Grafos

Árvores: Diâmetro

Prof. Edson Alves

Faculdade UnB Gama

Definição de diâmetro

Definição de diâmetro

Seja $G(V, E)$ um grafo. O **diâmetro** de G é igual ao maior dentre todos os tamanhos dos caminhos entre os pares de vértices $u, v \in V$.

Características do diâmetro

Características do diâmetro

- ⋆ O caminho cujo tamanho determina o diâmetro não é, necessariamente, único

Características do diâmetro

- $\star$ O caminho cujo tamanho determina o diâmetro não é, necessariamente, único

- $\star$ Computar todos os tamanho com Floyd-Warshall em $O(V^3)$ e determinar o maior dentre eles em $O(V^2)$ determina o diâmetro corretamente

Características do diâmetro

- $\star$ O caminho cujo tamanho determina o diâmetro não é, necessariamente, único
- $\star$ Computar todos os tamanho com Floyd-Warshall em $O(V^3)$ e determinar o maior dentre eles em $O(V^2)$ determina o diâmetro corretamente
- $\star$ Porém, é possível determinar o diâmetro usando programação dinâmica ou duas DFS

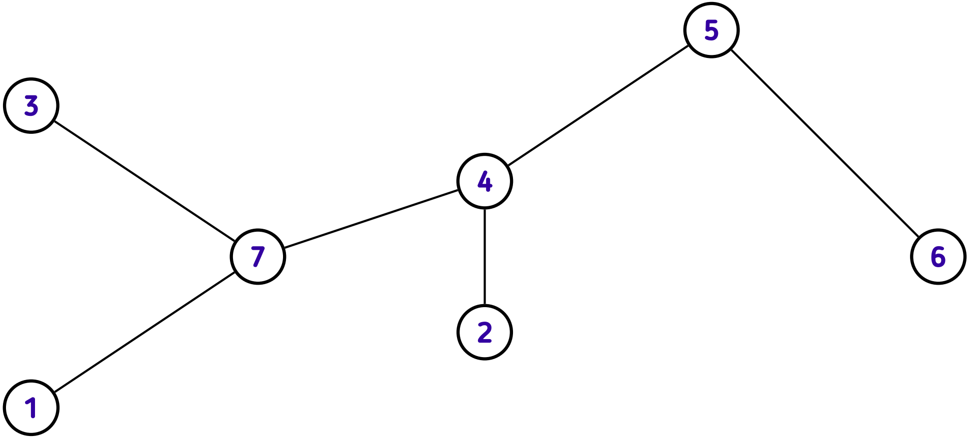
Características do diâmetro

- $\star$ O caminho cujo tamanho determina o diâmetro não é, necessariamente, único

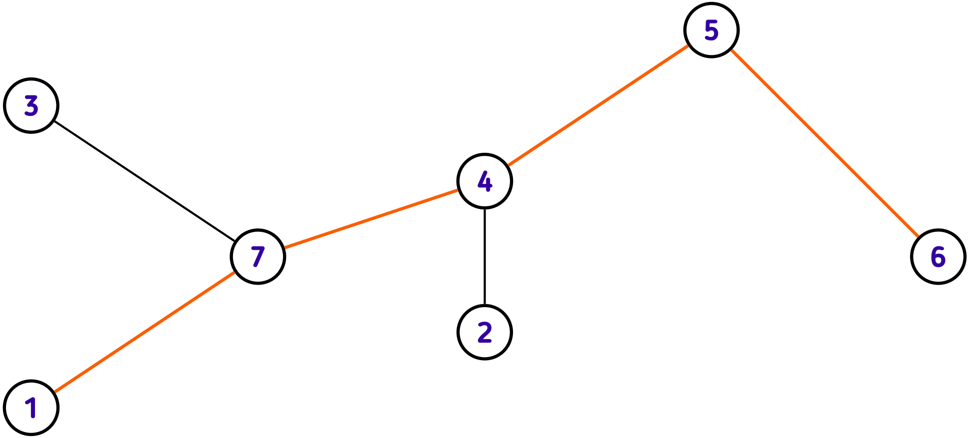
- $\star$ Computar todos os tamanho com Floyd-Warshall em $O(V^3)$ e determinar o maior dentre eles em $O(V^2)$ determina o diâmetro corretamente

