

Grafos

Árvores: Travessias

Prof. Edson Alves

Faculdade UnB Gama

DFS em árvores

DFS em árvores

- ★ A estrutura das árvores simplifica a implementação da DFS

DFS em árvores

- ★ A estrutura das árvores simplifica a implementação da DFS
- ★ Por conta da ausência de ciclos, a DFS em árvores dispensa o registro dos vértices já visitados

DFS em árvores

- ★ A estrutura das árvores simplifica a implementação da DFS
- ★ Por conta da ausência de ciclos, a DFS em árvores dispensa o registro dos vértices já visitados
- ★ Este registro é substituído por um parâmetro extra, com o vértice p que antecedeu u na travessia

DFS em árvores

- ★ A estrutura das árvores simplifica a implementação da DFS
- ★ Por conta da ausência de ciclos, a DFS em árvores dispensa o registro dos vértices já visitados
- ★ Este registro é substituído por um parâmetro extra, com o vértice p que antecedeu u na travessia
- ★ A complexidade de memória se reduz de $O(V)$ para $O(1)$

DFS em árvores

- ★ A estrutura das árvores simplifica a implementação da DFS
- ★ Por conta da ausência de ciclos, a DFS em árvores dispensa o registro dos vértices já visitados
- ★ Este registro é substituído por um parâmetro extra, com o vértice p que antecedeu u na travessia
- ★ A complexidade de memória se reduz de $O(V)$ para $O(1)$
- ★ Na primeira chamada, $p = 0$ (ou algum sentinela apropriado)

```
void dfs(int u, int p)
{
    // visita/processa u

    for (auto v : adj[u])
        if (v != p)
            dfs(v, u);
}
```


Números de nós na subárvore

Números de nós na subárvore

★ A DFS, em conjunto com técnicas de programação dinâmica, permite computar em $O(N)$ algumas características da árvore

Números de nós na subárvore

- ★ A DFS, em conjunto com técnicas de programação dinâmica, permite computar em $O(N)$ algumas características da árvore
- ★ Um exemplo seria o número de nós `nodes[u]` da subárvore cuja raiz é u

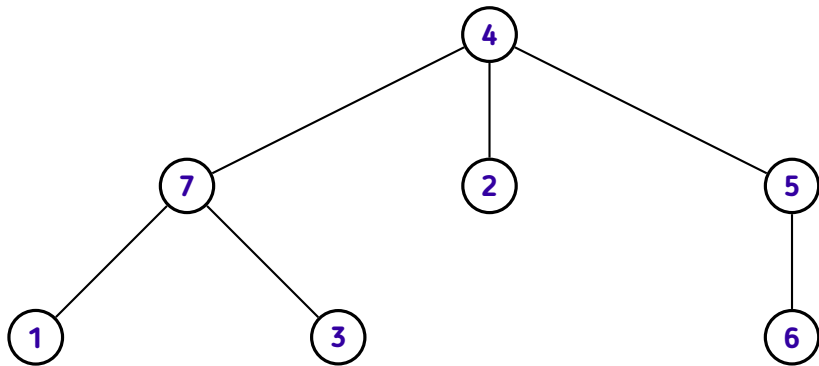
Números de nós na subárvore

- ★ A DFS, em conjunto com técnicas de programação dinâmica, permite computar em $O(N)$ algumas características da árvore
- ★ Um exemplo seria o número de nós $\text{nodes}[u]$ da subárvore cuja raiz é u
- ★ Se u é folha, então $\text{nodes}[u] = 1$

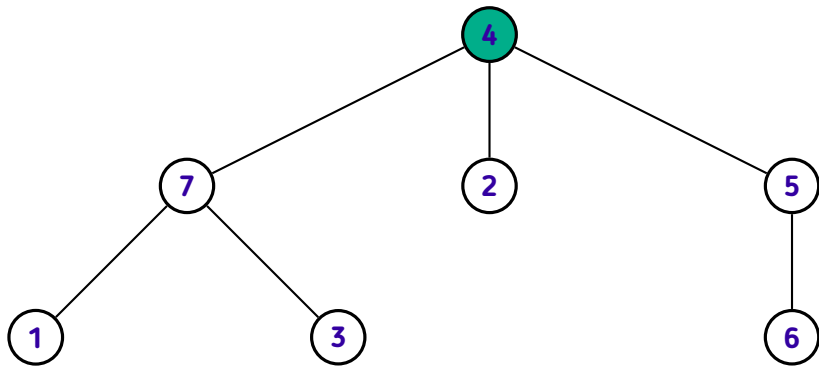
Números de nós na subárvore

- ★ A DFS, em conjunto com técnicas de programação dinâmica, permite computar em $O(N)$ algumas características da árvore
- ★ Um exemplo seria o número de nós $\text{nodes}[u]$ da subárvore cuja raiz é u
- ★ Se u é folha, então $\text{nodes}[u] = 1$
- ★ Caso contrário,

