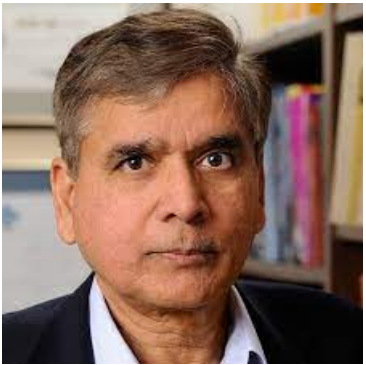
Grafos

Algoritmo de Kosaraju

Prof. Edson Alves

Faculdade UnB Gama

Proponente



S. Rao Kosaraju
(1978)

Características do algoritmo de Kosaraju

Características do algoritmo de Kosaraju

- O algoritmo de Kosaraju encontra os componentes fortemente conectados de um grafo direcionado

Características do algoritmo de Kosaraju

- O algoritmo de Kosaraju encontra os componentes fortemente conectados de um grafo direcionado
- O algoritmo realiza duas buscas em profundidade

Características do algoritmo de Kosaraju

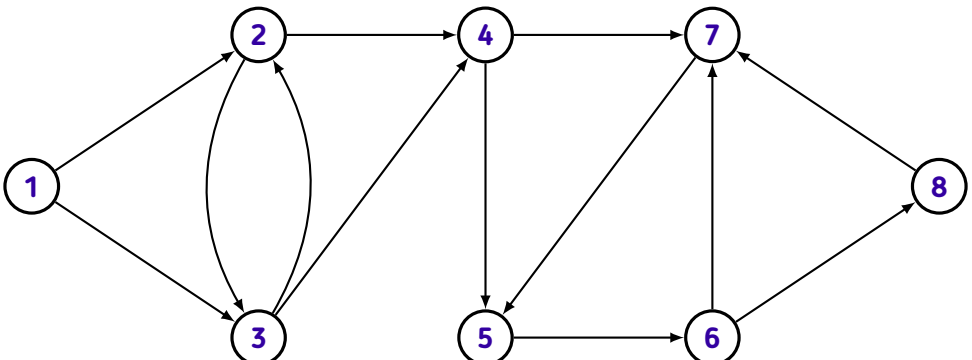
- O algoritmo de Kosaraju encontra os componentes fortemente conectados de um grafo direcionado
- O algoritmo realiza duas buscas em profundidade
- A primeira busca constrói uma lista de vértices, na ordem em que foram processados na DFS

Características do algoritmo de Kosaraju

- O algoritmo de Kosaraju encontra os componentes fortemente conectados de um grafo direcionado
- O algoritmo realiza duas buscas em profundidade
- A primeira busca constrói uma lista de vértices, na ordem em que foram processados na DFS
- A segunda busca identifica os componentes fortemente conectados do grafo

