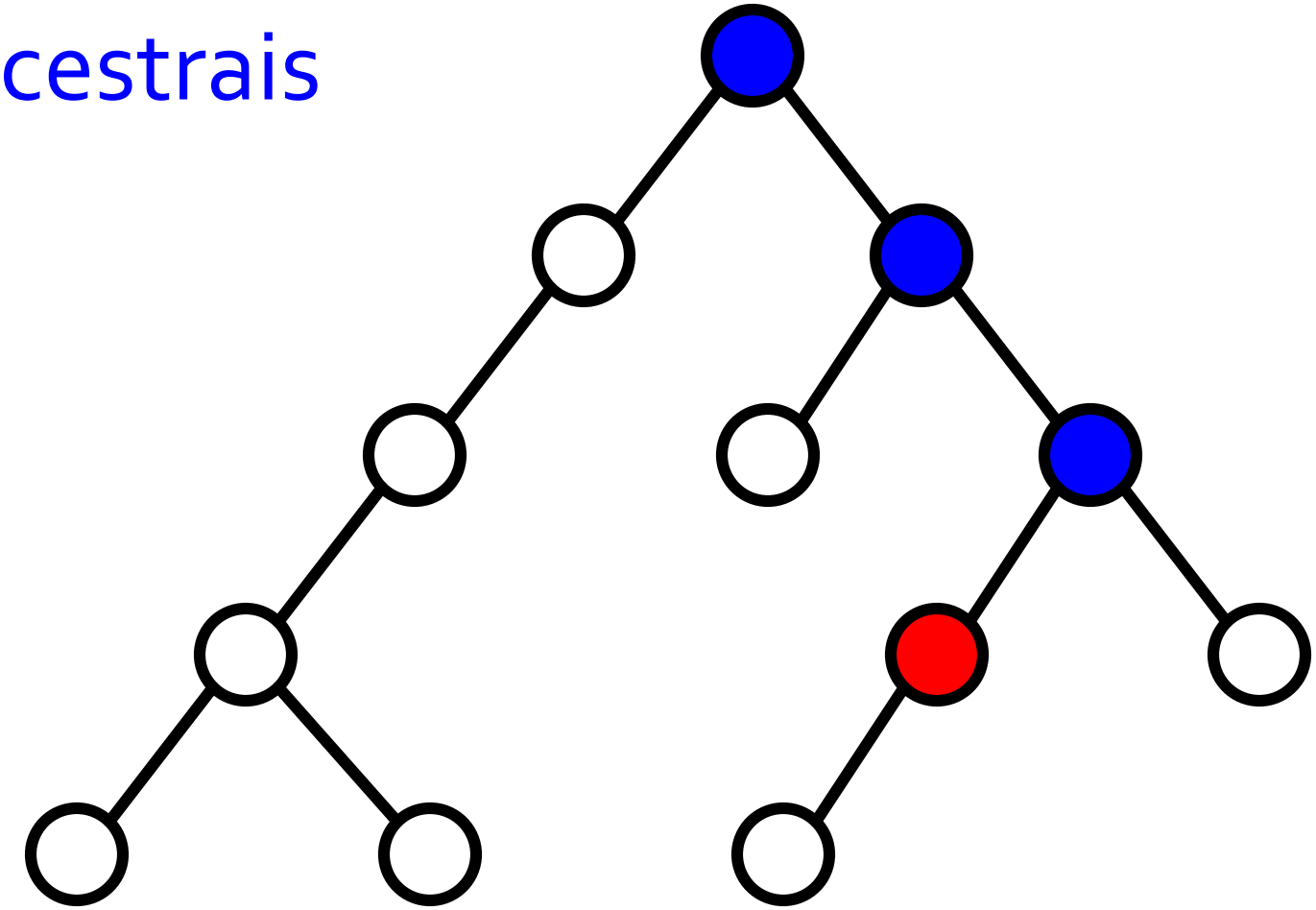
Terminologia

Descendentes



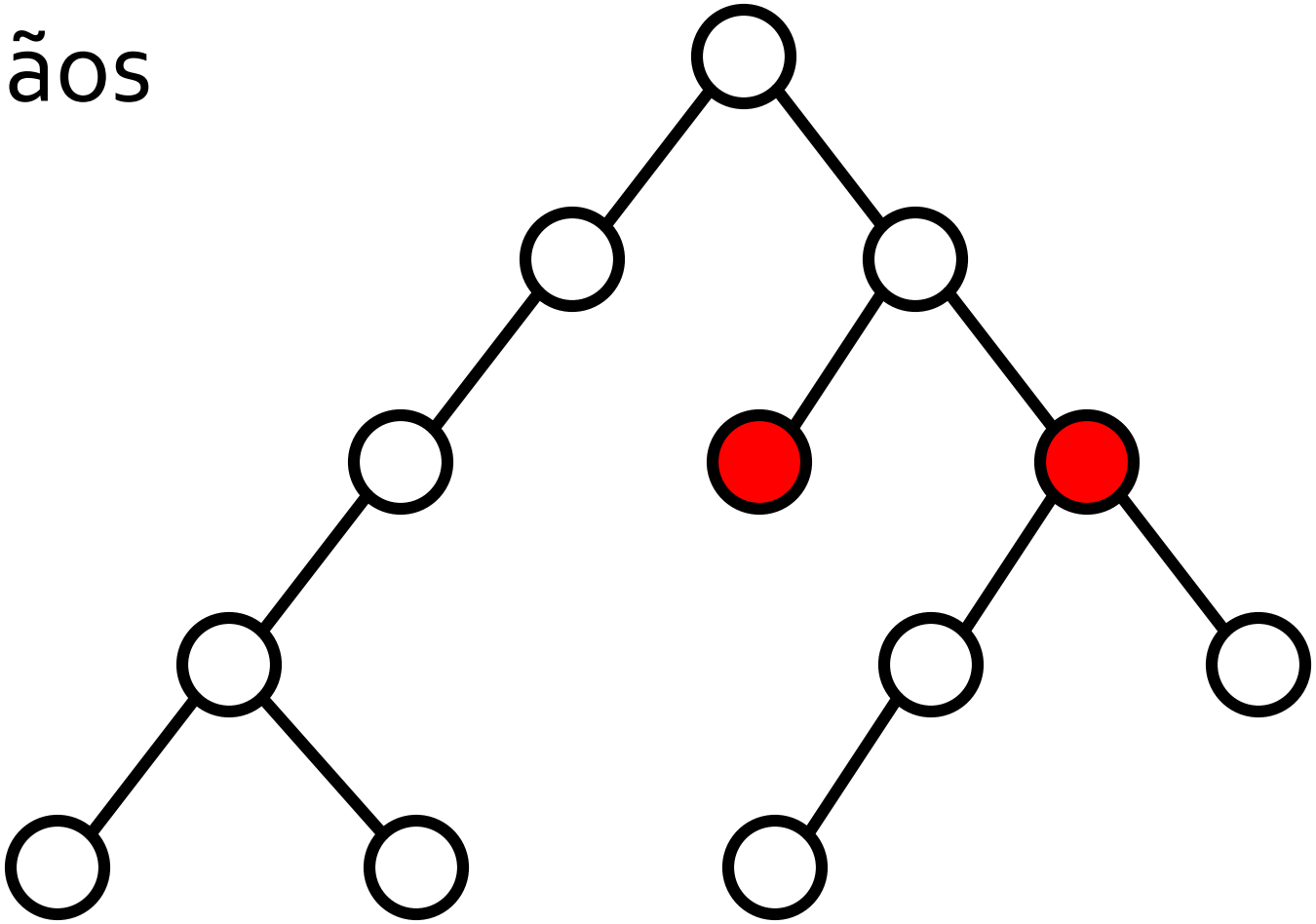
Terminologia

Ancestrais



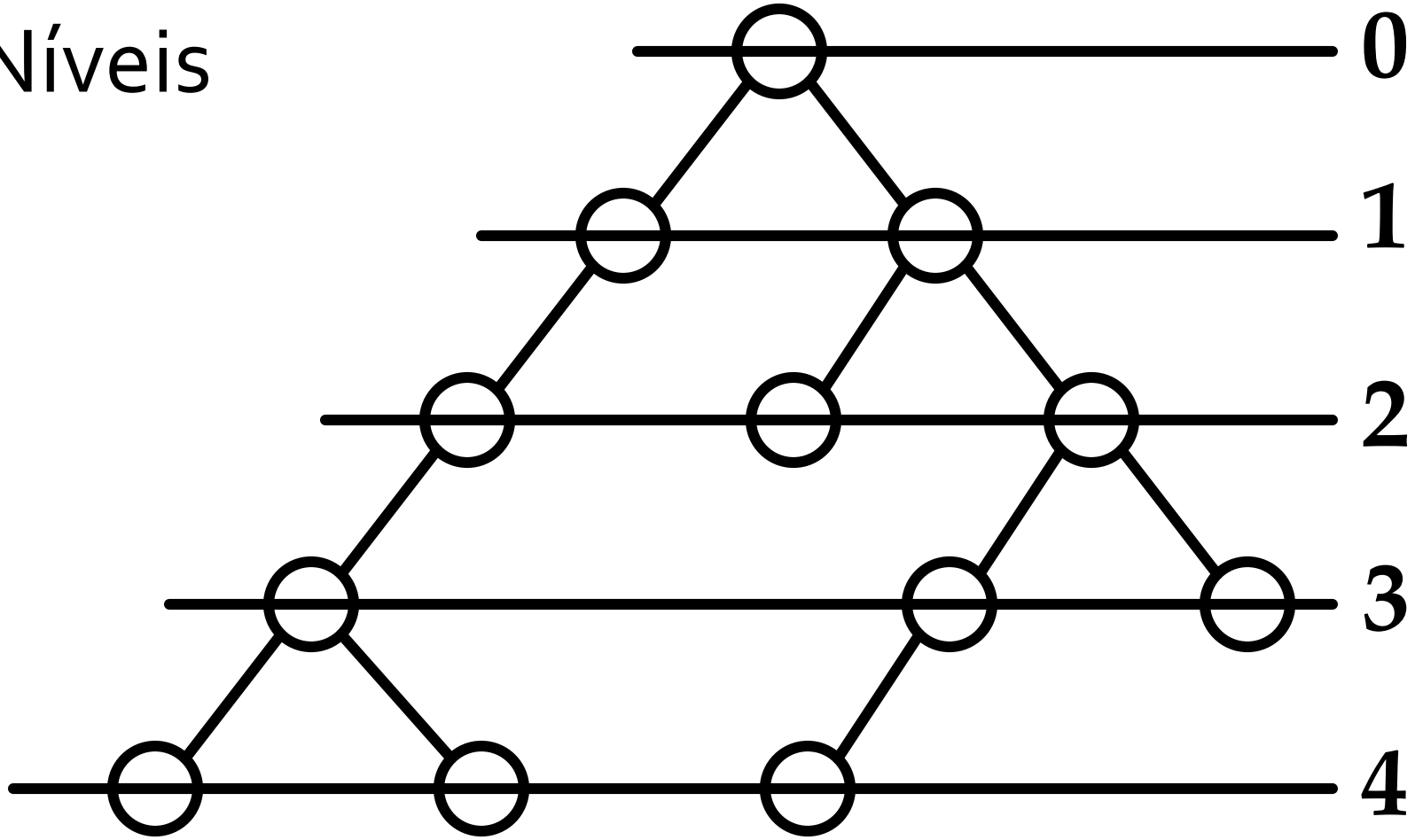
Terminologia

Irmãos



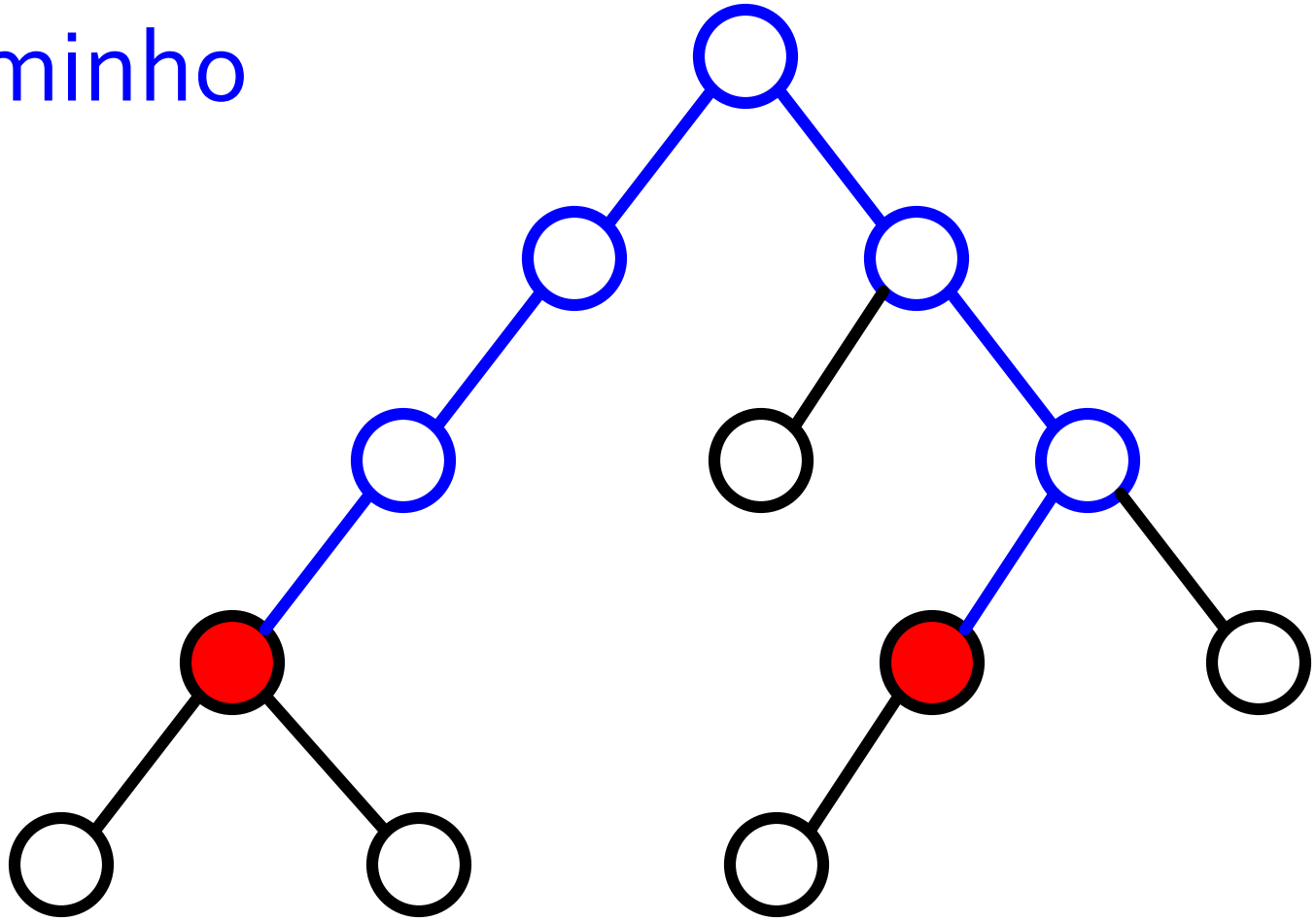
Terminologia

Níveis



Terminologia

Caminho



Propriedades

- Árvores não contém ciclos
  - Só existe **um** caminho entre qualquer par de nós
- Uma árvore com n nós tem altura pelo menos $\lg(n)$ e no máximo n-1

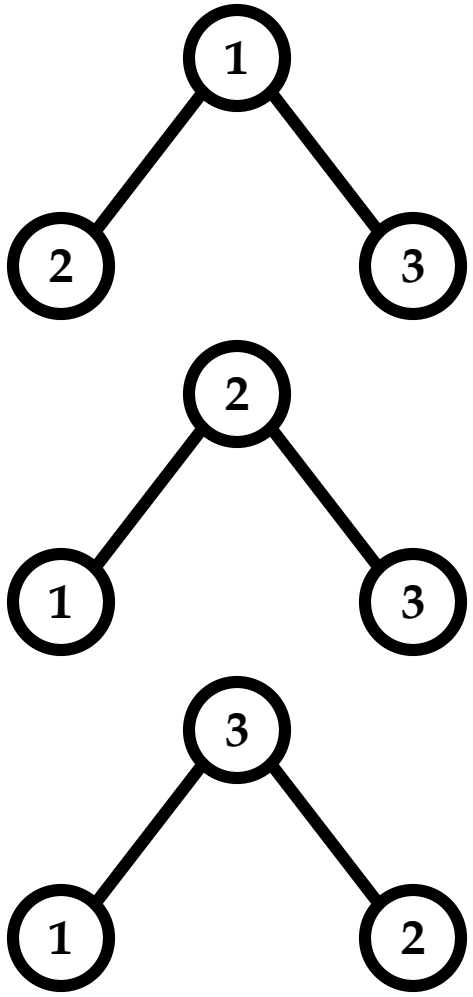
Definição: árvore binária