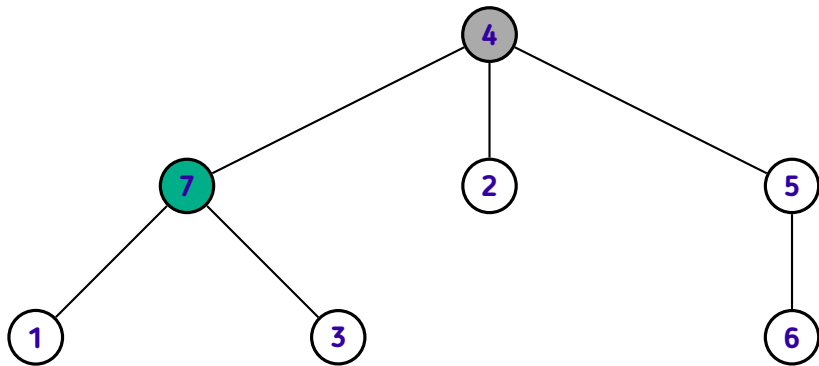
$$\text{nodes}[u] = 1 + \sum_{v \in \text{adj}[u]} \text{nodes}[v]$$



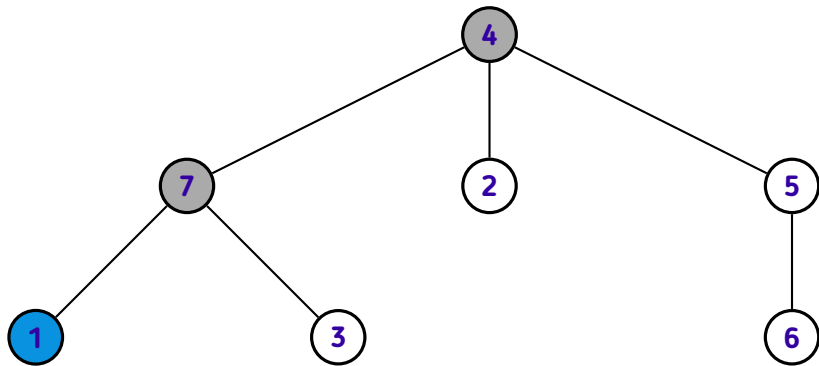
	1	2	3	4	5	6	7
nodes[u] =	-	-	-	-	-	-	-



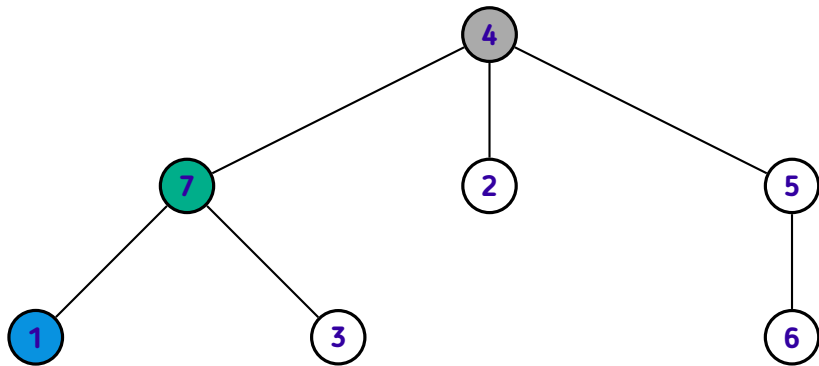
	1	2	3	4	5	6	7
<code>nodes[u] =</code>	-	-	-	1	-	-	-



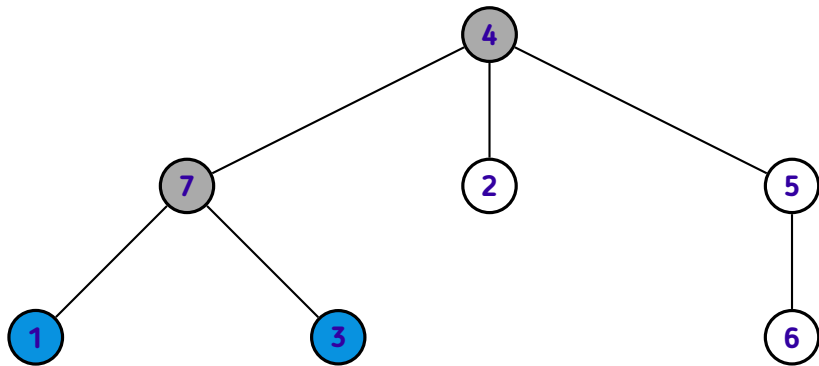
	1	2	3	4	5	6	7
<code>nodes[u] =</code>	-	-	-	1	-	-	1