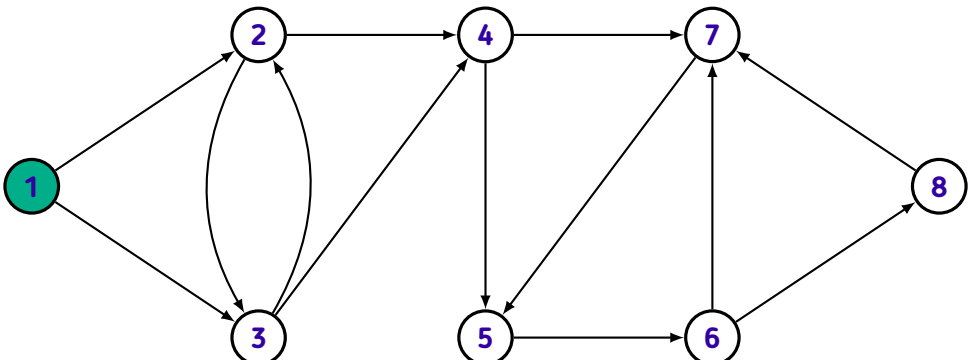
Primeira busca

$O = \{\ \}$



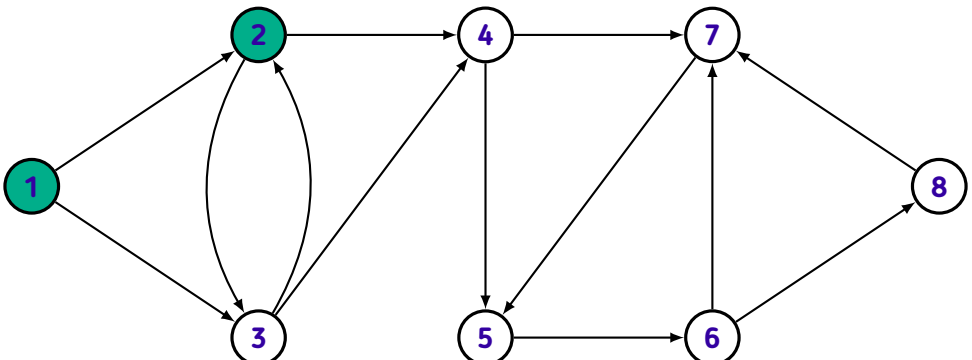
Primeira busca

$O = \{\ \}$



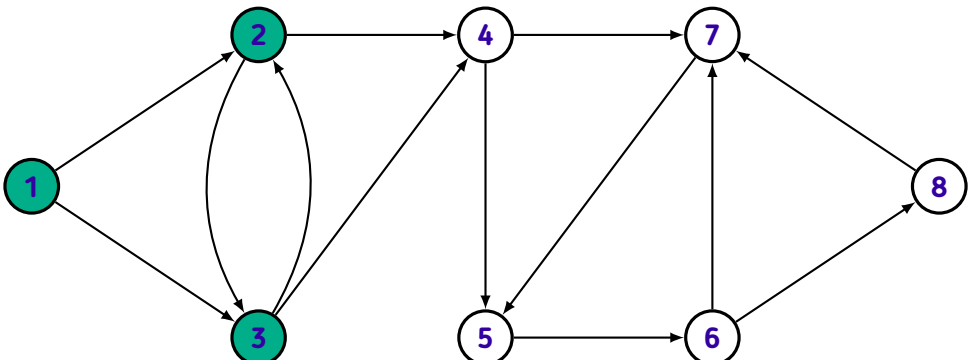
Primeira busca

$O = \{\ \}$



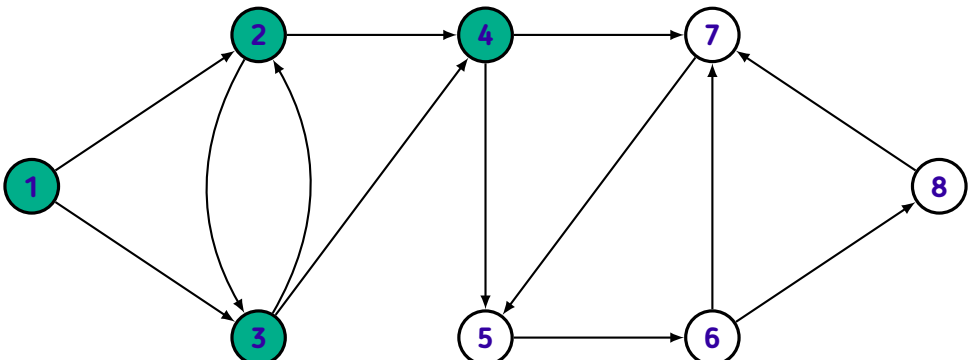
Primeira busca

$O = \{\ \}$



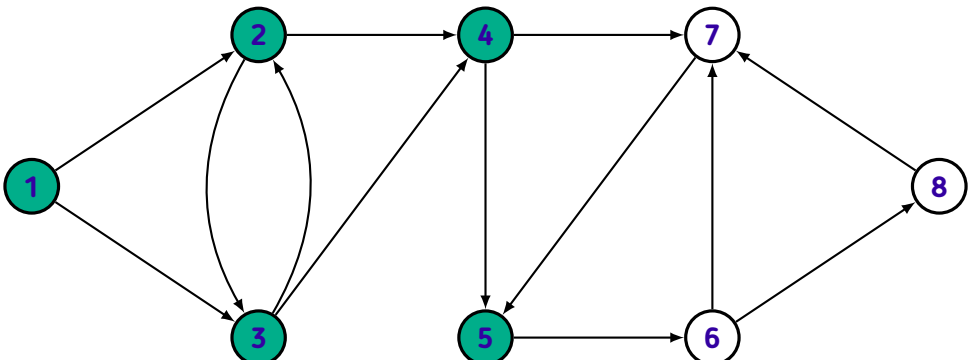
Primeira busca

$O = \{\ \}$



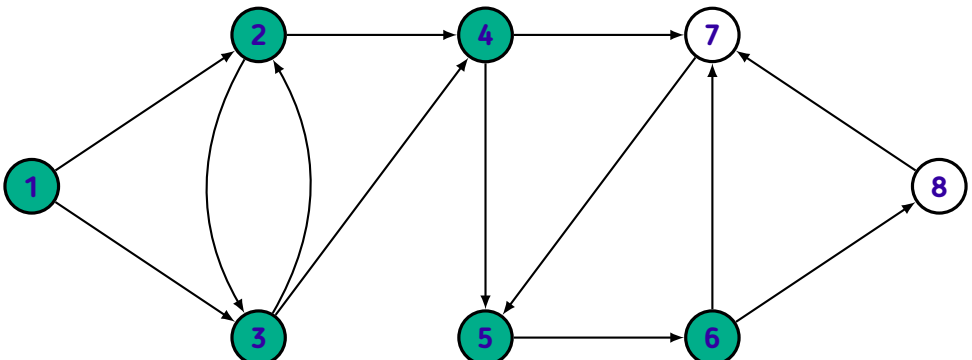
Primeira busca

$O = \{\ \}$



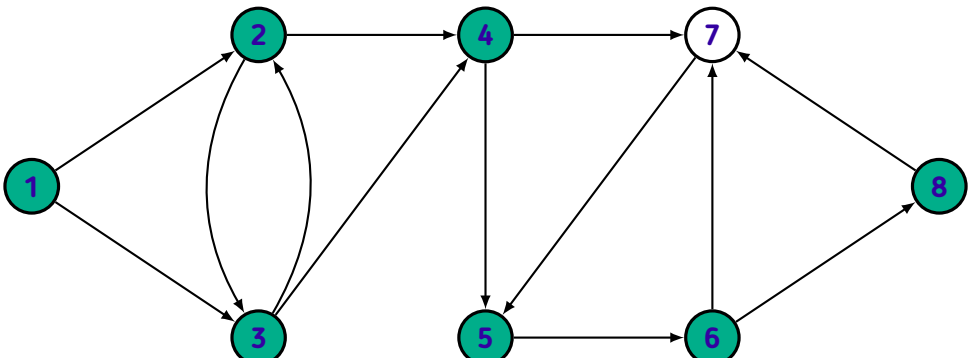
Primeira busca

$O = \{\ \}$



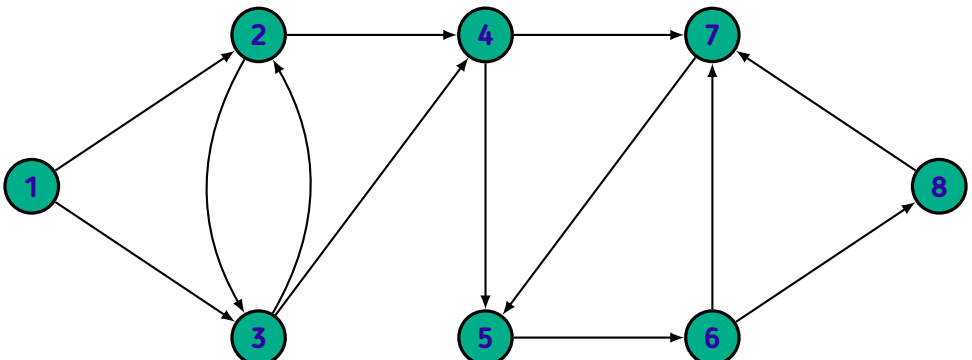
Primeira busca