- Uma árvore binária é uma árvore com zero, um ou dois filhos onde cada filho é também uma árvore binária
  - Definição é recursiva
  - Veremos que manipulação de árvores é fácil usando recursão

```
struct arvore {
    struct arvore *esq;
    struct arvore *dir;
    int dado;
};
```

Caminhamento de árvore

- Pré-ordem
- Ordem central
- Pós-ordem



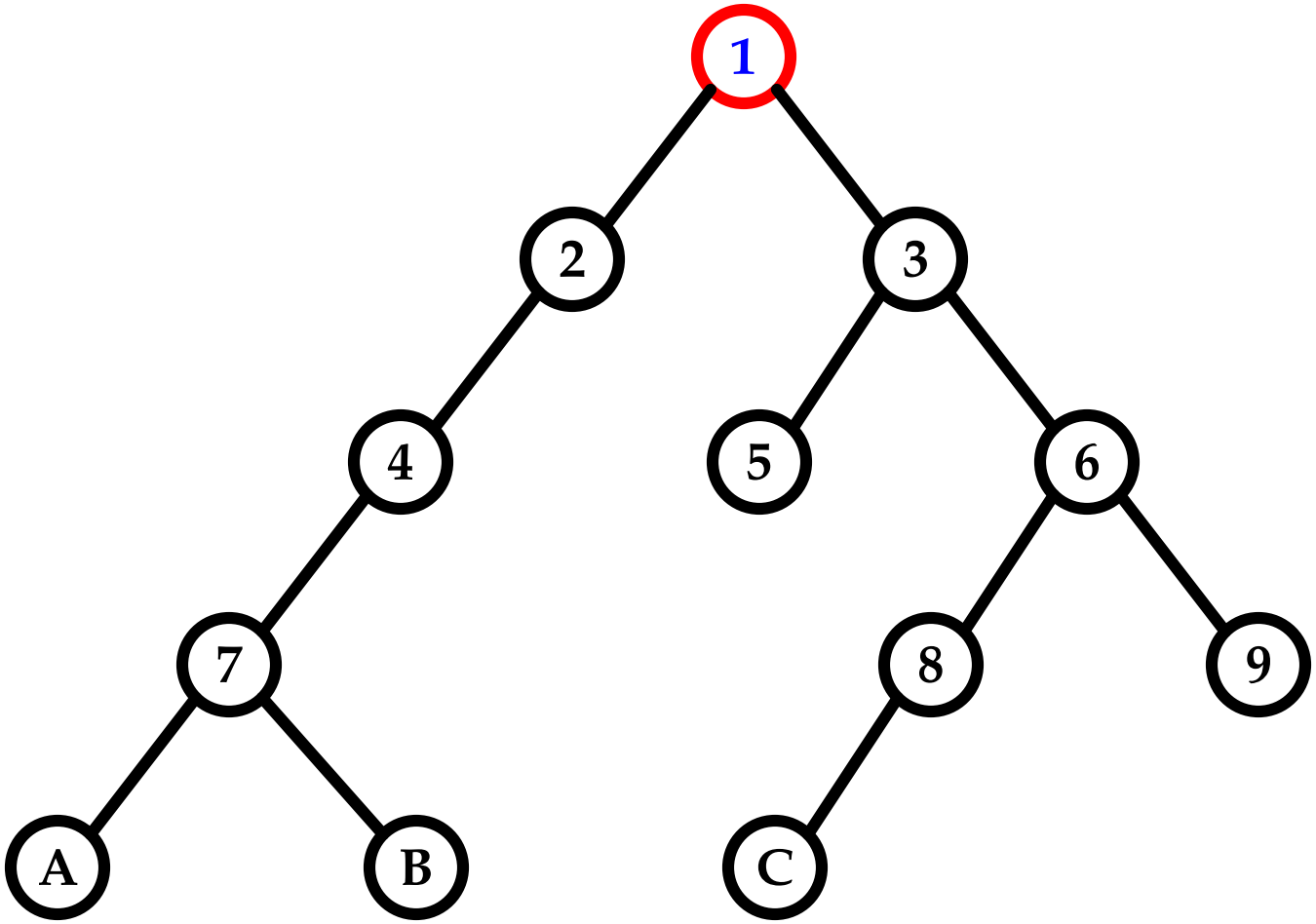
Caminhamento pré-ordem

```
void pre_ordem(struct arvore *f)
{

    printf("%d", f->dado);
    pre_ordem(f->esq);
    pre_ordem(f->dir);
}
```