- $\star$ Porém, é possível determinar o diâmetro usando programação dinâmica ou duas DFS

- $\star$ Em ambos casos, a complexidade é $O(V)$


5
3
4
7
6
2
1

Diâmetro: 4



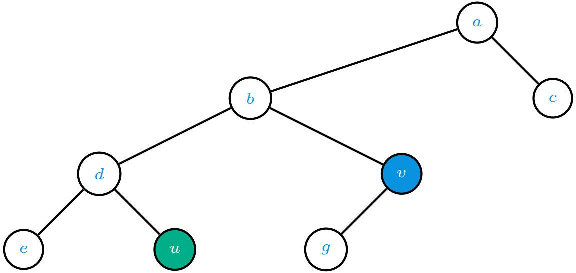
Definição de pico de um caminho

Definição de pico de um caminho

Seja T uma árvore enraizada e considere dois vértices u e v de T. O pico do caminho de u a v é o nó que ocupa o nível baixo em T.

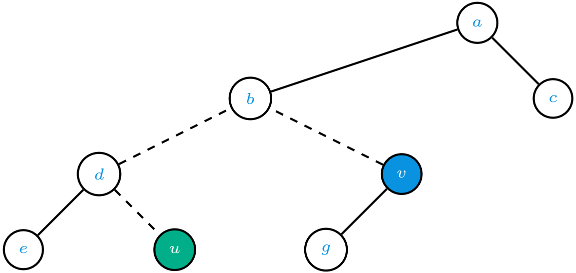
Definição de pico de um caminho

Seja T uma árvore enraizada e considere dois vértices u e v de T. O pico do caminho de u a v é o nó que ocupa o nível baixo em T.



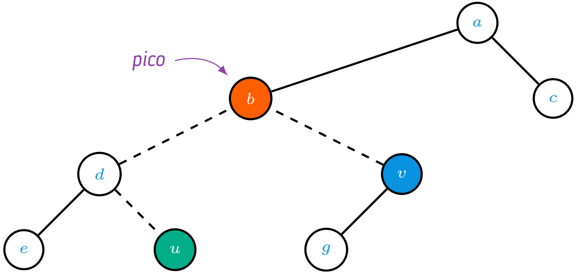
Definição de pico de um caminho

Seja T uma árvore enraizada e considere dois vértices u e v de T. O pico do caminho de u a v é o nó que ocupa o nível baixo em T.



Definição de pico de um caminho

Seja T uma árvore enraizada e considere dois vértices u e v de T. O pico do caminho de u a v é o nó que ocupa o nível baixo em T.



Diâmetro de uma árvore com programação dinâmica

Diâmetro de uma árvore com programação dinâmica

$\text{maxLength}[u]$

Diâmetro de uma árvore com programação dinâmica

Tamanho do maior caminho que tem pico igual a u

$\mathsf{maxLength}[u]$

Diâmetro de uma árvore com programação dinâmica

$$\text{maxLength}[u] = \begin{cases} 0, & \text{se } u \text{ não tem filhos} \\ & \end{cases}$$

Diâmetro de uma árvore com programação dinâmica

$$\text{maxLength}[u] = \begin{cases} 0, & \text{se } u \text{ não tem filhos} \\ 1 + \text{toLeaf}[v], & \text{se } u \text{ tem apenas um filho } v, \\ & \end{cases}$$

Diâmetro de uma árvore com programação dinâmica

$$\text{maxLength}[u] = \begin{cases} 0, & \text{se } u \text{ não tem filhos} \\ 1 + \text{toLeaf}[v], & \text{se } u \text{ tem apenas um filho } v, \\ 2 + \text{toLeaf}[v] + \text{toLeaf}[w], & \text{caso contrário,} \end{cases}$$

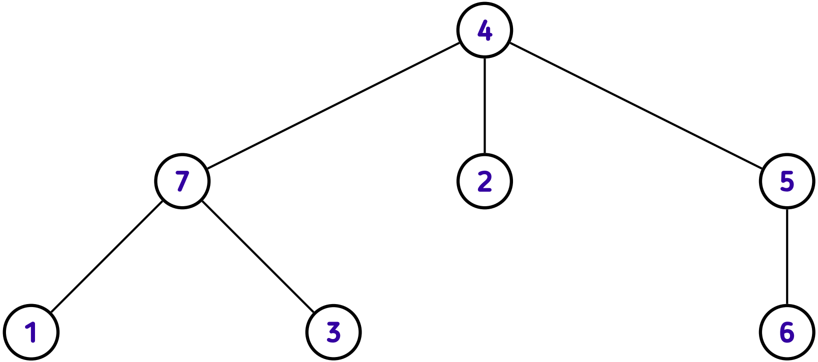
onde v **e** w **são dois filhos distintos de** u **com os dois maiores valores de** toLeaf