$O = \{\ \}$



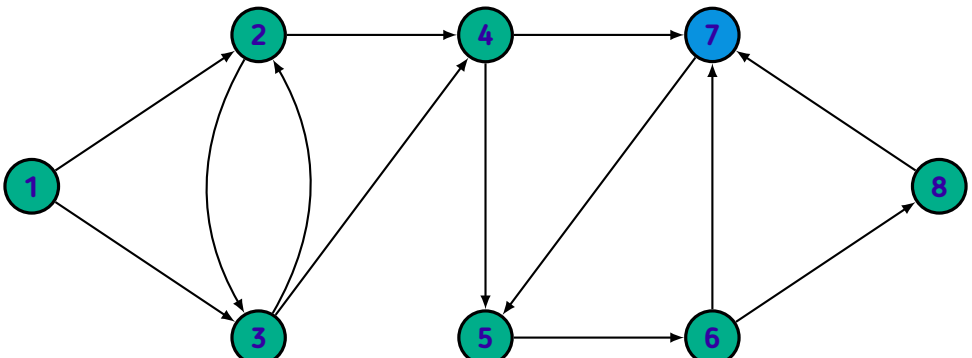
Primeira busca

$O = \{\ \}$



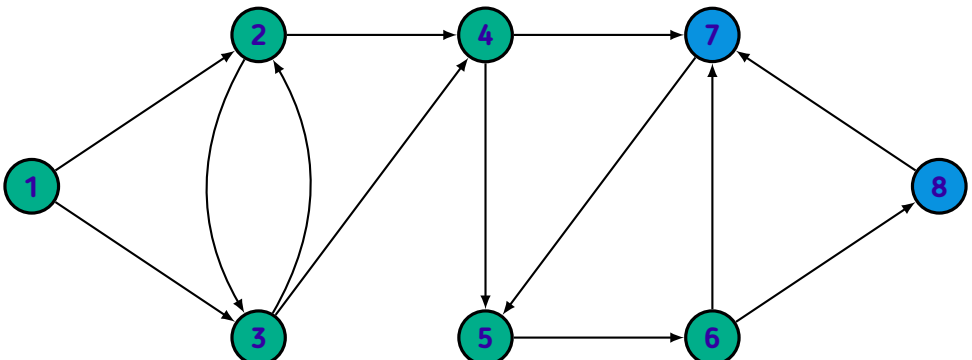
Primeira busca

$$O = \{\ 7\ \}$$



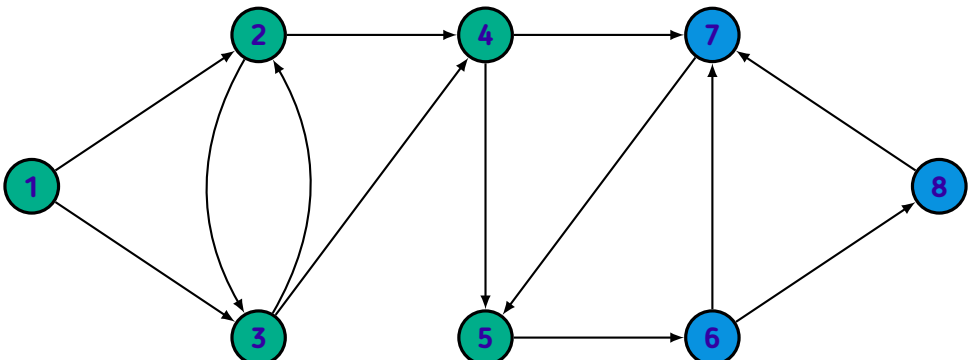
Primeira busca

$$O = \{\ 7, 8\ \}$$



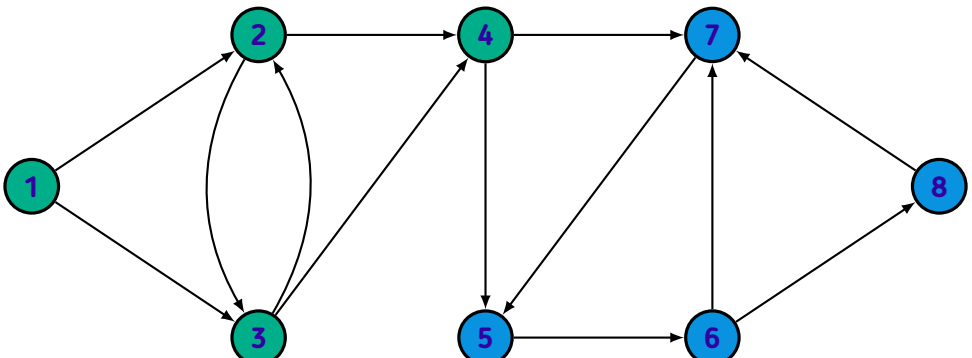
Primeira busca

$$O = \{\ 7, 8, 6\ \}$$



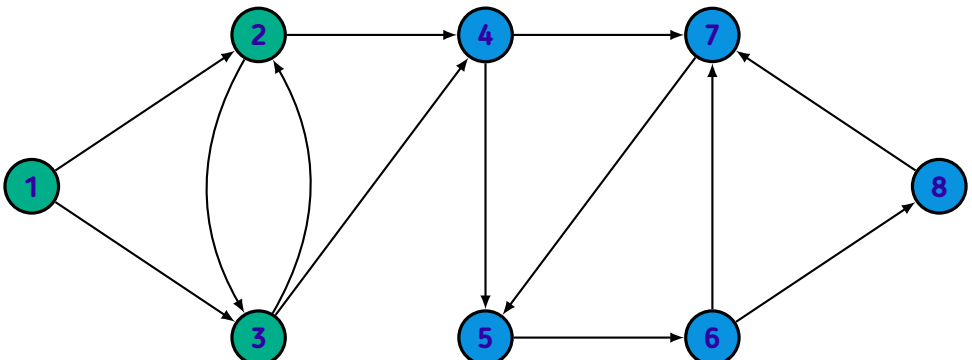
Primeira busca

$$O = \{\ 7, 8, 6, 5\ \}$$



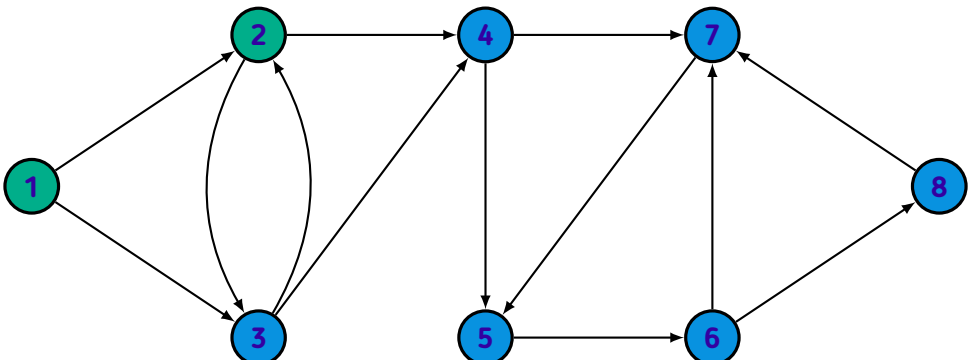
Primeira busca

$$O = \{\ 7, 8, 6, 5, 4\ \}$$



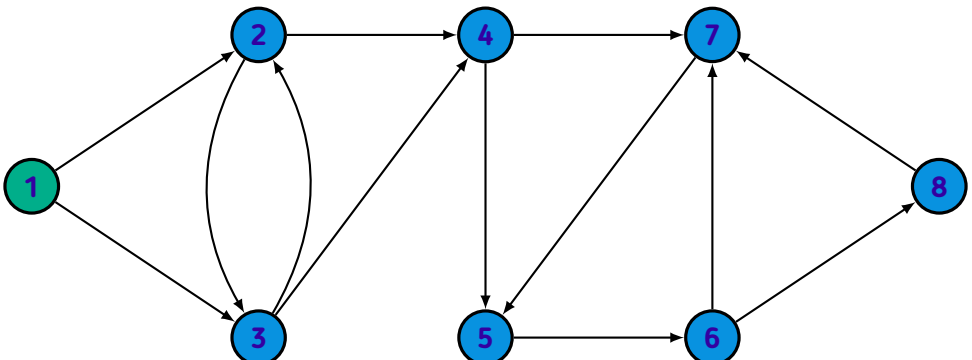
Primeira busca