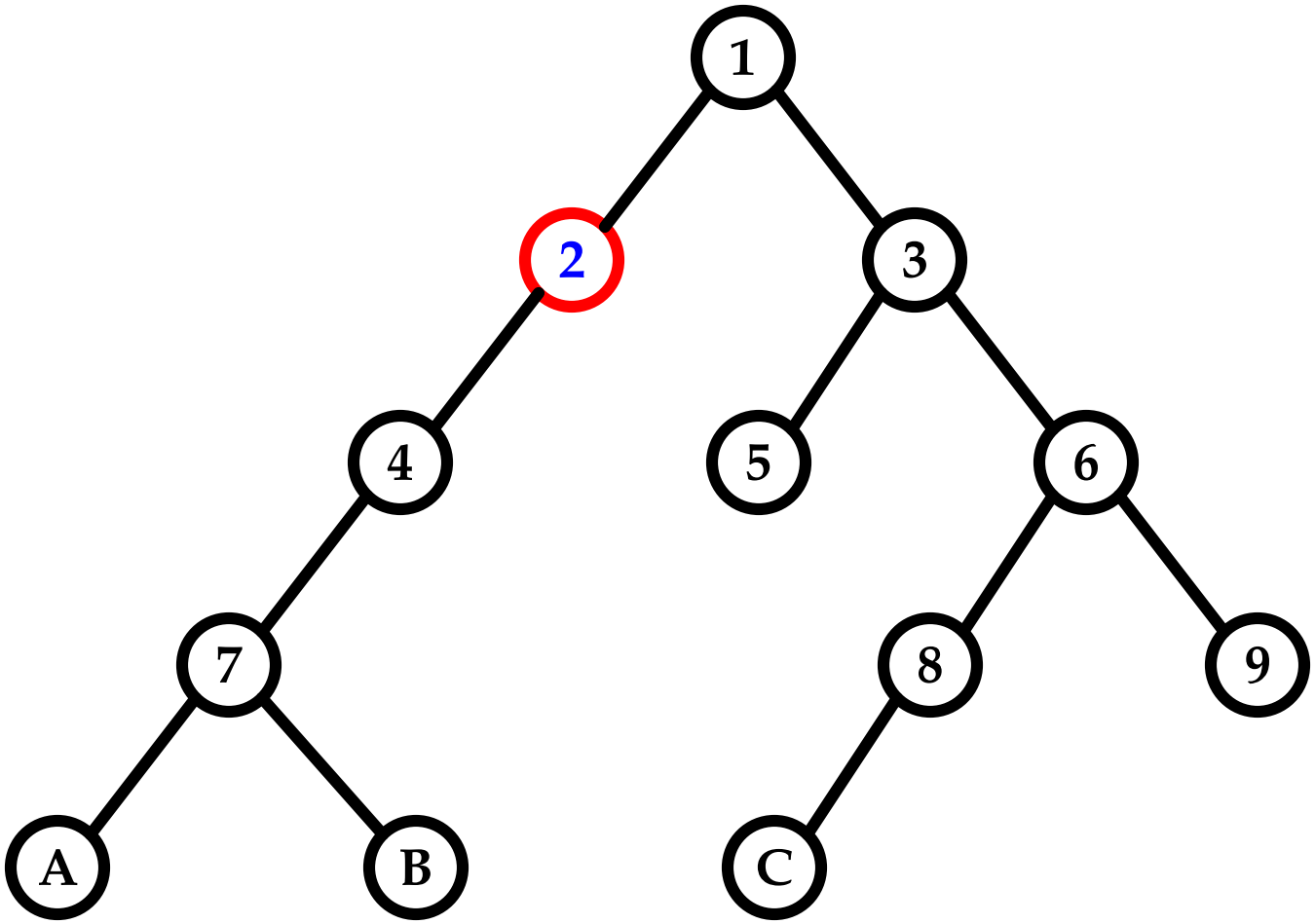
Caminhamento pré-ordem

1



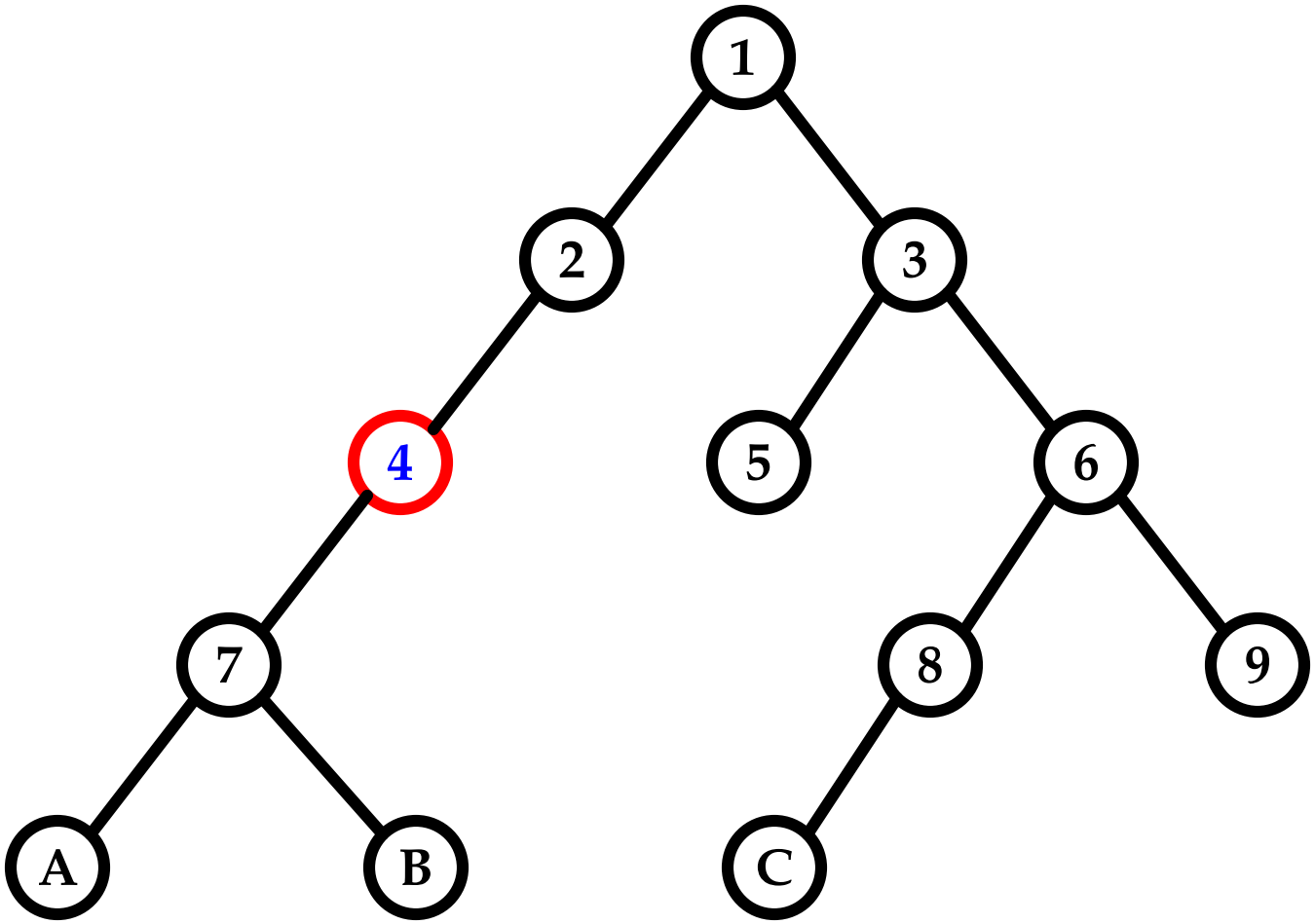
Caminhamento pré-ordem

1 2



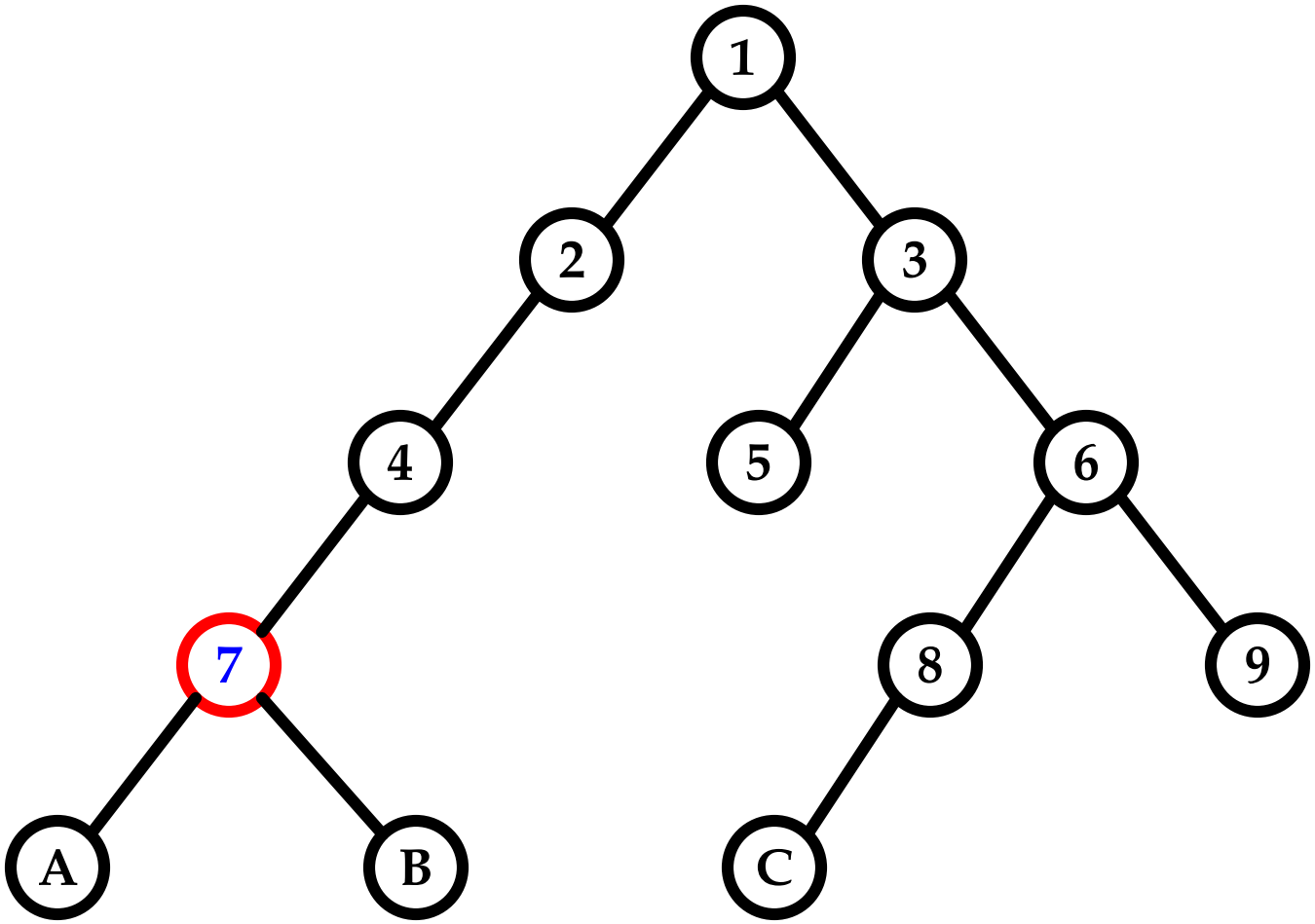
Caminhamento pré-ordem

1 2 4



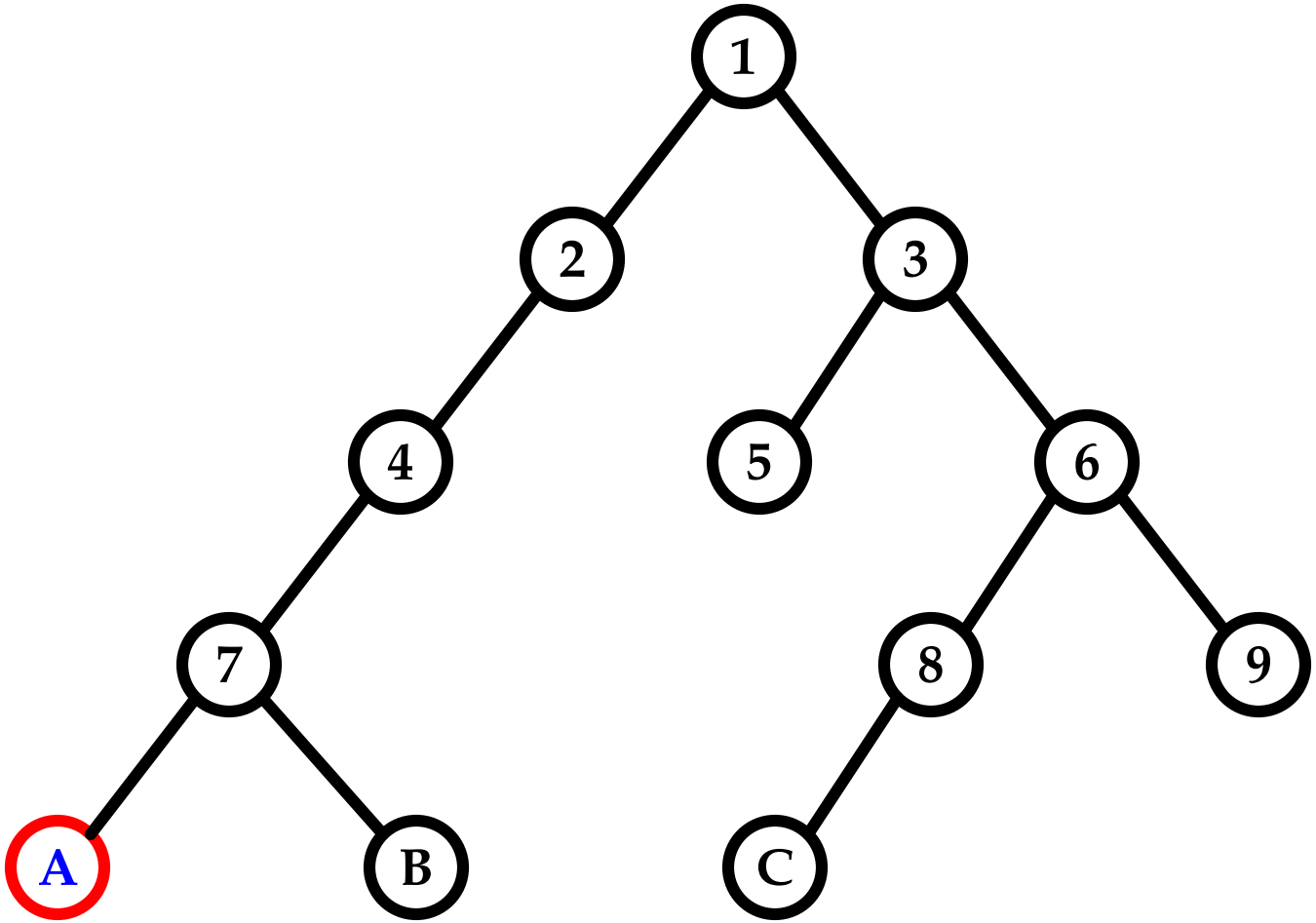
Caminhamento pré-ordem

1 2 4 7



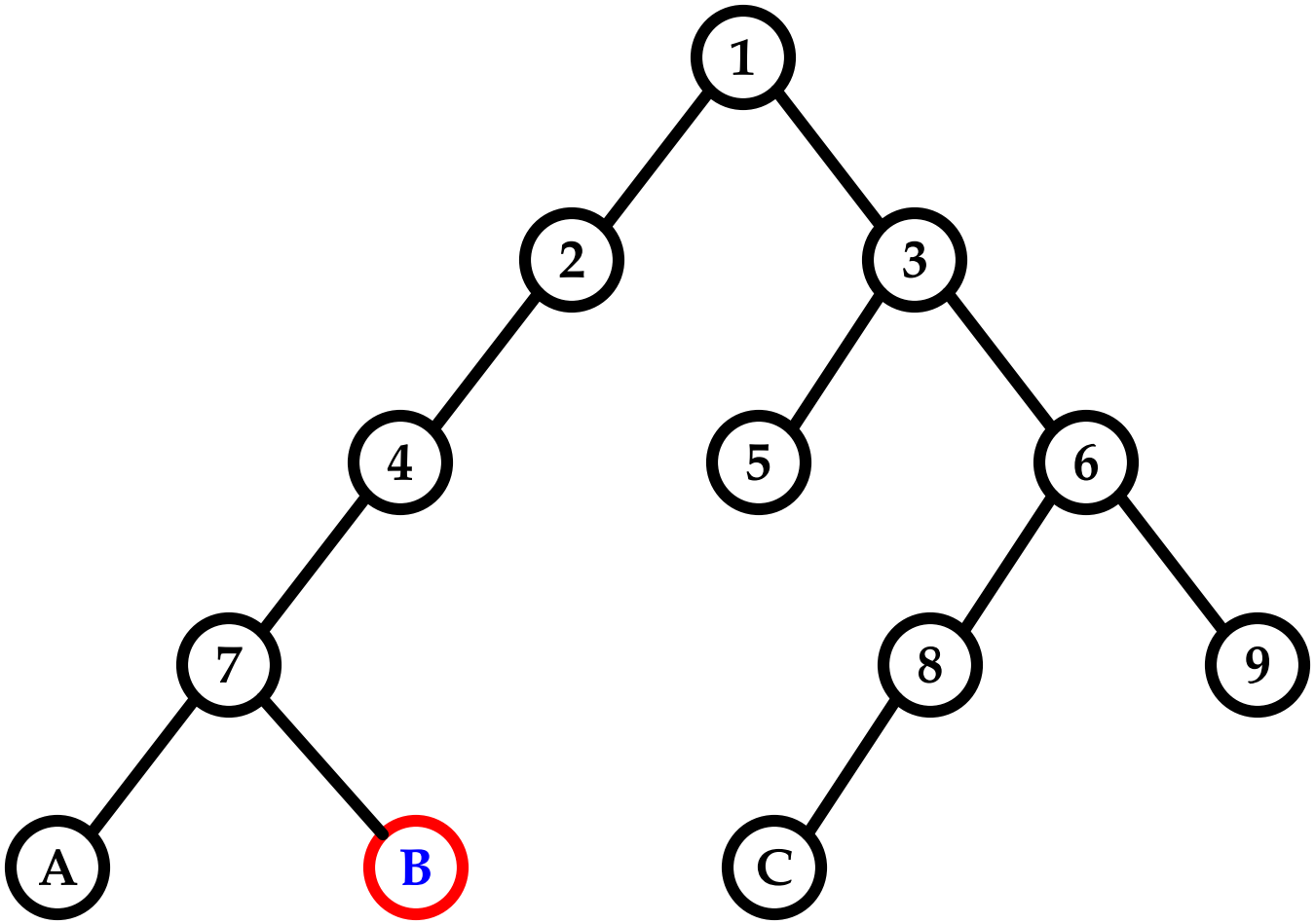
Caminhamento pré-ordem

1 2 4 7 A



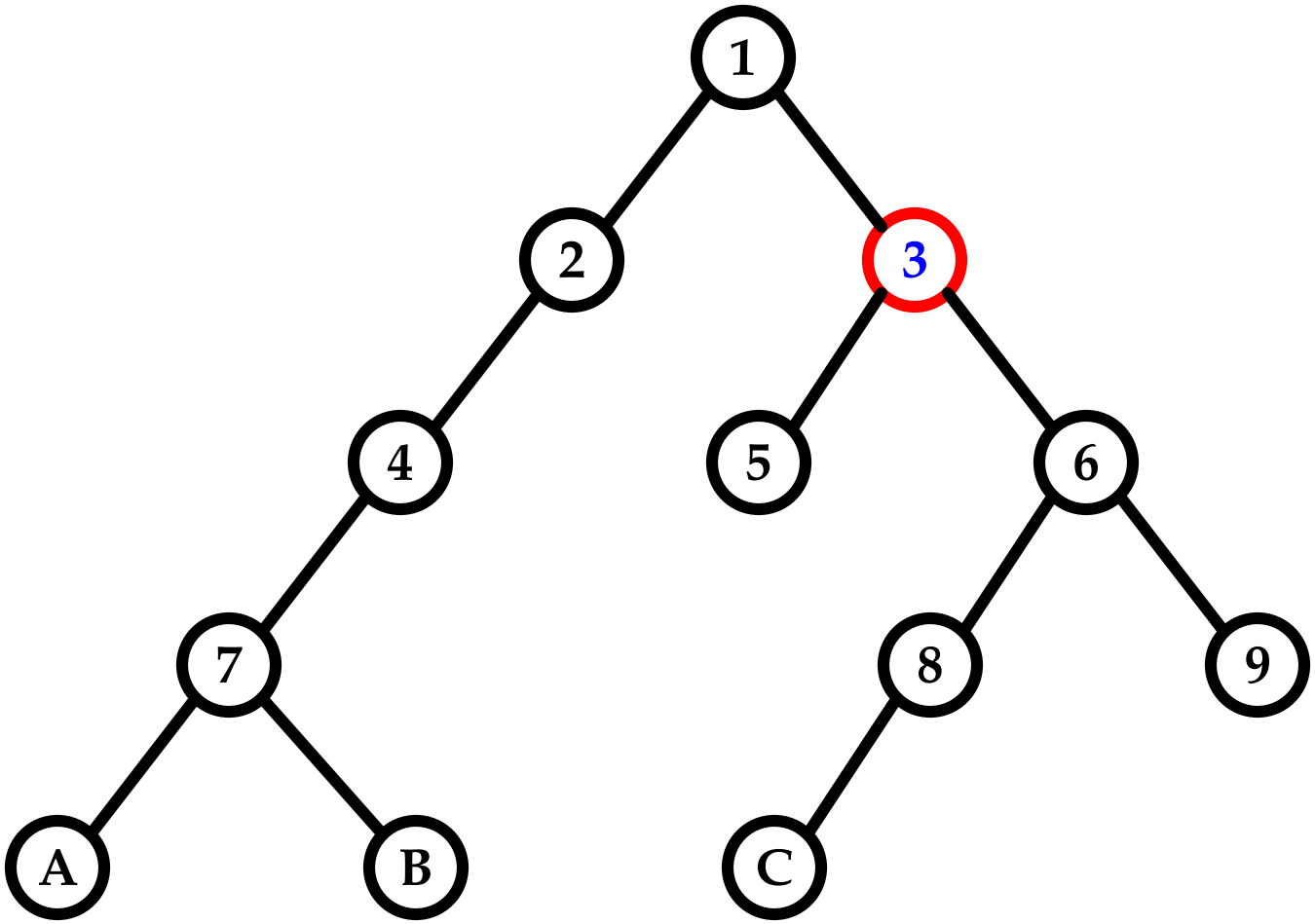
Caminhamento pré-ordem