Diâmetro de uma árvore com programação dinâmica

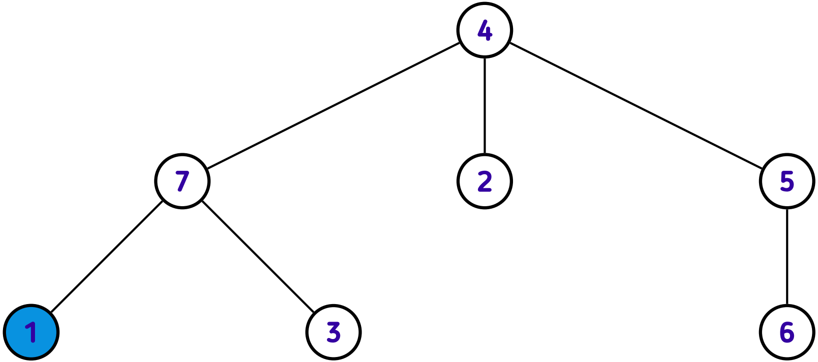
$$\mathsf{diameter}(G) = \max_{u \in V} \{\ \mathsf{maxLength}[u]\ \}$$

$$\mathsf{maxLength}[u] = \begin{cases} 0, & \text{se } u \text{ não tem filhos} \\ 1 + \mathsf{toLeaf}[v], & \text{se } u \text{ tem apenas um filho } v, \\ 2 + \mathsf{toLeaf}[v] + \mathsf{toLeaf}[w], & \text{caso contrário,} \end{cases}$$

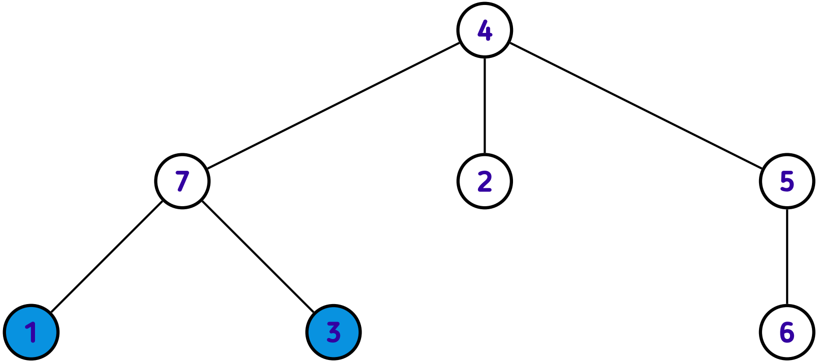
onde v **e** w **são dois filhos distintos de** u **com os dois maiores valores de** toLeaf



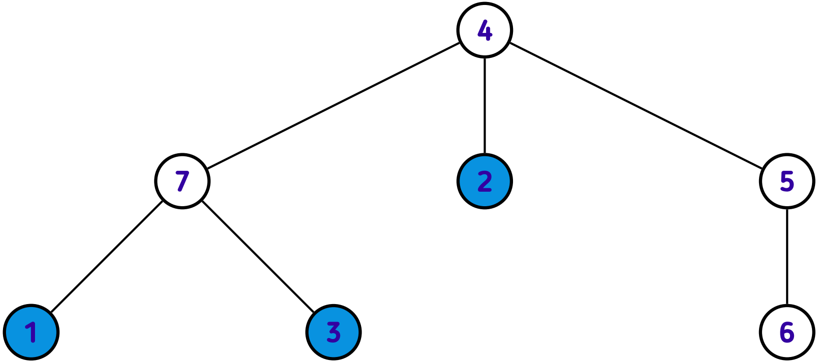
	1	2	3	4	5	6	7
toLeaf$[u]$ =	-	-	-	-	-	-	-
maxLength$[u]$ =	-	-	-	-	-	-	-