$$O = \{\ 7, 8, 6, 5, 4, 3\ \}$$



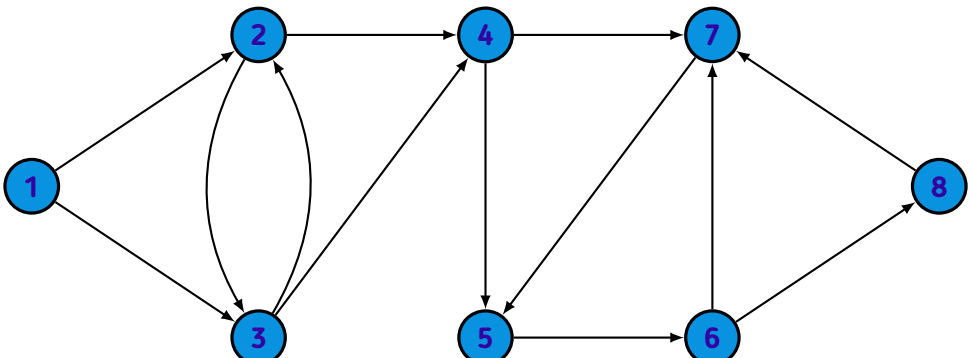
Primeira busca

$$O = \{\ 7, 8, 6, 5, 4, 3, 2\ \}$$



Primeira busca

$$O = \{\ 7, 8, 6, 5, 4, 3, 2, 1\ \}$$

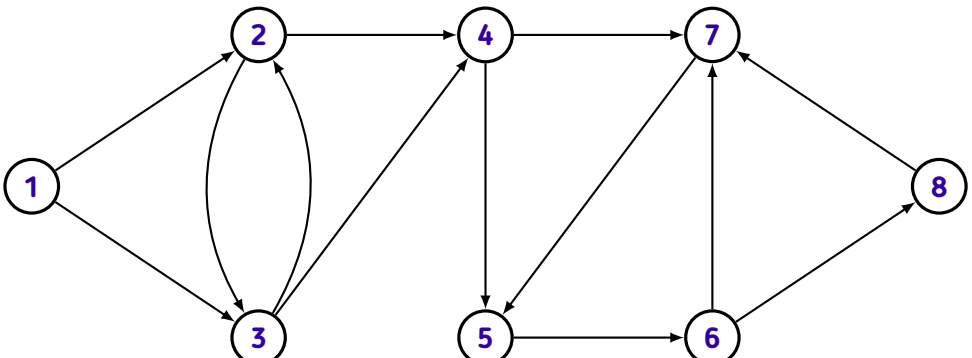


```
vector<int> dfs_order(int N)
{
    vector<int> order;

    for (int u = 1; u <= N; ++u)
        dfs(u, order);

    return order;
}
```

```cpp
void dfs(int u, vector<int>& order)
{
    if (visited[u])
        return;

    visited[u] = true;

    for (auto v : adj[u])
        dfs(v, order);

    order.emplace_back(u);
}
```