1 2 4 7 A B



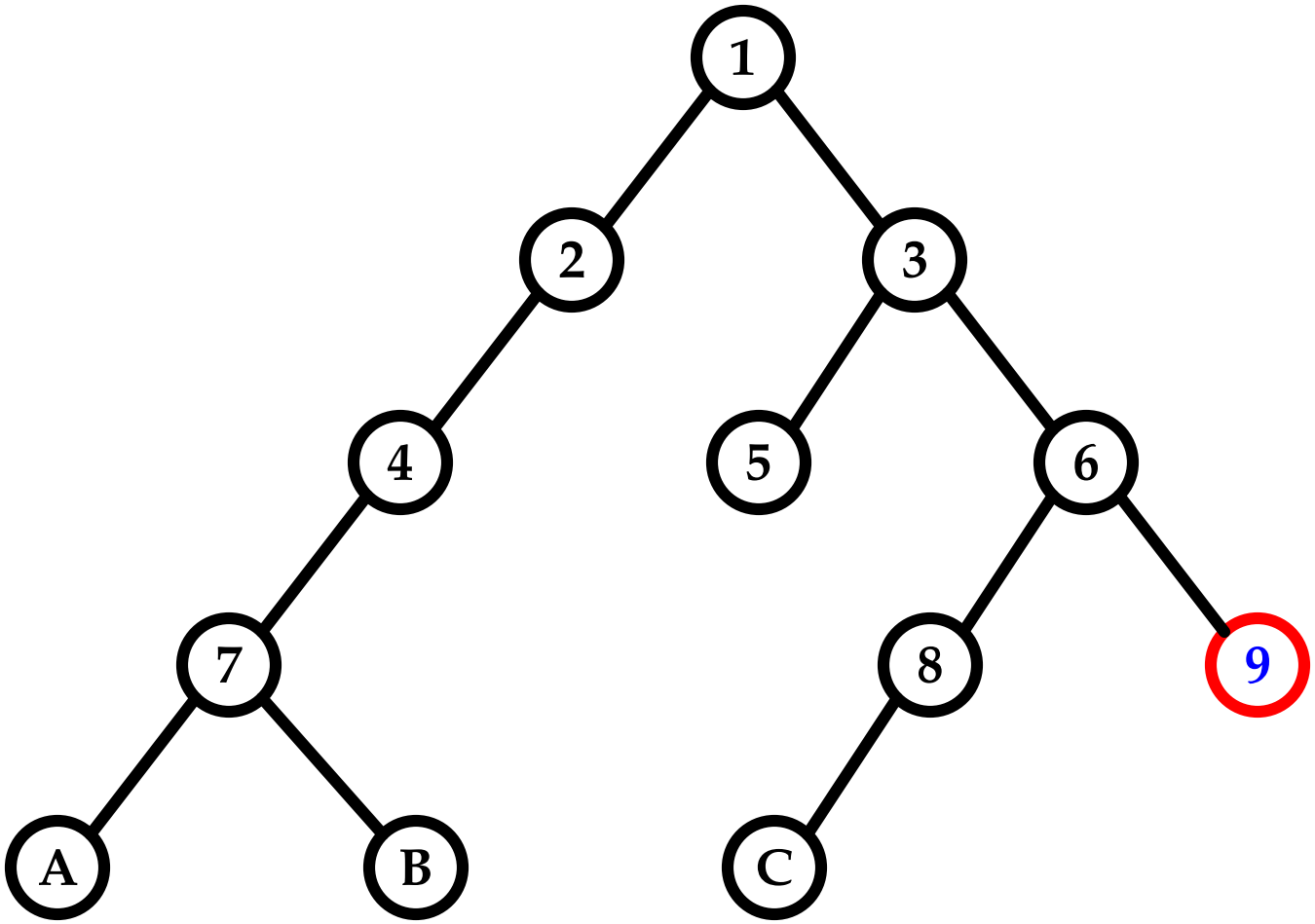
Caminhamento pré-ordem

1 2 4 7 A B 3



Caminhamento pré-ordem

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

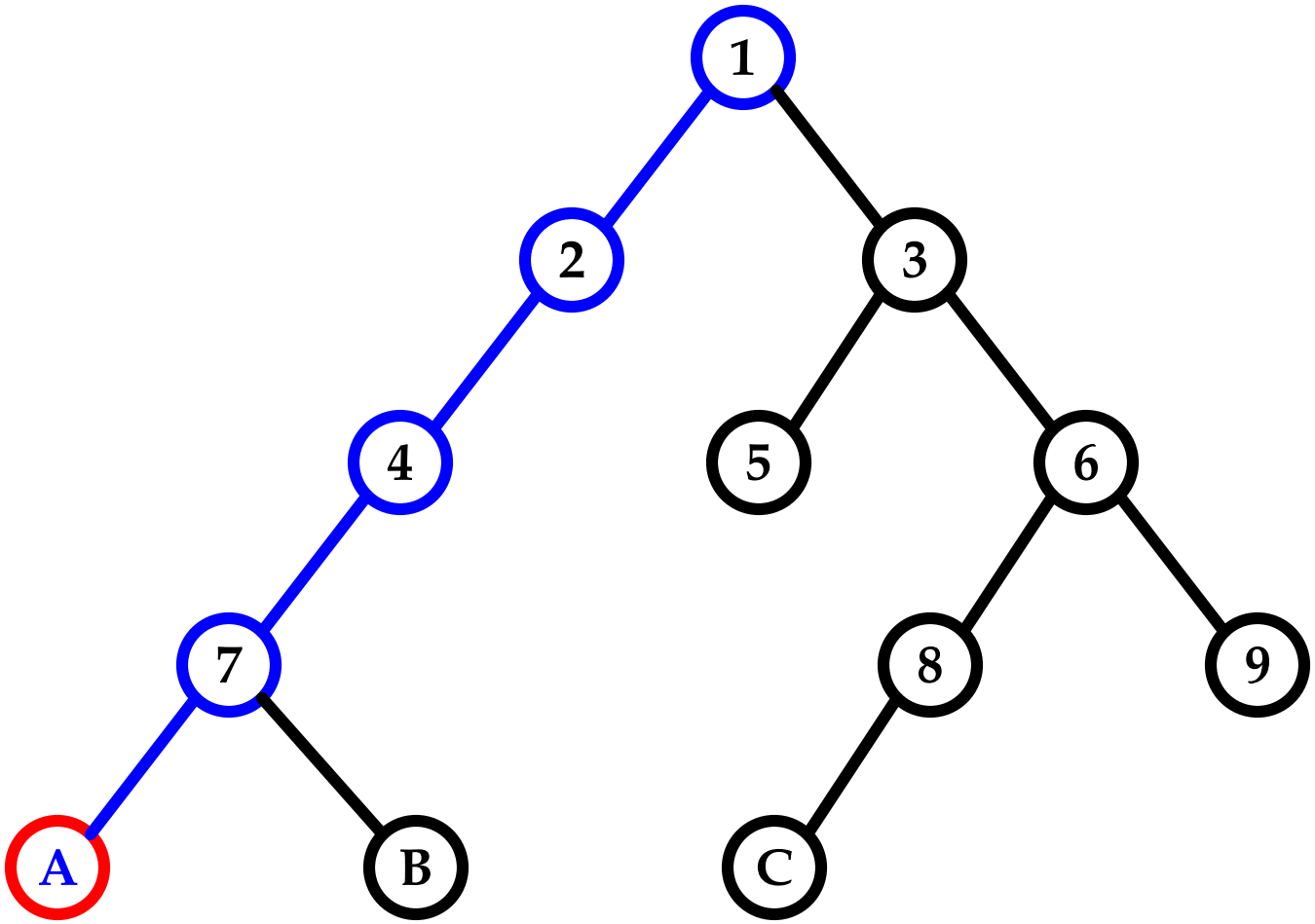


Caminhamento central e pós-ordem

```
void ordem_central(struct arvore *f)
{
    if(f == NULL) { return; }
    ordem_central(f->esq);
    printf("%d", f->dado);
    ordem_central(f->dir);
}
```