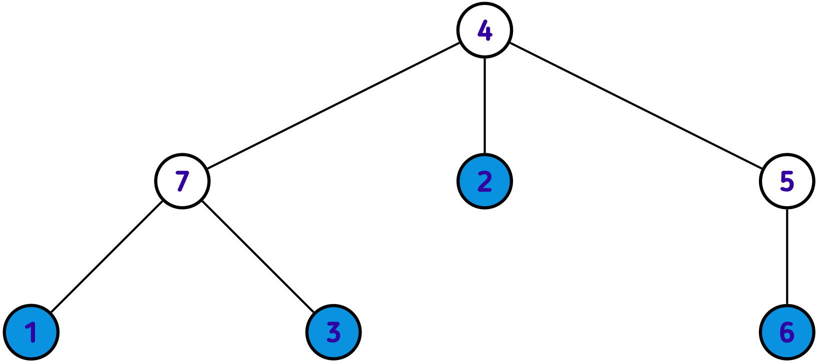
	1	2	3	4	5	6	7
toLeaf$[u]$ =	0	-	-	-	-	-	-
maxLength$[u]$ =	0	-	-	-	-	-	-



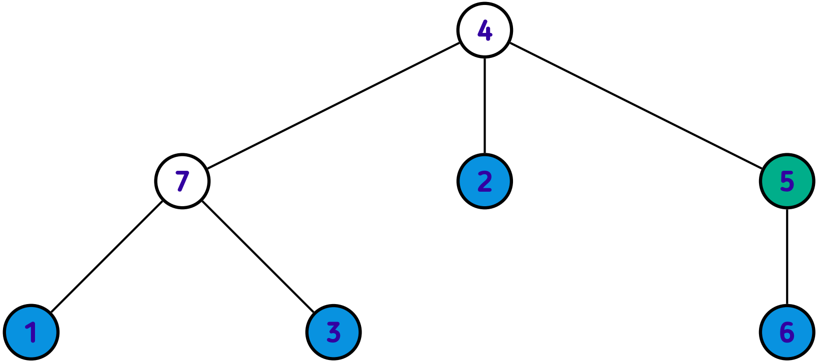
	1	2	3	4	5	6	7
toLeaf$[u]$ =	0	-	**0**	-	-	-	-
maxLength$[u]$ =	0	-	**0**	-	-	-	-



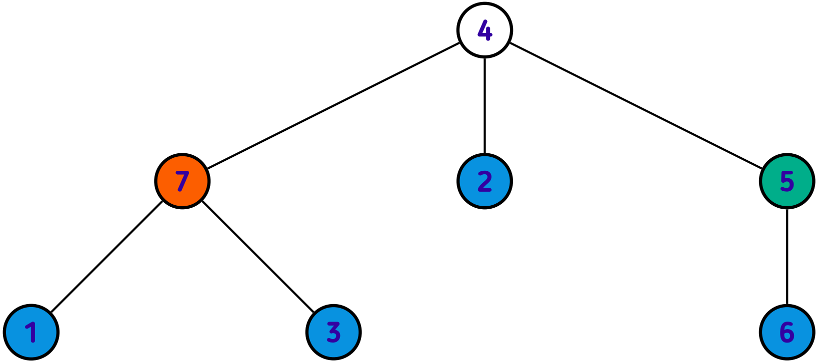
	1	2	3	4	5	6	7
toLeaf$[u]$ =	0	**0**	0	-	-	-	-
maxLength$[u]$ =	0	**0**	0	-	-	-	-



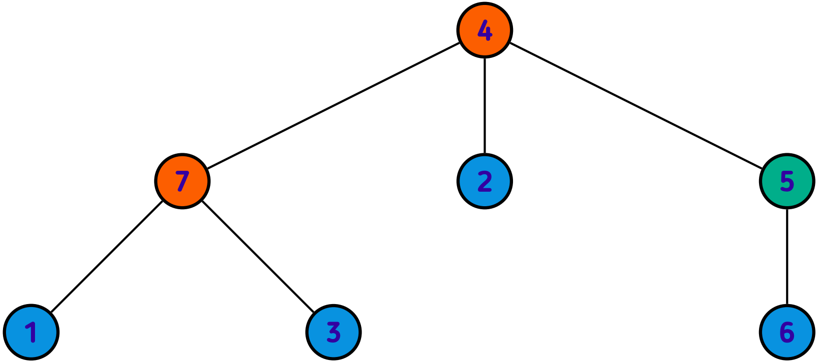
	1	2	3	4	5	6	7
toLeaf$[u]$ =	0	0	0	-	-	0	-
maxLength$[u]$ =	0	0	0	-	-	0	-