Segunda busca

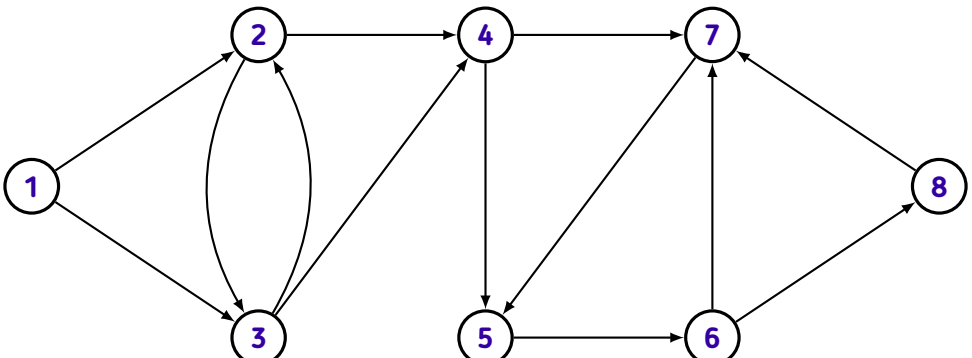
Passo #1: reverta a ordem da DFS



Segunda busca

Passo #1: reverta a ordem da DFS

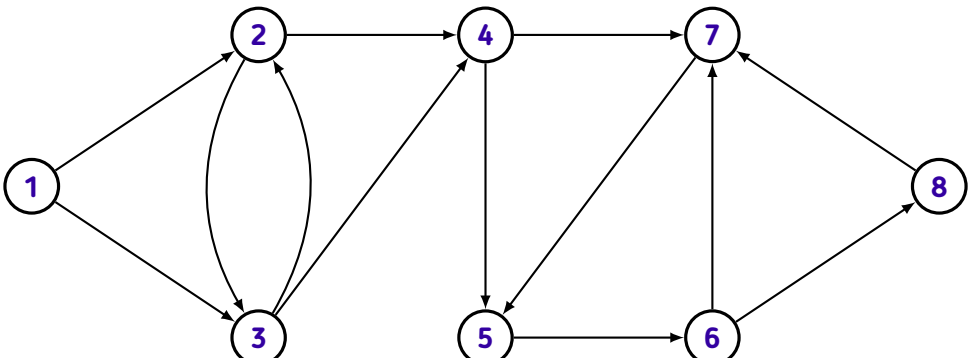
$$O = \{\ 7, 8, 6, 5, 4, 3, 2, 1\ \}$$



Segunda busca

Passo #1: reverta a ordem da DFS

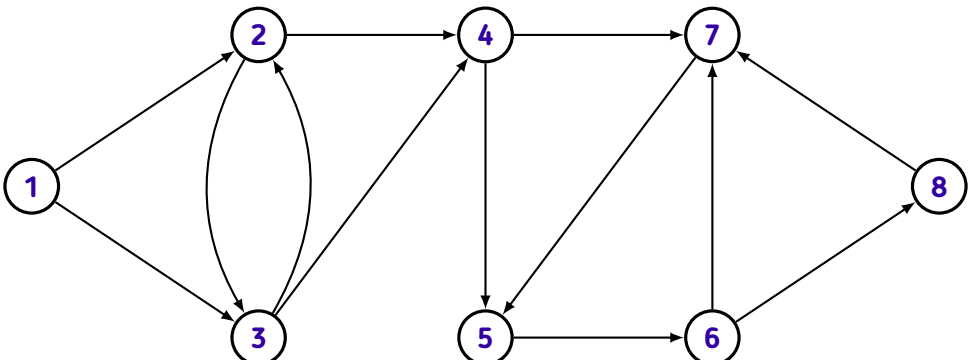
$$O = \{\ 1, 2, 3, 4, 5, 6, 8, 7\ \}$$



Segunda busca

Passo #2: reverta as arestas do grafo

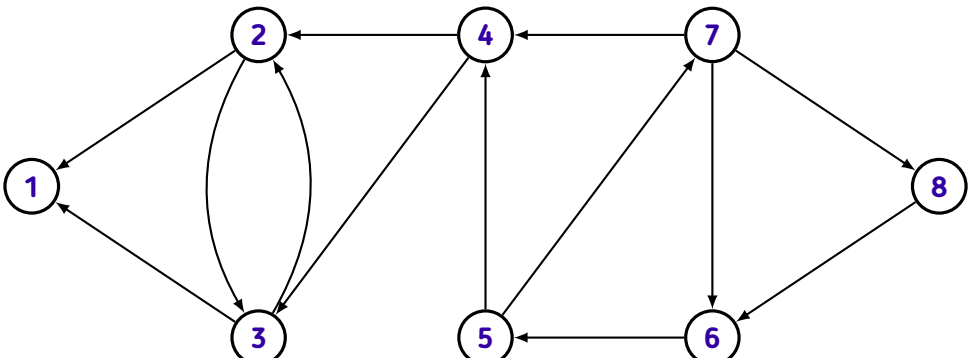
$$O = \{\ 1, 2, 3, 4, 5, 6, 8, 7\ \}$$



Segunda busca