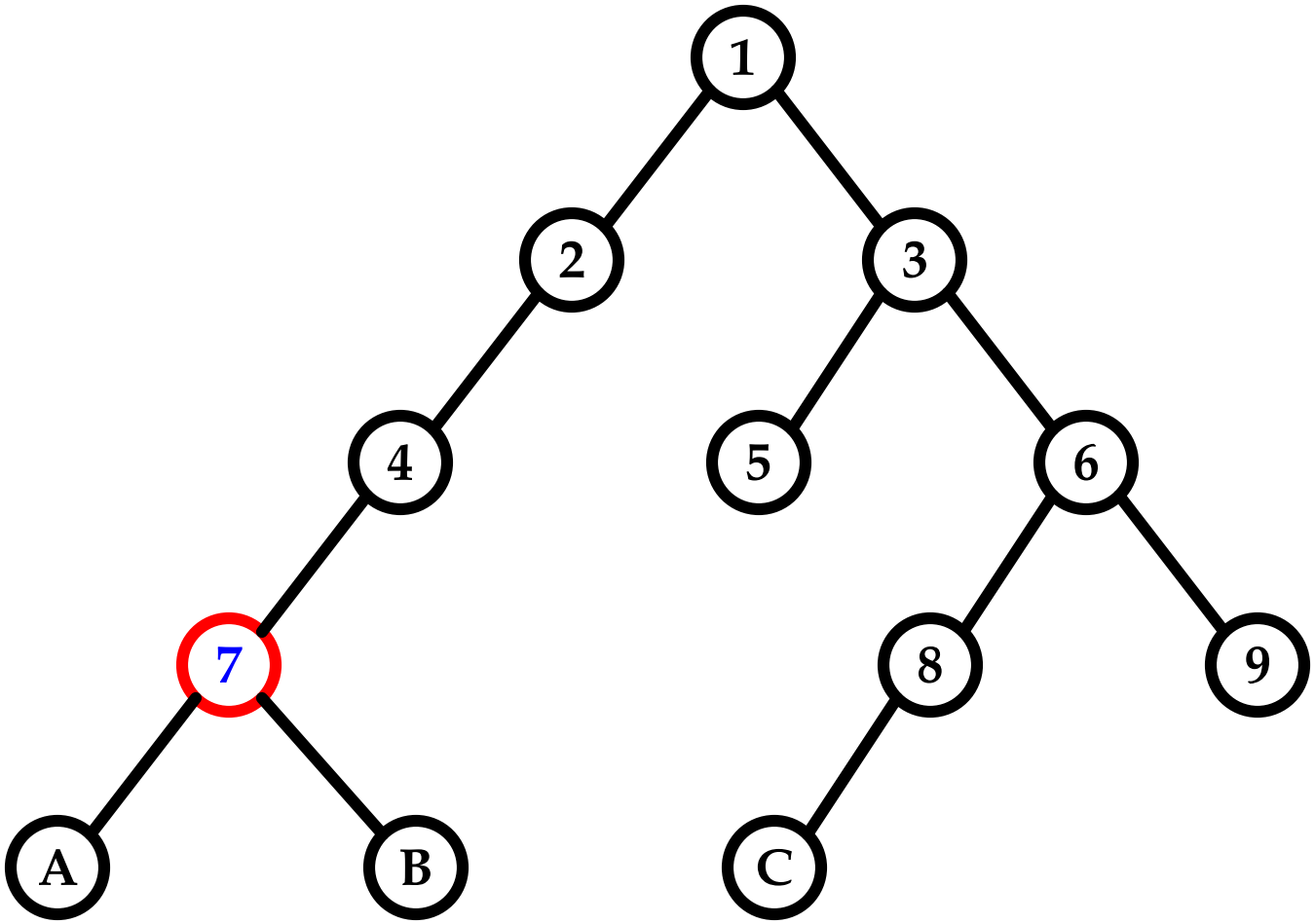
Caminhamento central

A



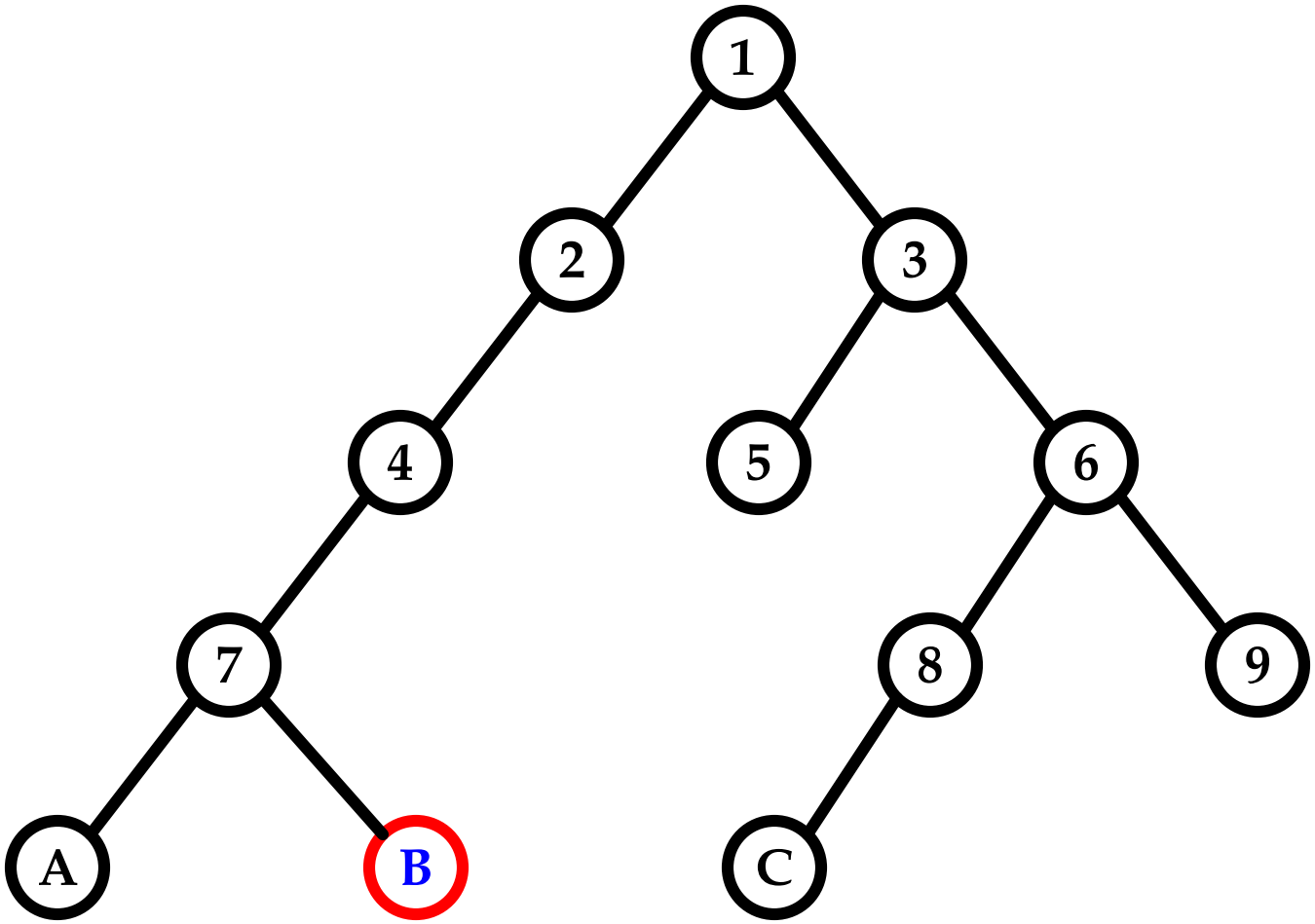
Caminhamento central

A 7



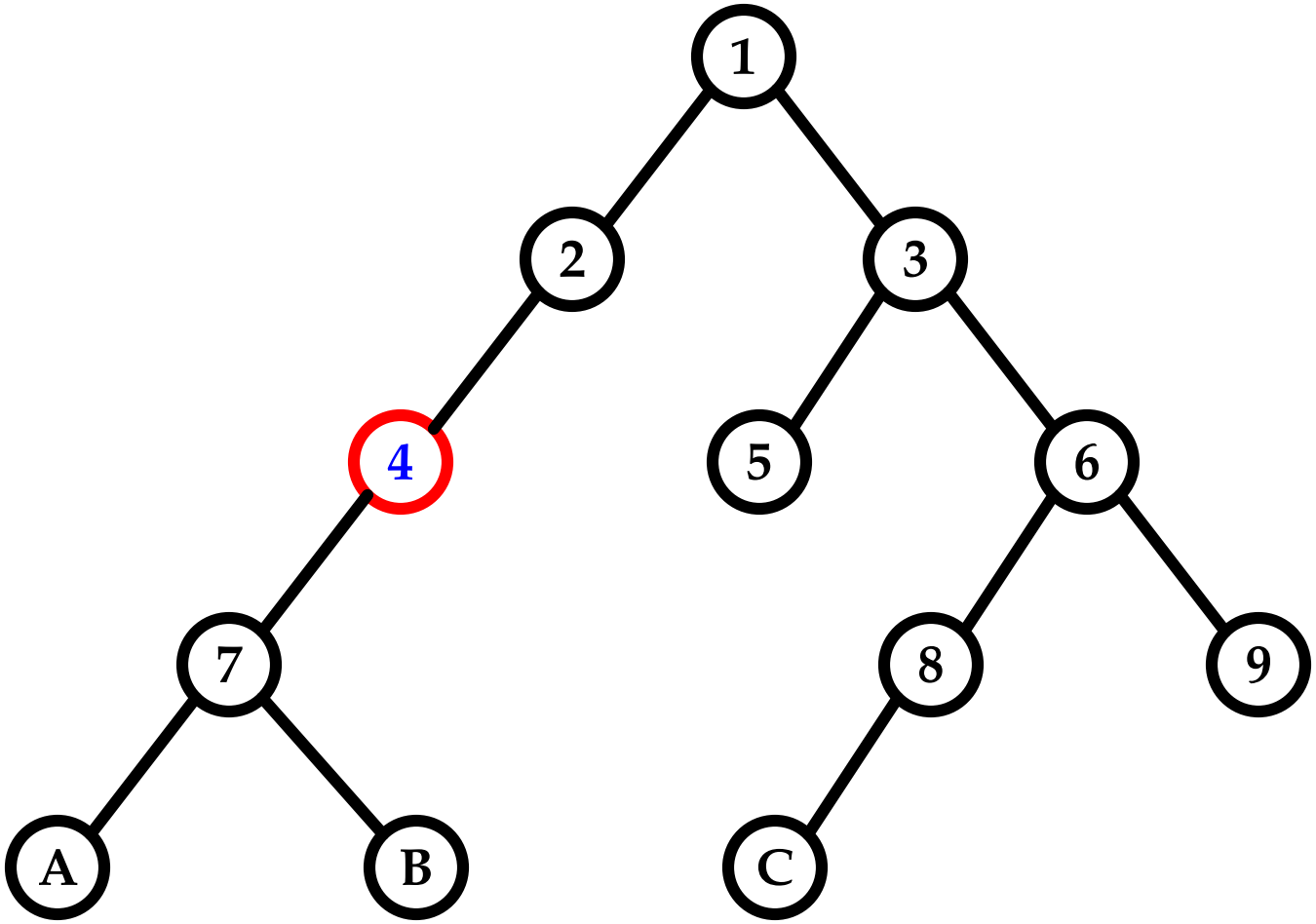
Caminhamento central

A 7 B



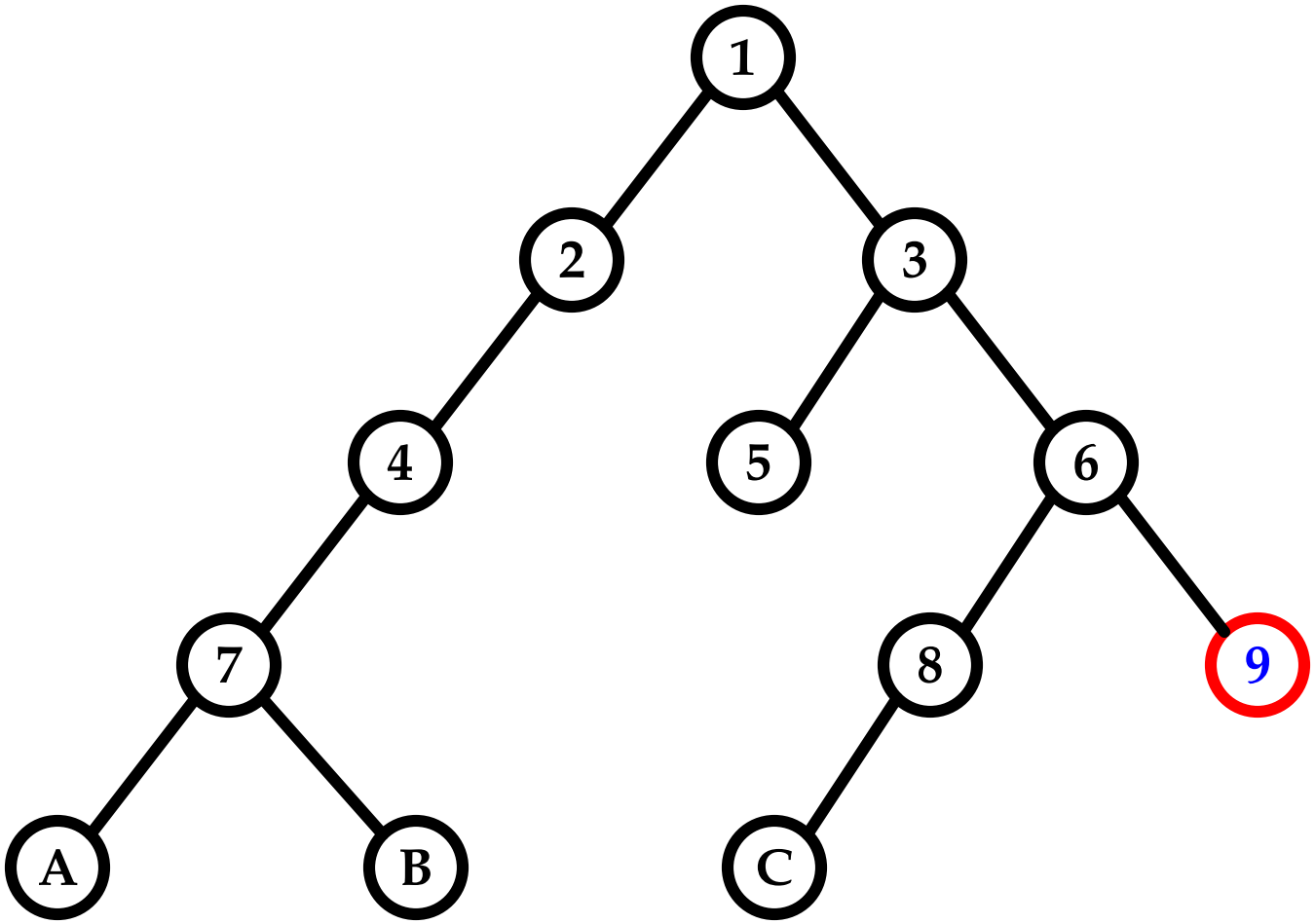
Caminhamento central

A 7 B 4