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



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



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

```
int diameter(int root, int N)
{
    dfs(root, 0);

    int d = 0;

    for (int u = 1; u <= N; ++u)
        d = max(d, max_length[u]);

    return d;
}
```

```cpp
void dfs(int u, int p)
{
    vector<int> ds;

    for (auto v : adj[u])
    {
        if (v == p)
            continue;

        dfs(v, u);
        ds.push_back(to_leaf[v]);
    }

    sort(ds.begin(), ds.end());

    to_leaf[u] = ds.empty() ? 0 : ds.back() + 1;
```

```
    int N = (int) ds.size();

    switch (N) {
    case 0:
        max_length[u] = 0;
        break;

    case 1:
        max_length[u] = ds.back() + 1;
        break;

    default:
        max_length[u] = ds[N - 1] + ds[N - 2] + 2;
    }
}
```

Maior caminho até uma folha

Maior caminho até uma folha

- $\star$ **Outra aplicação de DFS com DP é o cálculo do tamanho, em arestas, do maior caminho** toLeaf$[u]$ **de** u **até uma folha**

Maior caminho até uma folha

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

$\star$ **Se** u **for uma folha, então** $\textsf{toLeaf}[u] = 0$

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,**

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

Maior caminho até uma folha

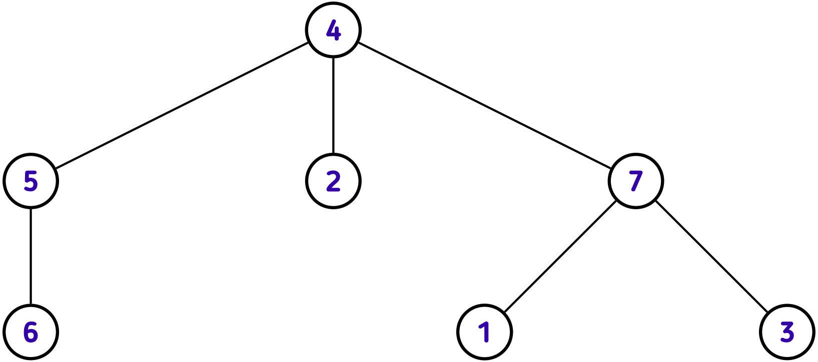
- Outra aplicação de DFS com DP é o cálculo do tamanho, em arestas, do maior caminho $\textsf{toLeaf}[u]$ de u até uma folha

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

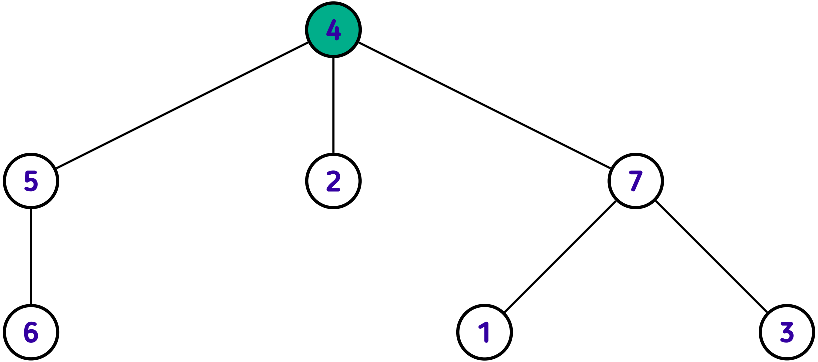
- Caso contrário,

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

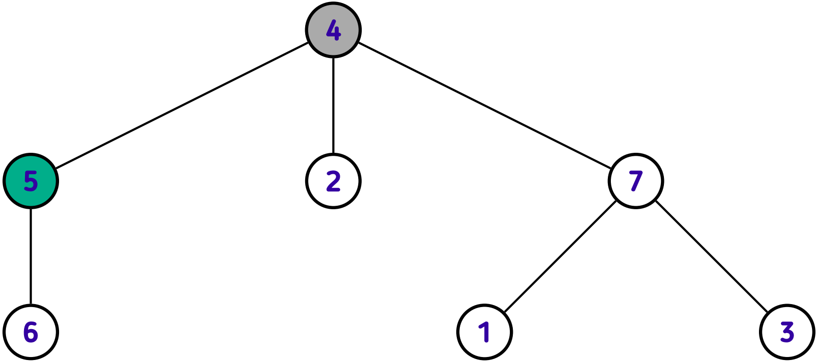
- Este algoritmo pode ser adaptado para computar o tamanho como soma dos pesos das arestas do caminho que vai de u até a folha