Passo #2: reverta as arestas do grafo

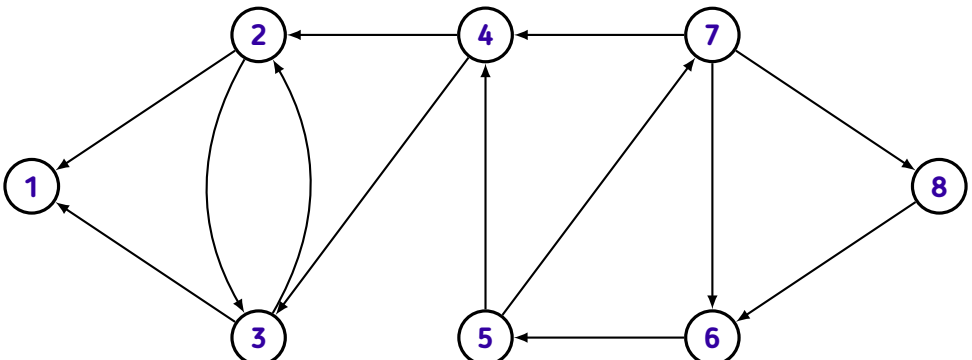
$$O = \{\ 1, 2, 3, 4, 5, 6, 8, 7\ \}$$



Segunda busca

Passo #3: identifique os componentes

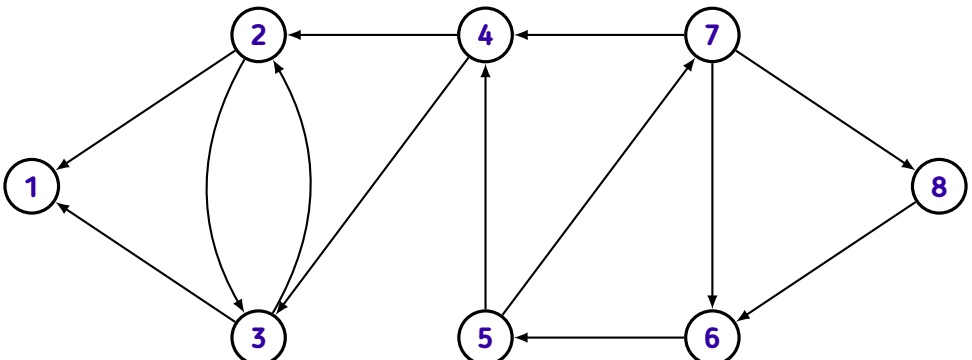
$$O = \{\ 1, 2, 3, 4, 5, 6, 8, 7\ \}$$



Segunda busca

Passo #3: identifique os componentes

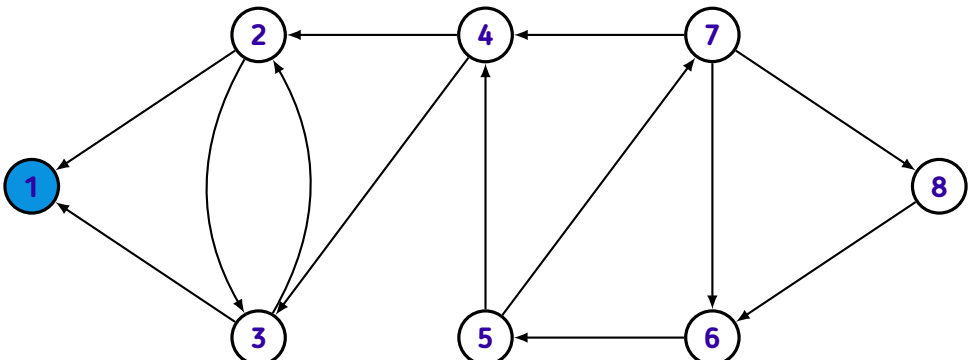
$O = \{\ \mathbf{1}, 2, 3, 4, 5, 6, 8, 7\ \}$



Segunda busca

Passo #3: identifique os componentes

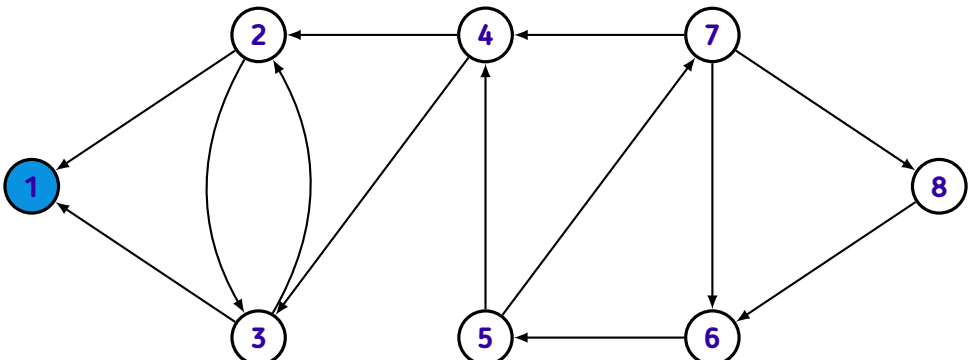
$$O = \{\ \mathbf{1}, 2, 3, 4, 5, 6, 8, 7\ \}$$