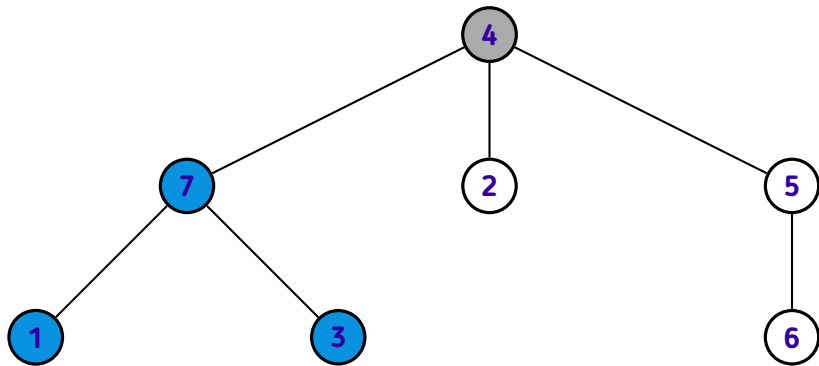
	1	2	3	4	5	6	7
$\text{nodes}[u] =$	1	-	-	1	-	-	1



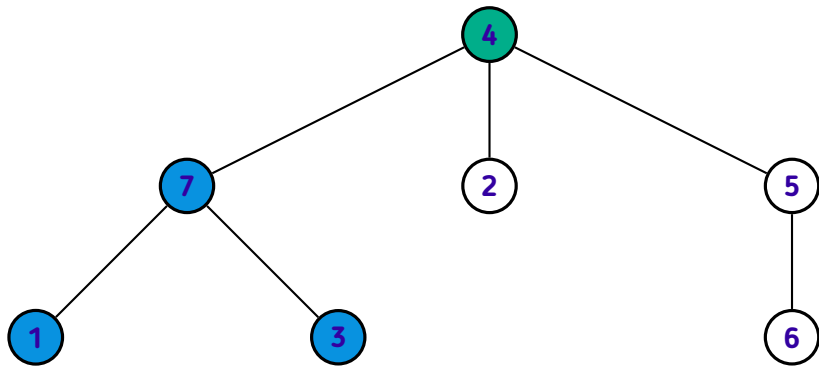
	1	2	3	4	5	6	7
nodes[u] =	1	-	-	1	-	-	2



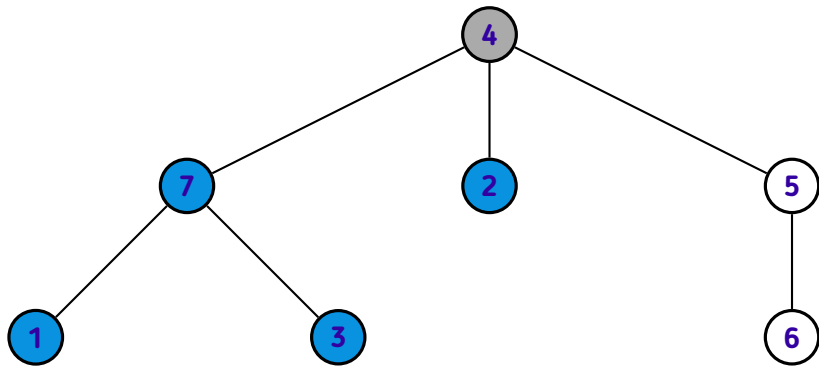
	1	2	3	4	5	6	7
nodes[u] =	1	-	1	1	-	-	2



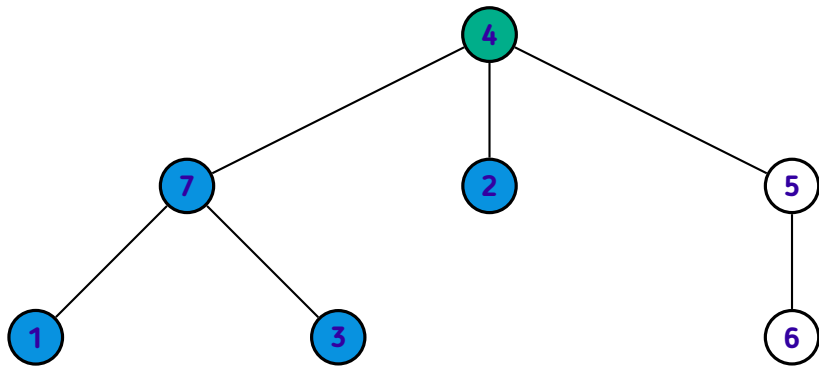
	1	2	3	4	5	6	7
nodes[u] =	1	-	1	1	-	-	3