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

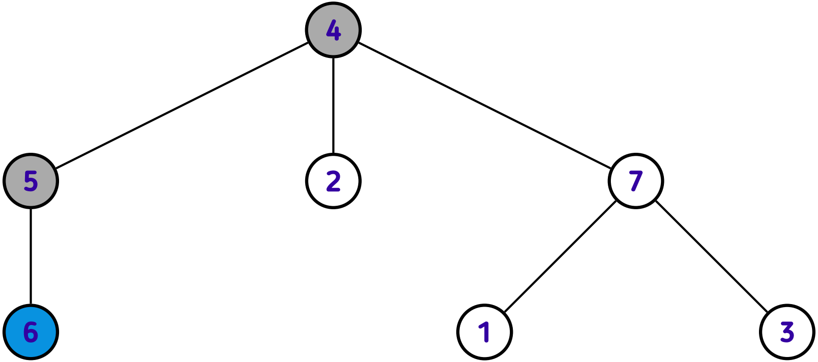


	1	2	3	4	5	6	7
$\text{toLeaf}[u] =$	-	-	-	-	-	-	-

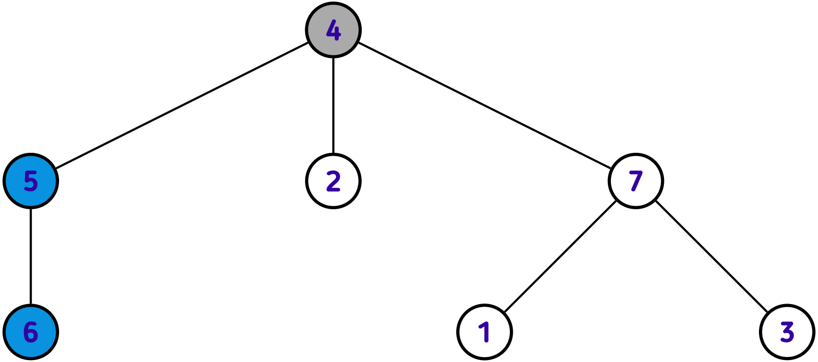


toLeaf$[u] =$

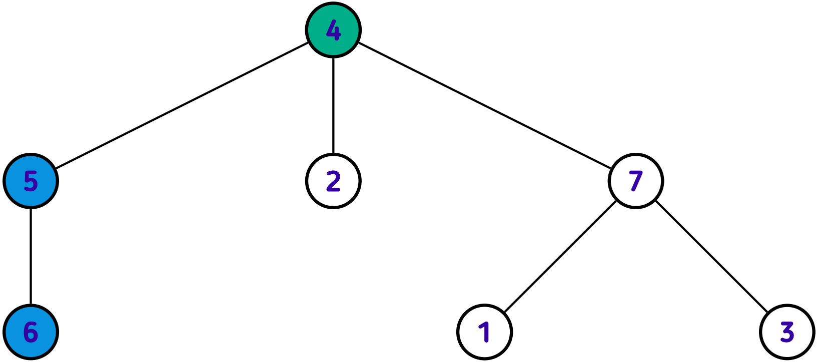
1	2	3	4	5	6	7
-	-	-	-	-	-	-



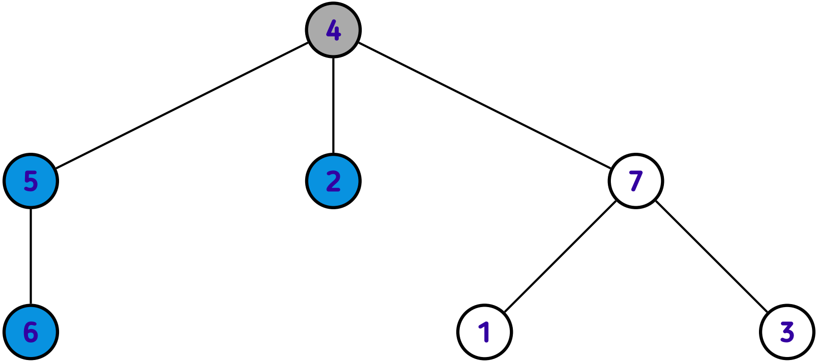
$\texttt{toLeaf}[u] =$	1	2	3	4	5	6	7
	-	-	-	-	-	0	-