Segunda busca

Passo #3: identifique os componentes

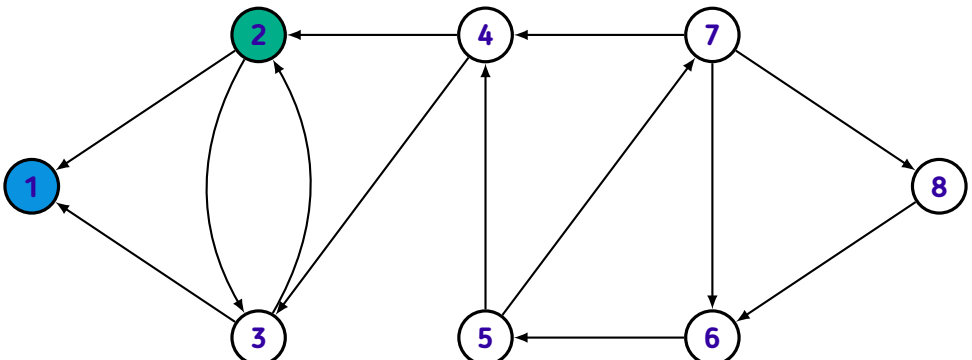
$$O = \{\ 1, \mathbf{2}, 3, 4, 5, 6, 8, 7\ \}$$



Segunda busca

Passo #3: identifique os componentes

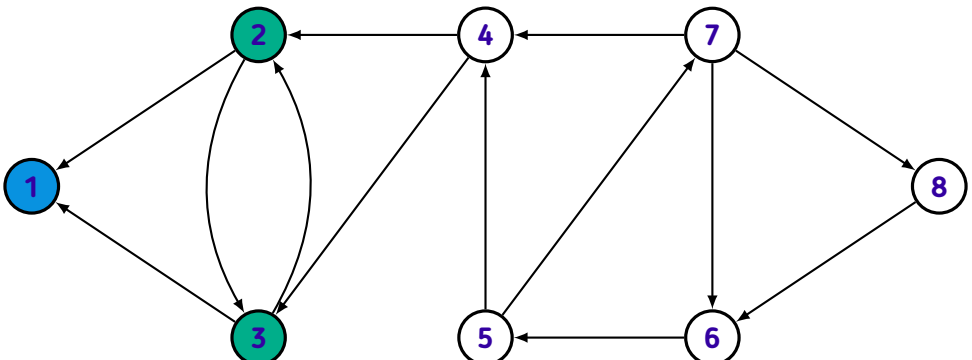
$$O = \{\ 1, \mathbf{2}, 3, 4, 5, 6, 8, 7\ \}$$



Segunda busca

Passo #3: identifique os componentes

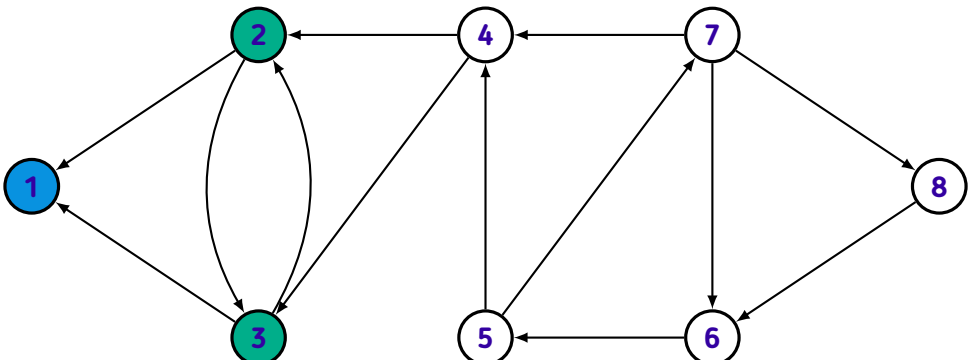
$$O = \{\ 1, \mathbf{2}, 3, 4, 5, 6, 8, 7\ \}$$



Segunda busca

Passo #3: identifique os componentes

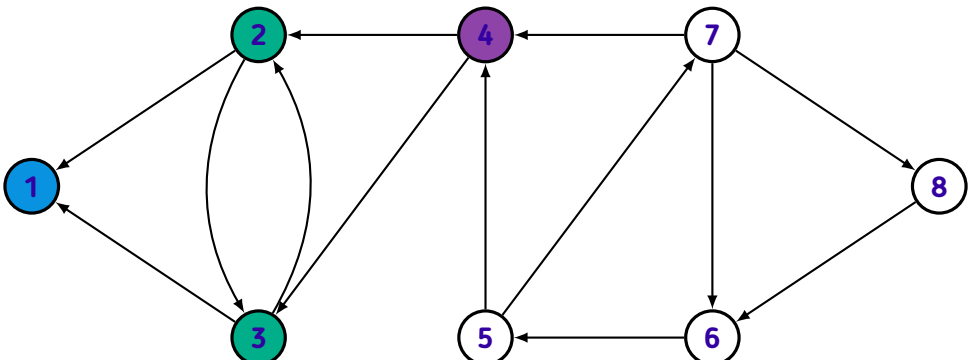
$$O = \{\ 1, 2, 3, \mathbf{4}, 5, 6, 8, 7\ \}$$



Segunda busca

Passo #3: identifique os componentes

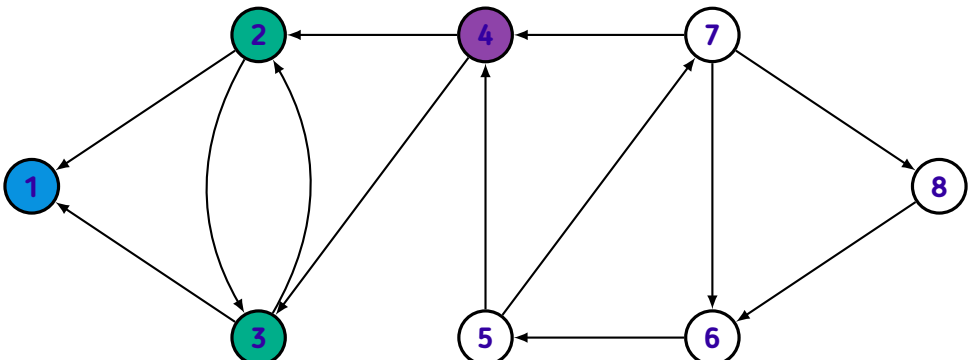
$$O = \{\ 1, 2, 3, \mathbf{4}, 5, 6, 8, 7\ \}$$