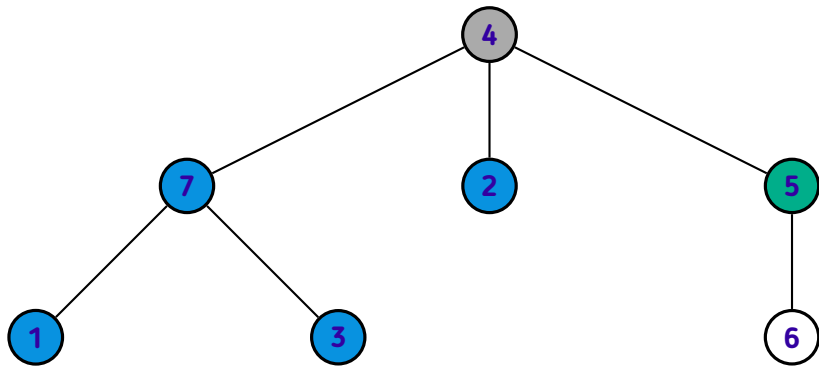
	1	2	3	4	5	6	7
nodes[u] =	1	-	1	4	-	-	3



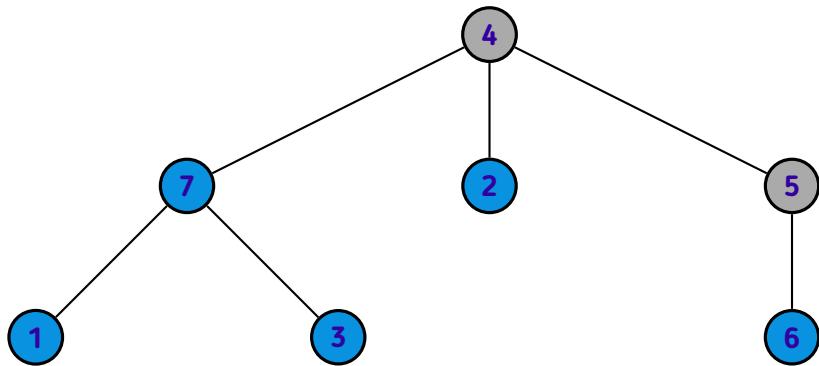
	1	2	3	4	5	6	7
nodes[u] =	1	1	1	4	-	-	3



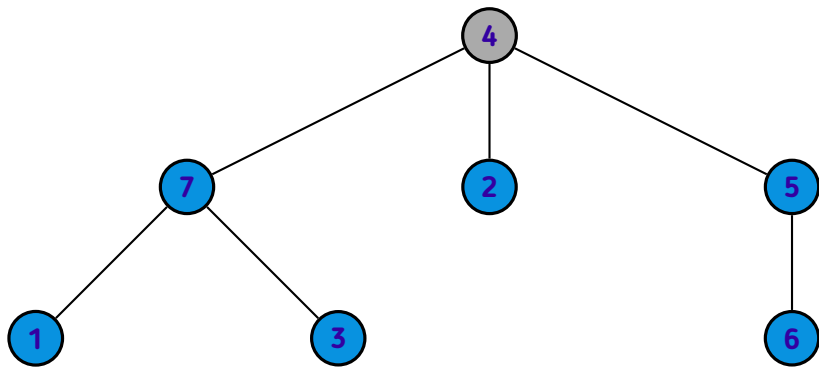
	1	2	3	4	5	6	7
nodes[u] =	1	1	1	5	-	-	3



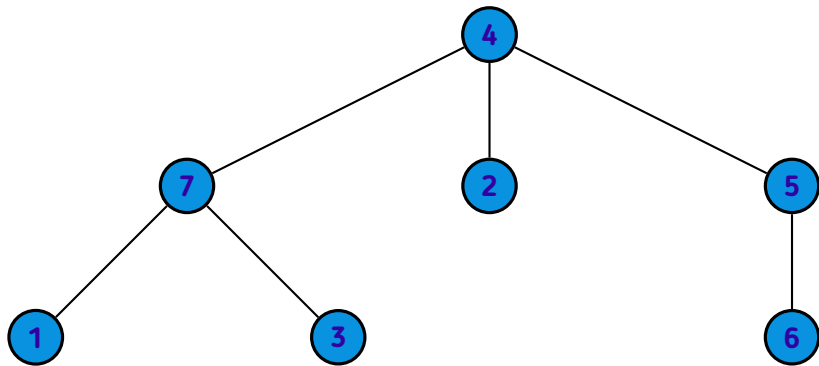
	1	2	3	4	5	6	7
<code>nodes[u]</code> =	1	1	1	5	1	-	3