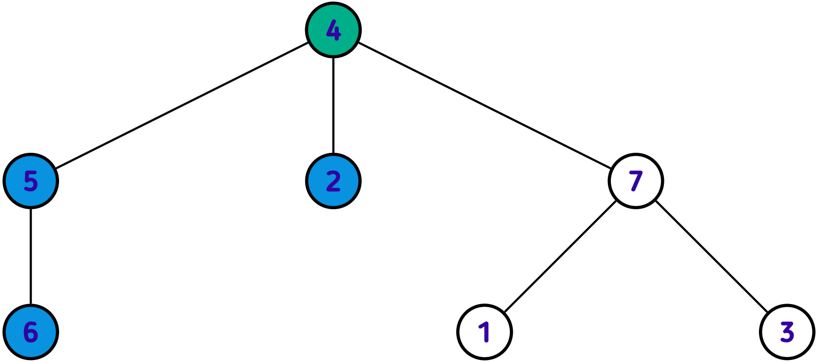
	1	2	3	4	5	6	7
toLeaf$[u]$ =	-	-	-	-	1	0	-



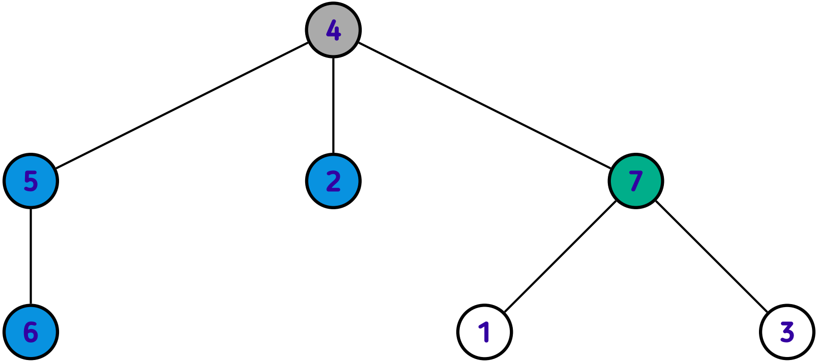
	1	2	3	4	5	6	7
toLeaf$[u]$ =	-	-	-	2	1	0	-



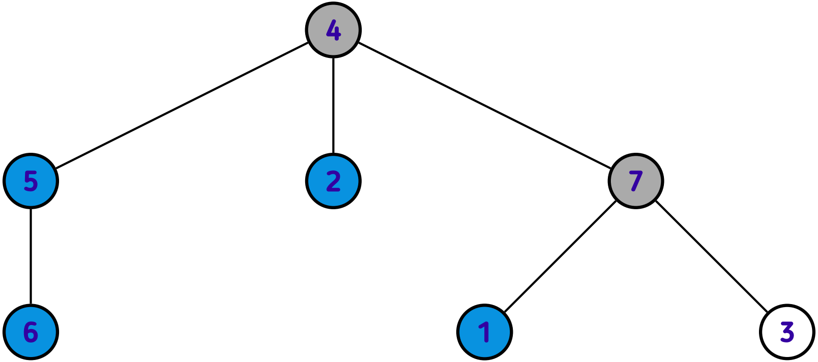
	1	2	3	4	5	6	7
toLeaf$[u]$ =	-	0	-	2	1	0	-