Caminhamento central

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

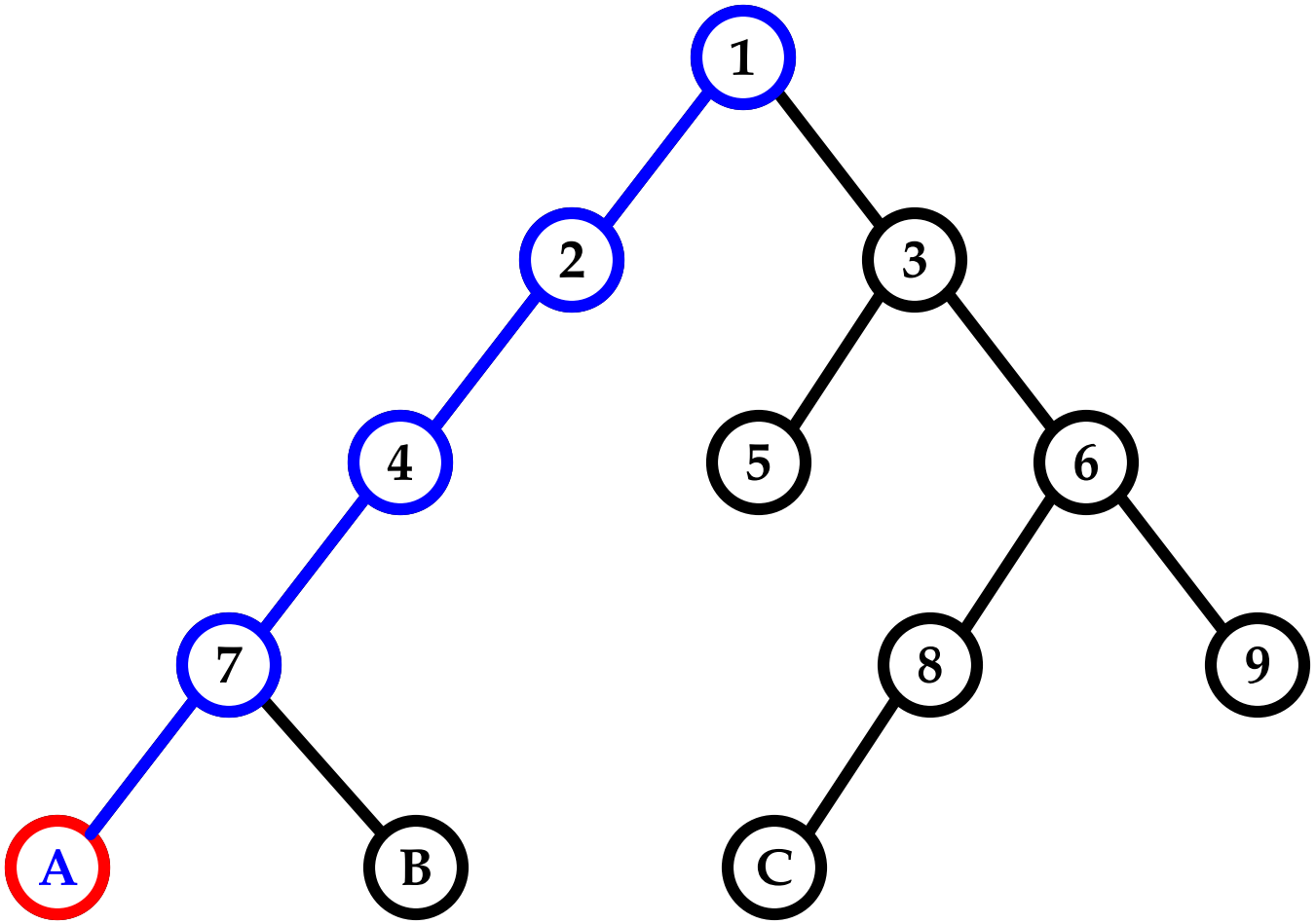


Caminhamento pós-ordem

```
void pos_ordem(struct arvore *f)
{
    if(f == NULL) { return; }
    pos_ordem(f->esq);
    pos_ordem(f->dir);
    printf("%d", f->dado);
}
```