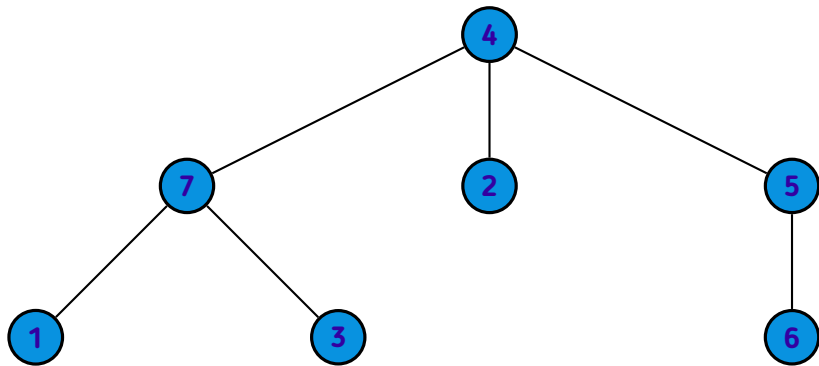
	1	2	3	4	5	6	7
nodes[u] =	1	1	1	5	1	1	3



	1	2	3	4	5	6	7
nodes[u] =	1	1	1	5	2	1	3



	1	2	3	4	5	6	7
<code>nodes[u] =</code>	1	1	1	7	2	1	3



	1	2	3	4	5	6	7
<code>nodes[u] =</code>	1	1	1	7	2	1	3

```
void dfs(int u, int p, vector<int>& ns)
{
    for (auto v : adj[u])
    {
        if (v == p)
            continue;

        dfs(v, u, ns);
        ns[u] += ns[v];
    }
}

vector<int> nodes(int u, int N)
{
    vector<int> ns(N + 1, 1);

    dfs(u, 0, ns);

    return ns;
}
```

Maior caminho até uma folha

Maior caminho até uma folha

★ Outra aplicação de DFS com DP é o cálculo do tamanho, em arestas, do maior caminho $\text{toLeaf}[u]$ de u até uma folha

Maior caminho até uma folha

★ Outra aplicação de DFS com DP é o cálculo do tamanho, em arestas, do maior caminho $\text{toLeaf}[u]$ de u até uma folha

★ Se u for uma folha, então $\text{toLeaf}[u] = 0$

Maior caminho até uma folha

★ Outra aplicação de DFS com DP é o cálculo do tamanho, em arestas, do maior caminho $\text{toLeaf}[u]$ de u até uma folha

★ Se u for uma folha, então $\text{toLeaf}[u] = 0$

★ Caso contrário,

$$\text{toLeaf}[u] = 1 + \max_{v \in \text{adj}[u]} \{ \text{toLeaf}[v] \}$$

Maior caminho até uma folha

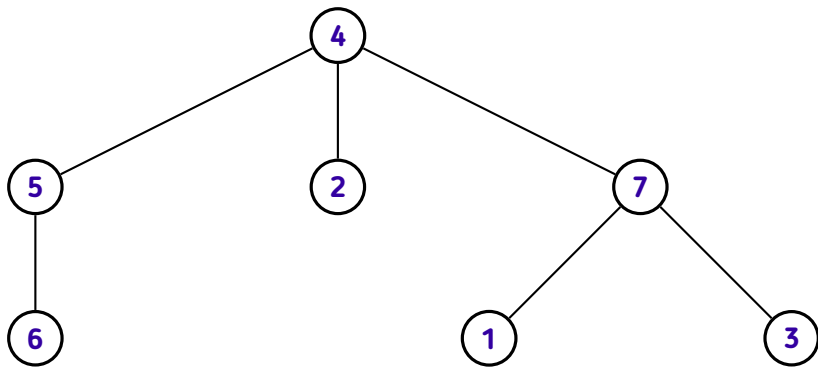
★ Outra aplicação de DFS com DP é o cálculo do tamanho, em arestas, do maior caminho $\text{toLeaf}[u]$ de u até uma folha

★ Se u for uma folha, então $\text{toLeaf}[u] = 0$

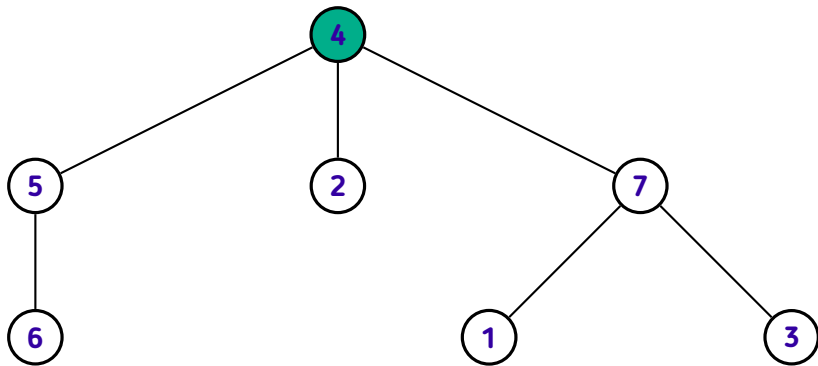
★ Caso contrário,

$$\text{toLeaf}[u] = 1 + \max_{v \in \text{adj}[u]} \{ \text{toLeaf}[v] \}$$

★ Este algoritmo pode ser adaptado para computar o tamanho como soma dos pesos das arestas do caminho que vai de u até a folha