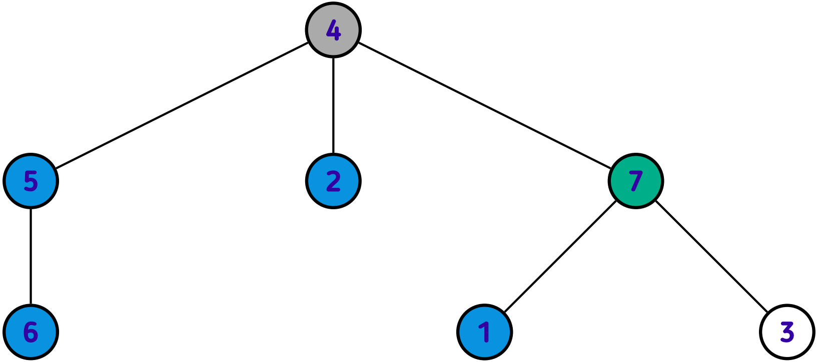
	1	2	3	4	5	6	7
$\text{toLeaf}[u] =$	-	0	-	2	1	0	-



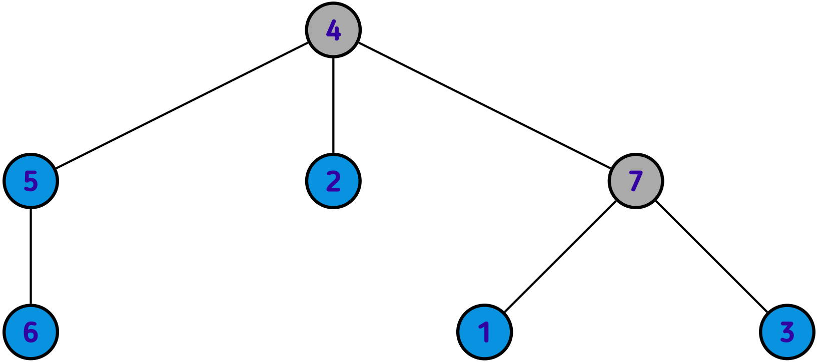
	1	2	3	4	5	6	7
toLeaf$[u] =$	-	0	-	2	1	0	-



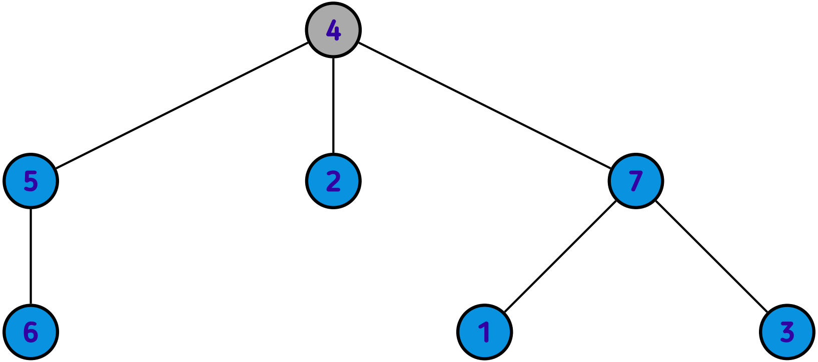
	1	2	3	4	5	6	7
toLeaf$[u]$ =	**0**	0	-	2	1	0	-



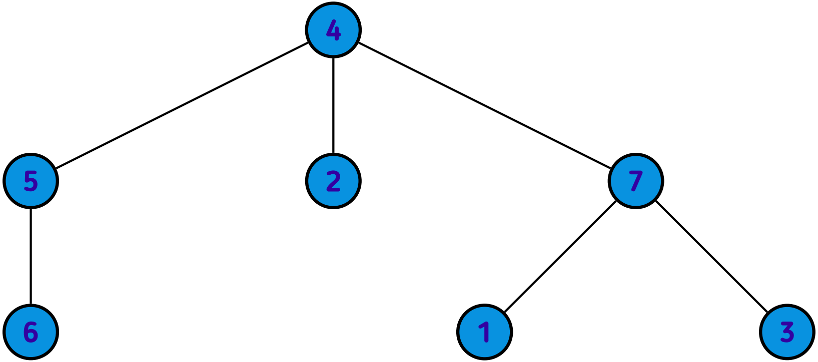
	1	2	3	4	5	6	7
toLeaf$[u]$ =	0	0	-	2	1	0	**0**



	1	2	3	4	5	6	7
$\mathsf{toLeaf}[u] =$	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

Problemas sugeridos

1. AtCoder Beginner Contest 126 – Problem D: Even Relation
2. Codeforces Beta Round #87 (Div. 1 Only) – Problem A: Party
3. Codeforces Round #660 (Div. 2) – Problem C: Uncle Bogdan and Country Happiness
4. OJ 10459 – The Tree Root

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.**