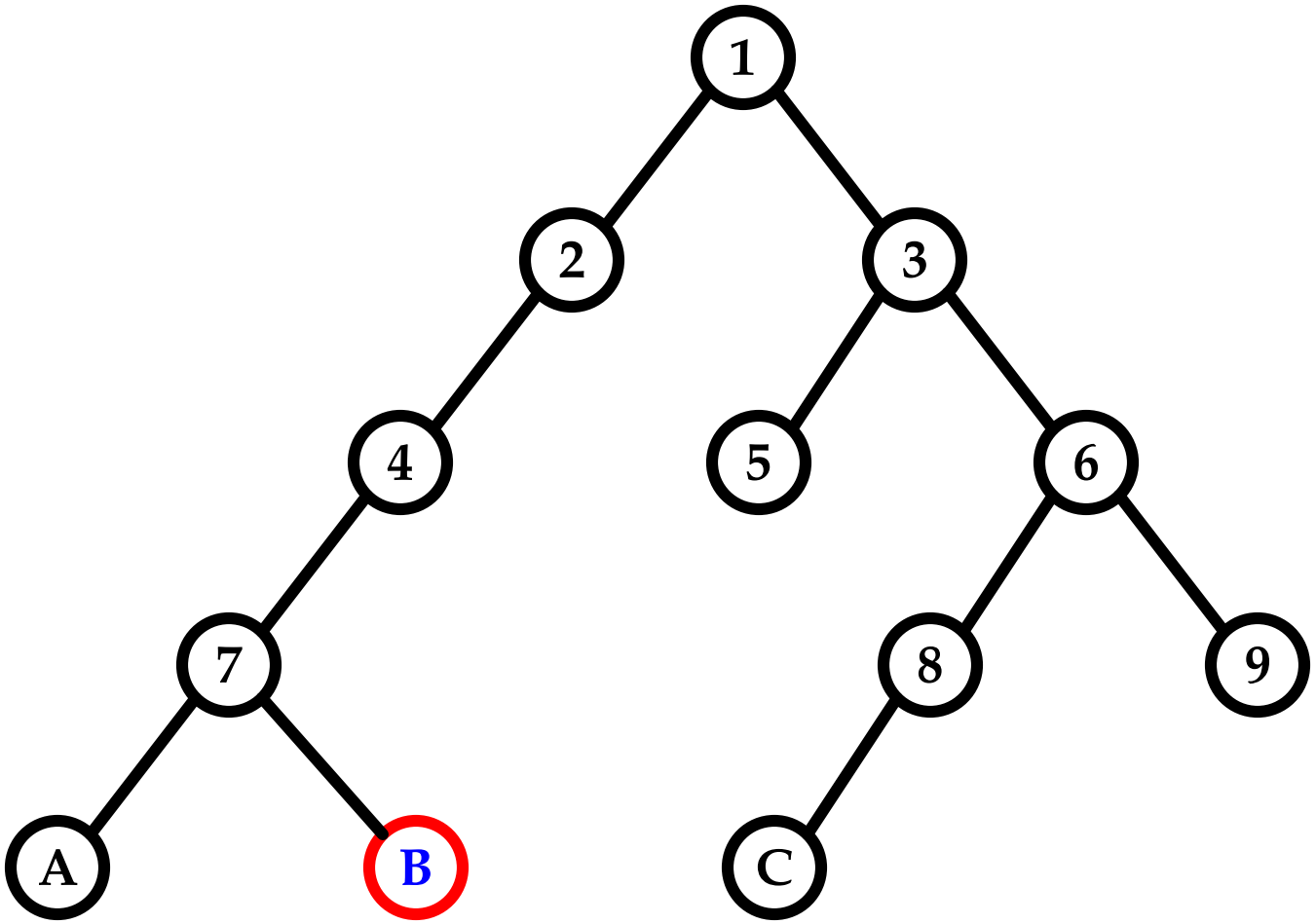
Caminhamento pós-ordem

A



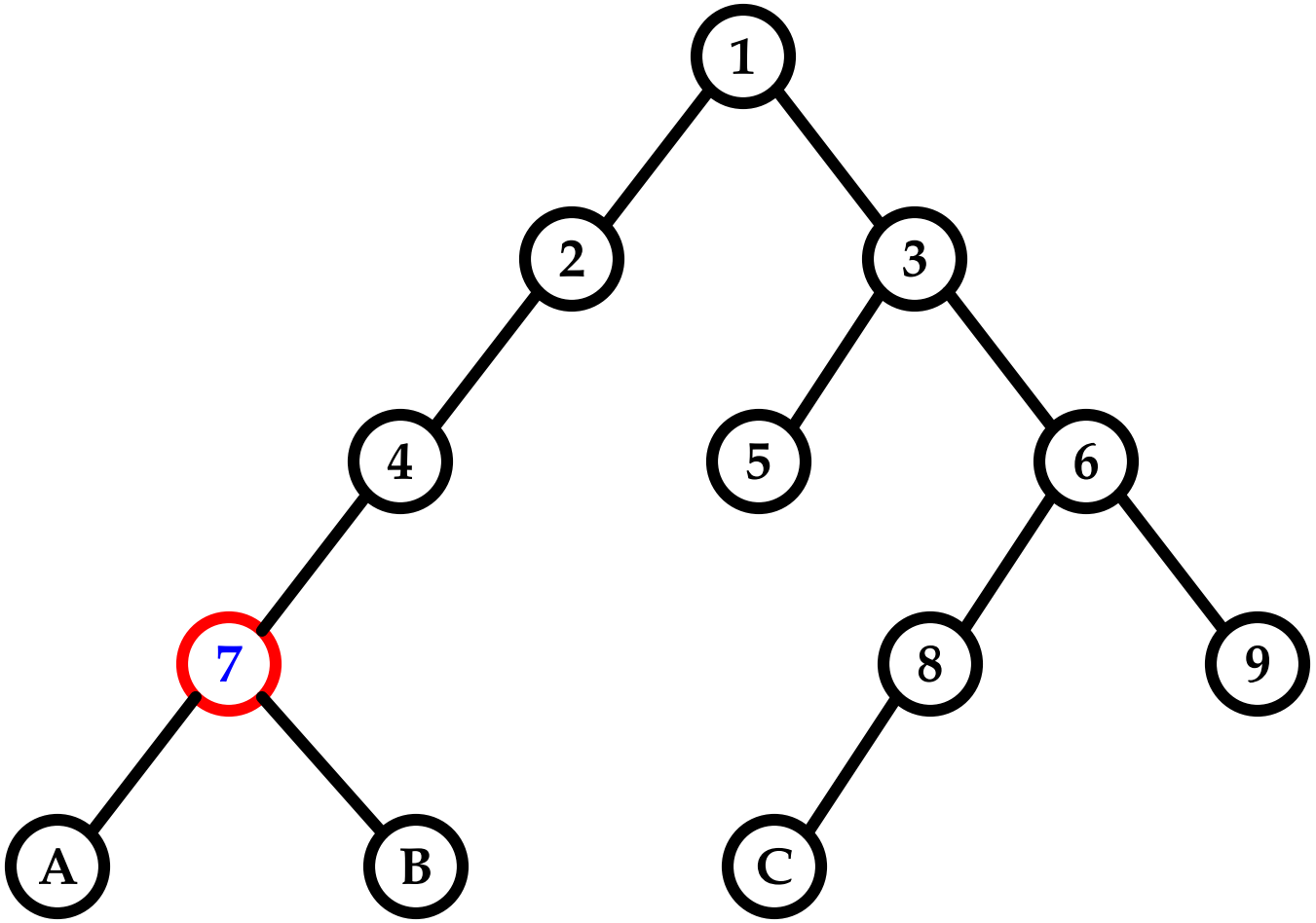
Caminhamento pós-ordem

A B



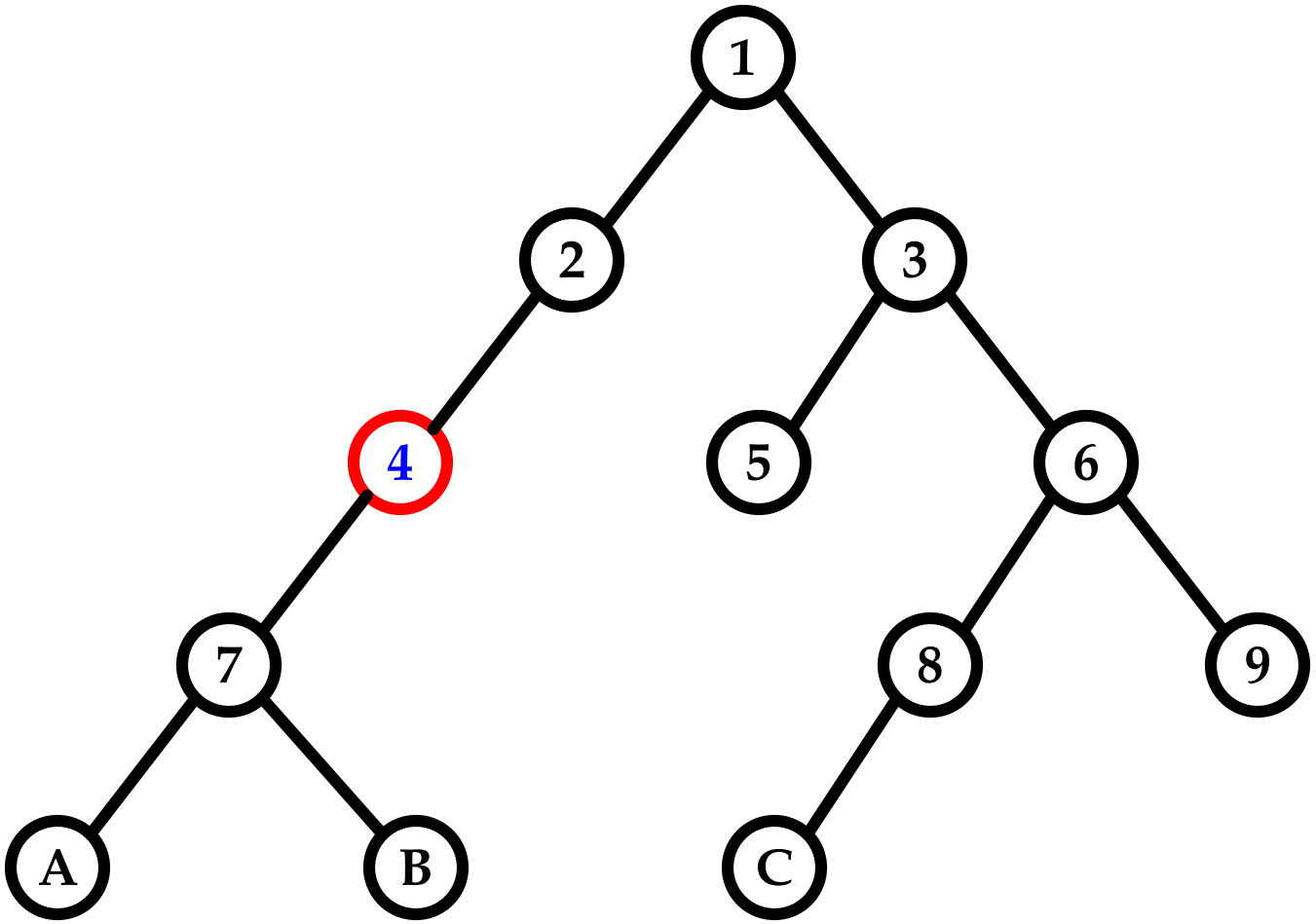
Caminhamento pós-ordem

A B 7



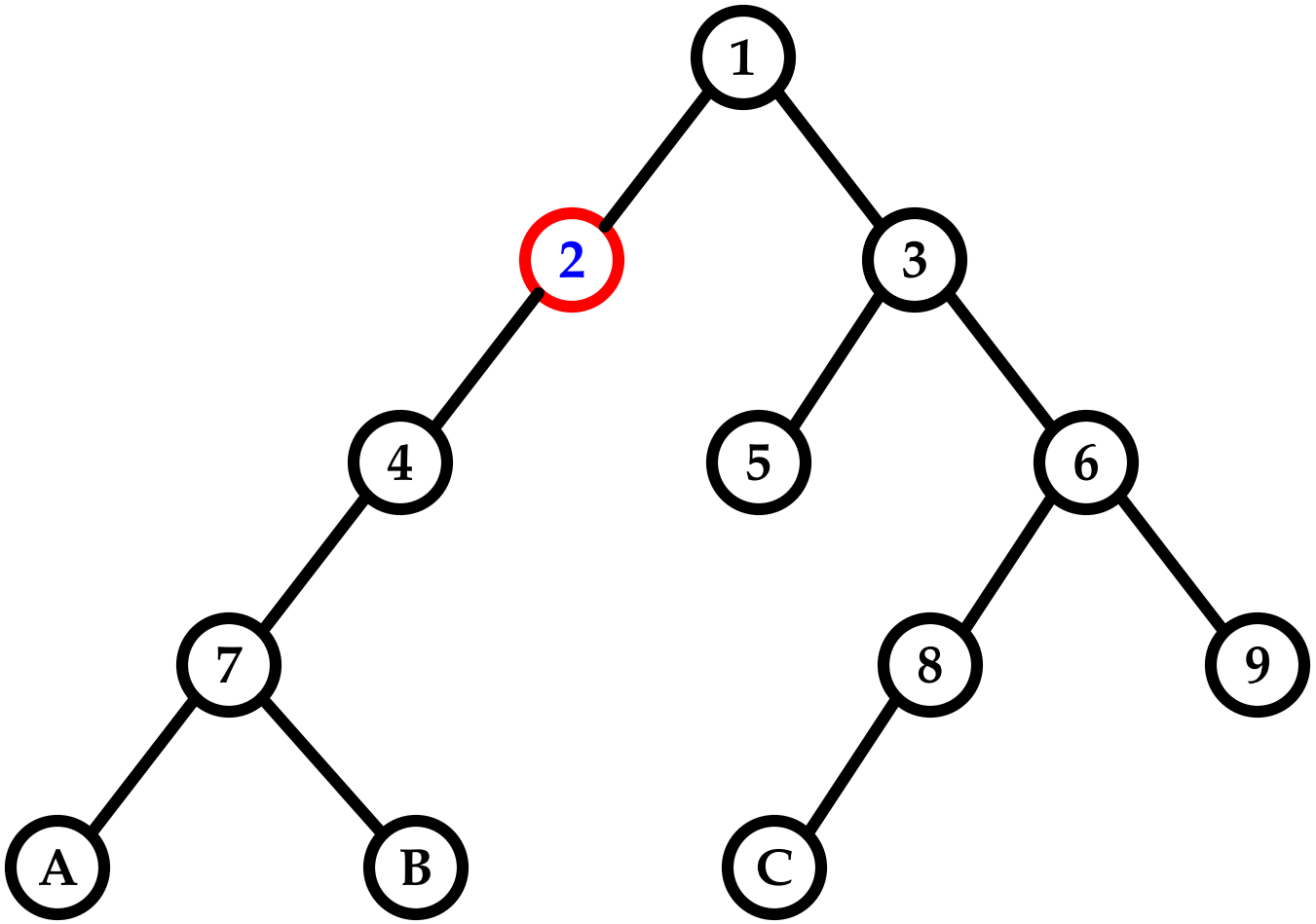
Caminhamento pós-ordem