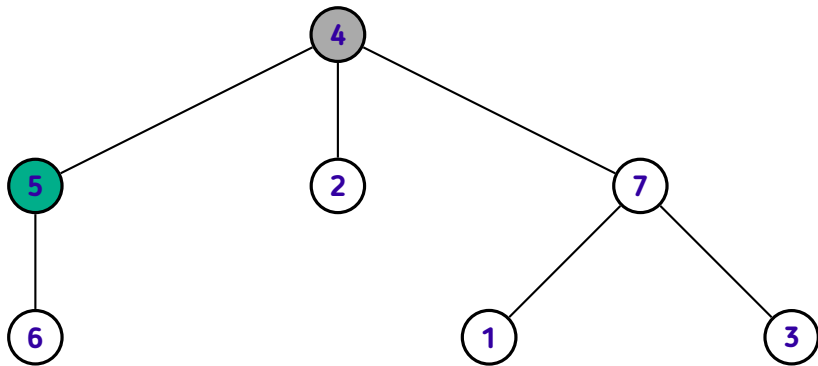


	1	2	3	4	5	6	7
toLeaf[u] =	-	-	-	-	-	-	-



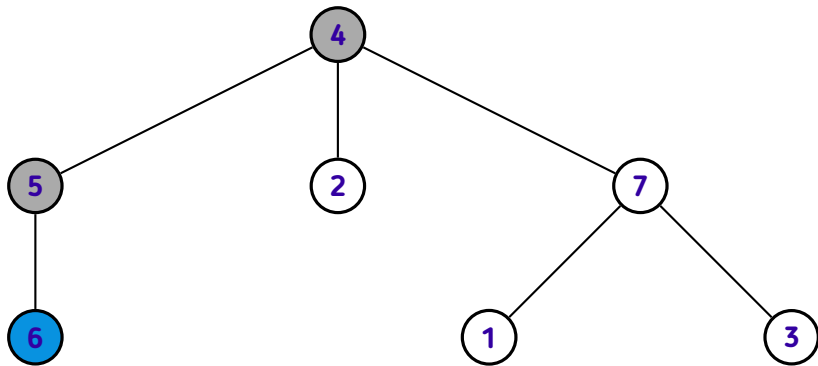
toLeaf[u] =

1	2	3	4	5	6	7
-	-	-	-	-	-	-



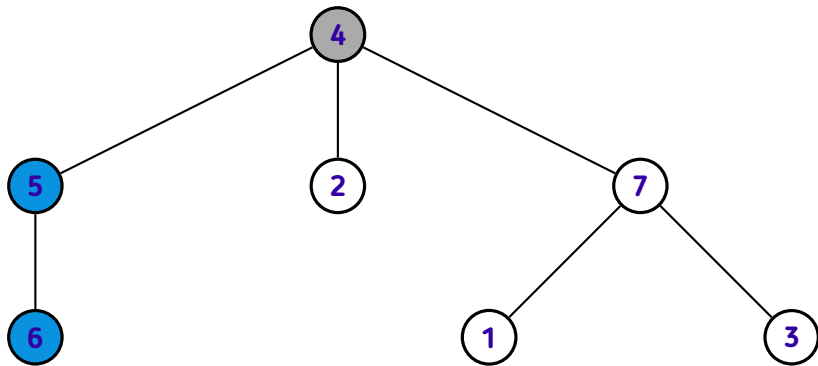
$\text{toLeaf}[u] =$

1	2	3	4	5	6	7
-	-	-	-	-	-	-



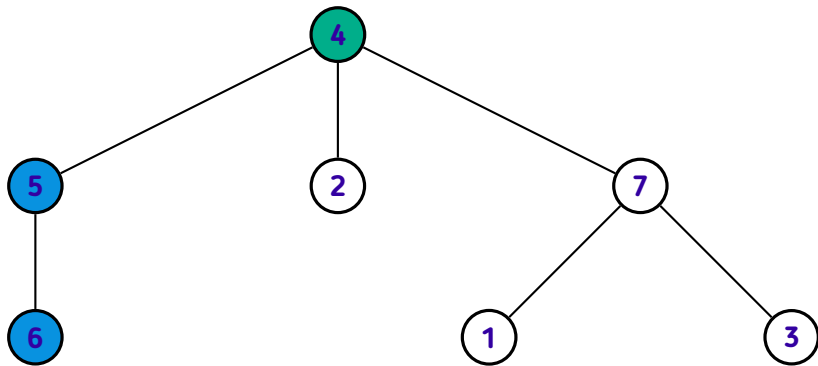
$\text{toLeaf}[u] =$

1	2	3	4	5	6	7
-	-	-	-	-	0	-



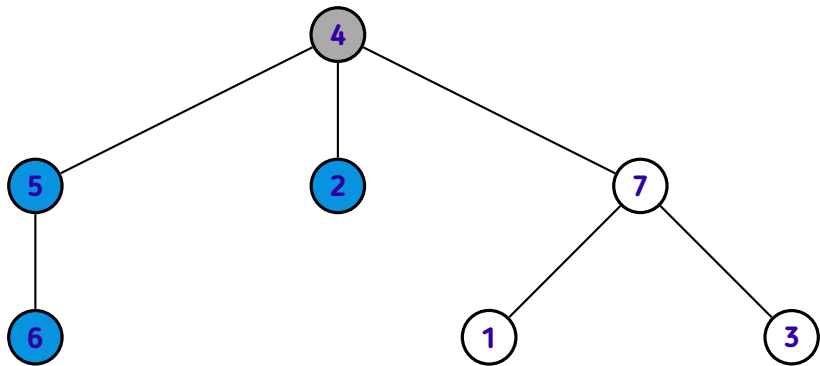
$\text{toLeaf}[u] =$

1	2	3	4	5	6	7
-	-	-	-	1	0	-

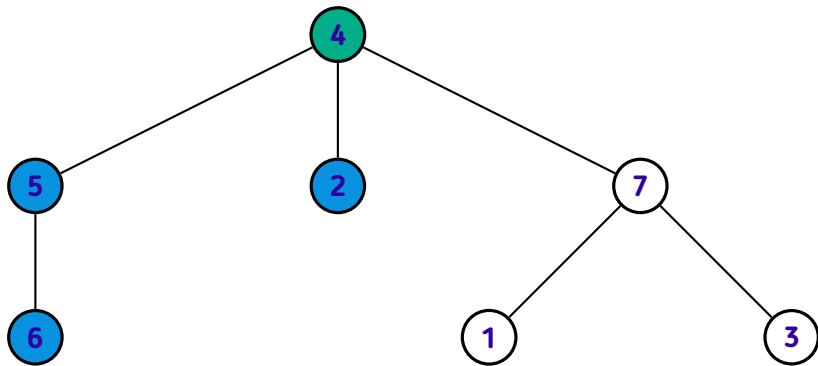


toLeaf[u] =

1	2	3	4	5	6	7
-	-	-	2	1	0	-

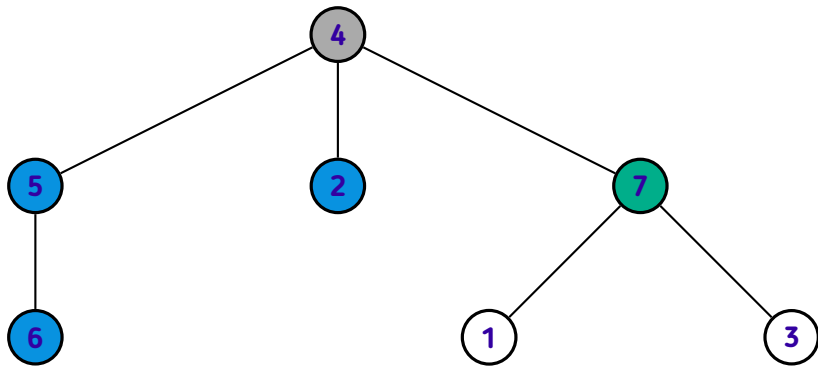


	1	2	3	4	5	6	7
toLeaf[u] =	-	0	-	2	1	0	-

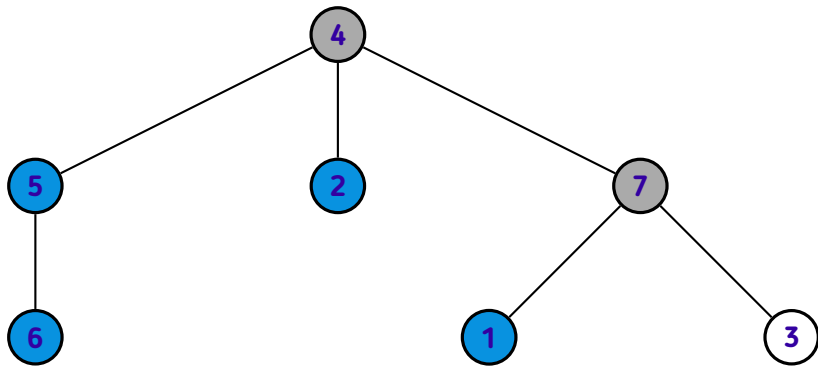


$\text{toLeaf}[u] =$

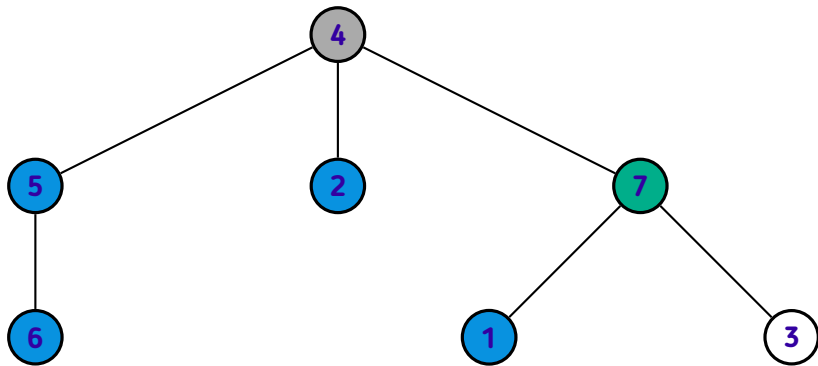
1	2	3	4	5	6	7
-	0	-	2	1	0	-



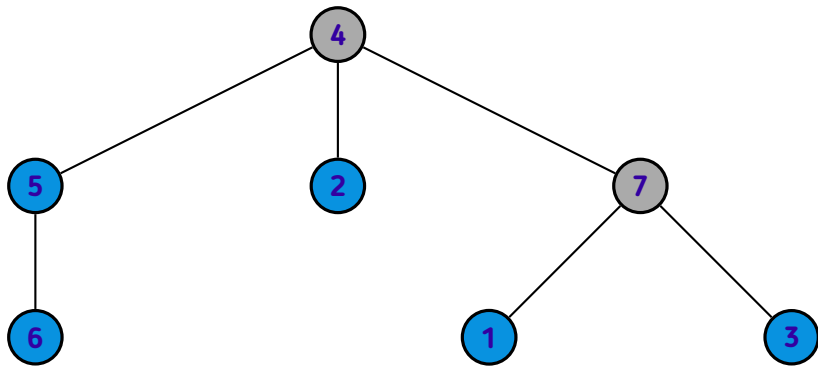
	1	2	3	4	5	6	7
toLeaf[u] =	-	0	-	2	1	0	-



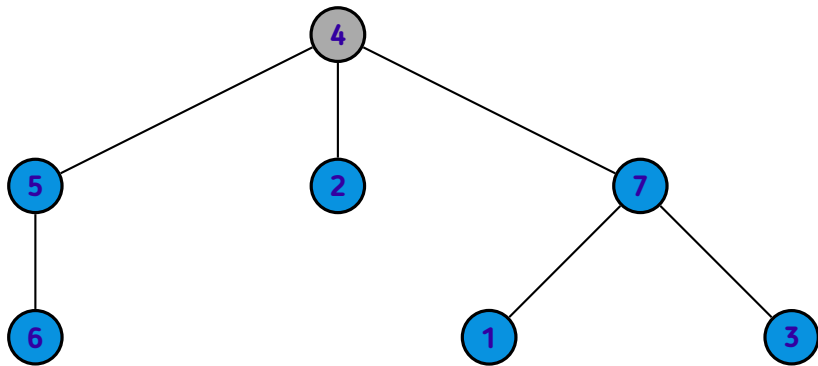
	1	2	3	4	5	6	7
toLeaf[u] =	0	0	-	2	1	0	-



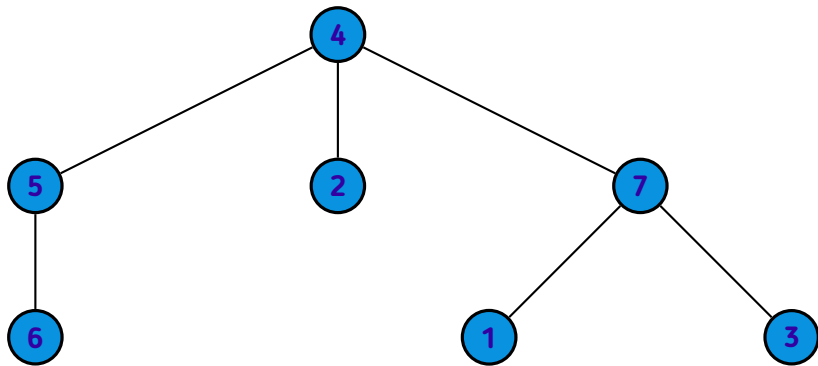
	1	2	3	4	5	6	7
toLeaf[u] =	0	0	-	2	1	0	0



	1	2	3	4	5	6	7
toLeaf[u] =	0	0	0	2	1	0	0



	1	2	3	4	5	6	7
toLeaf[u] =	0	0	0	2	1	0	1



	1	2	3	4	5	6	7
toLeaf[u] =	0	0	0	2	1	0	1


```
void dfs(int u, int p, vector<int>& t1)
{
    for (auto v : adj[u])
    {
        if (v == p)
            continue;

        dfs(v, u, t1);
        t1[u] = max(t1[u], 1 + t1[v]);
    }
}

vector<int> to_leaf(int u, int N)
{
    vector<int> t1(N + 1, 0);

    dfs(u, 0, t1);

    return t1;
}
```

Referências

1. DROZDEK, Adam. *Algoritmos e Estruturas de Dados em C++*, 2002.
2. HALIM, Felix; HALIM, Steve. *Competitive Programming 3*, 2010.
3. LAAKSONEN, Antti. *Competitive Programmer's Handbook*, 2018.
4. SKIENA, Steven; REVILLA, Miguel. *Programming Challenges*, 2003.
5. Wikipédia. *Tree (graph theory)*, acesso em 06/08/2021.