A B 7 4



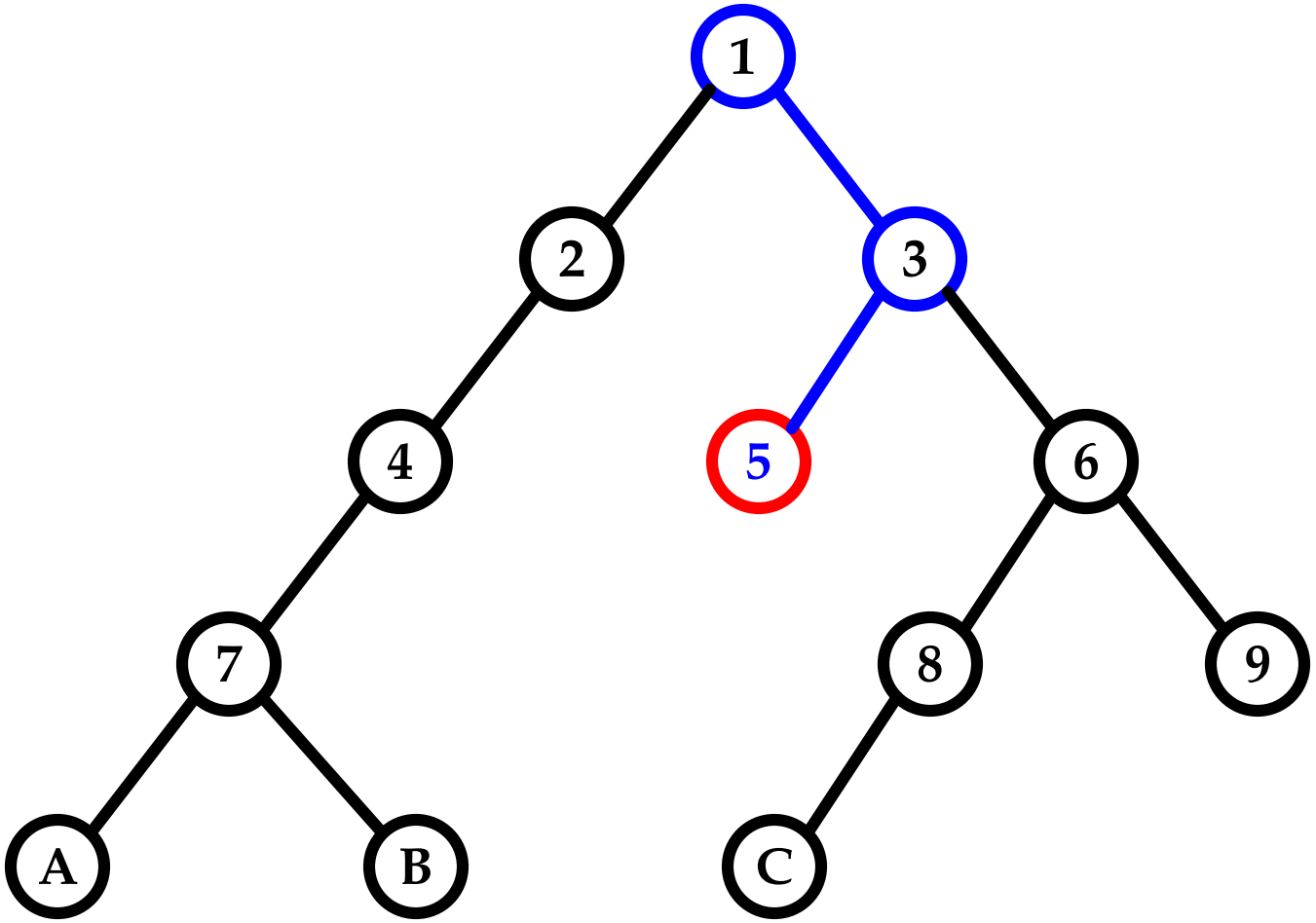
Caminhamento pós-ordem

A B 7 4 2



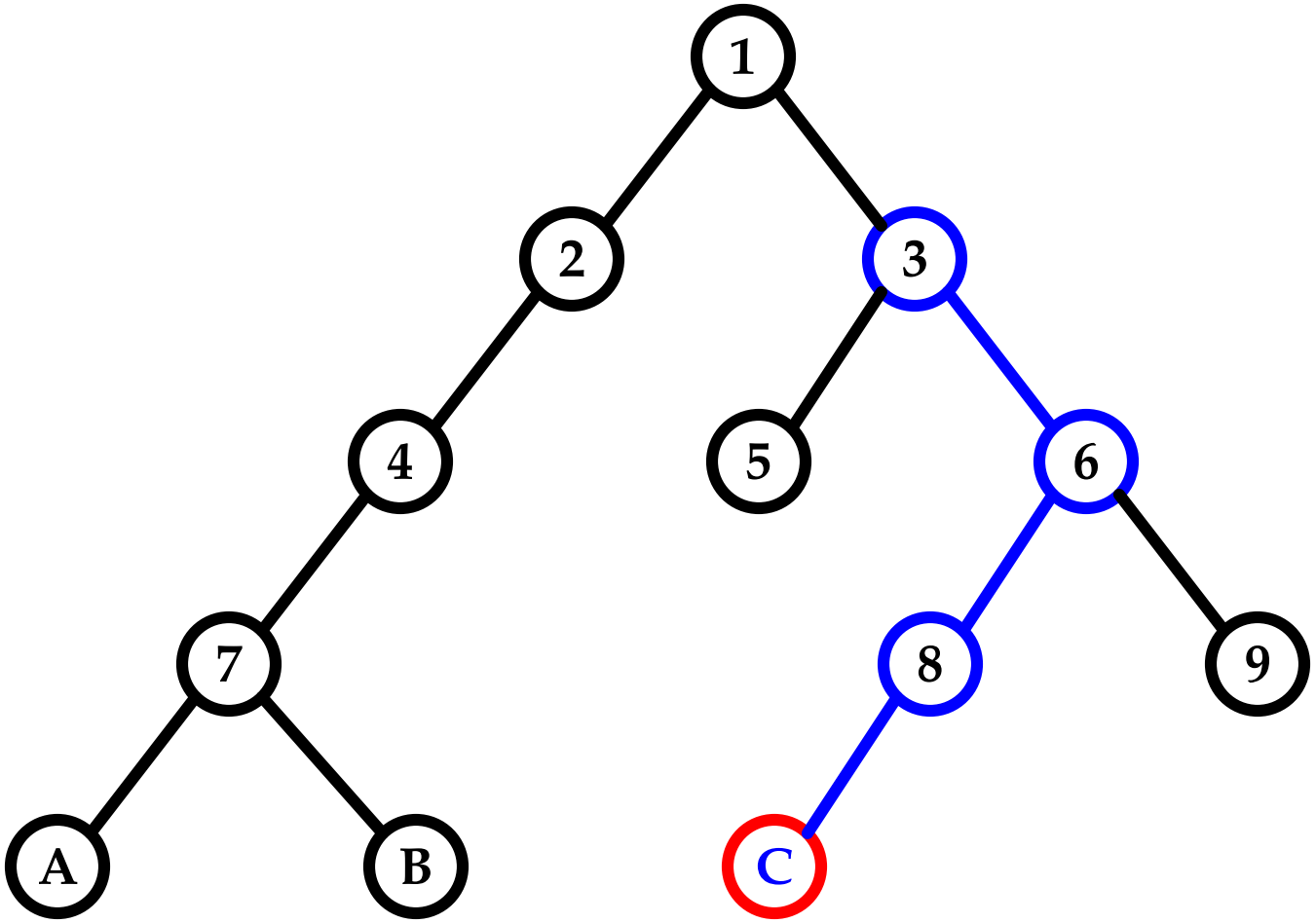
Caminhamento pós-ordem

A B 7 4 2 5



Caminhamento pós-ordem

A B 7 4 2 5 C



Caminhamento pós-ordem

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