Segunda busca

Passo #3: identifique os componentes

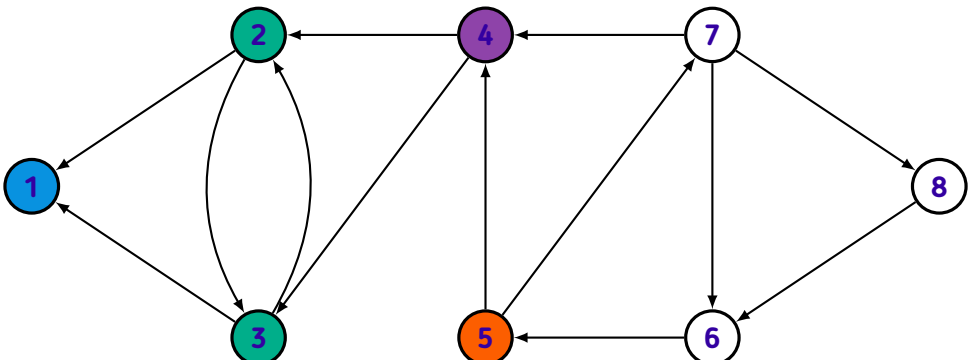
$$O = \{\ 1, 2, 3, 4, \mathbf{5}, 6, 8, 7\ \}$$



Segunda busca

Passo #3: identifique os componentes

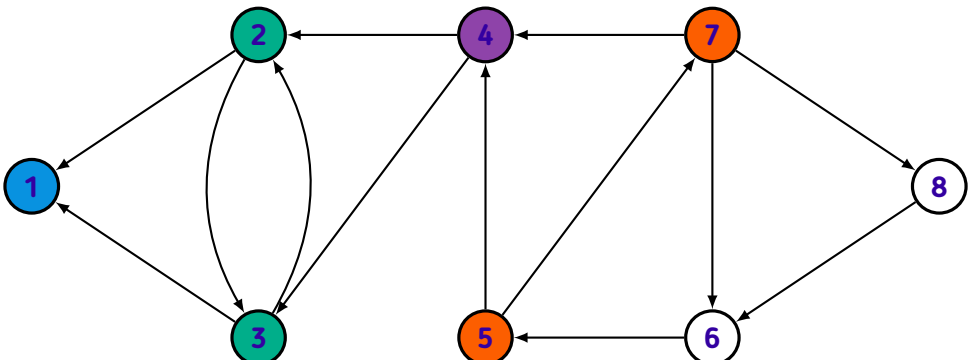
$$O = \{\ 1, 2, 3, 4, \mathbf{5}, 6, 8, 7\ \}$$



Segunda busca

Passo #3: identifique os componentes

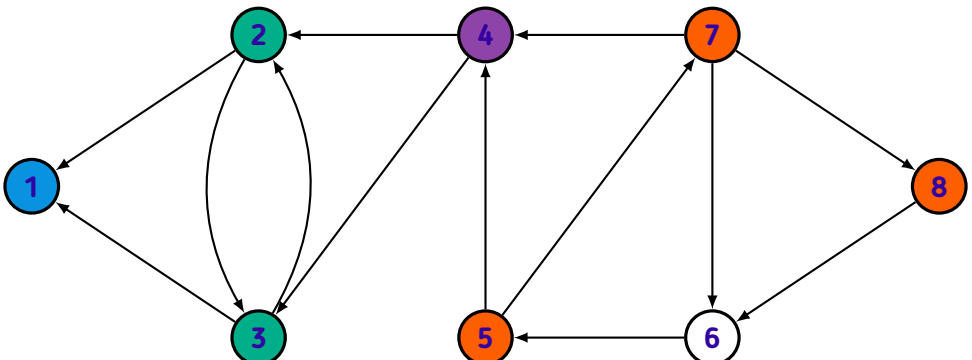
$$O = \{\ 1, 2, 3, 4, \mathbf{5}, 6, 8, 7\ \}$$



Segunda busca

Passo #3: identifique os componentes

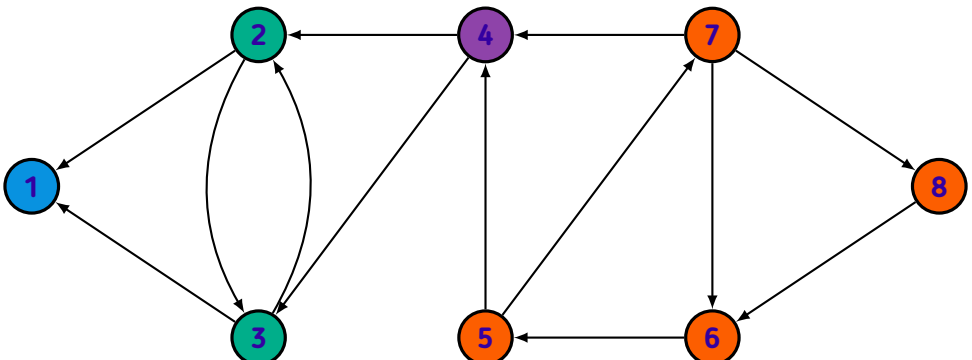
$$O = \{\ 1, 2, 3, 4, \mathbf{5}, 6, 8, 7\ \}$$



Segunda busca

Passo #3: identifique os componentes

$$O = \{\ 1, 2, 3, 4, \mathbf{5}, 6, 8, 7\ \}$$



```
vector<vector<int>> kosaraju(int N) {
    auto order = dfs_order(N);
    reverse(order.begin(), order.end());

    for (int u = 1; u <= N; ++u)
        for (auto v : adj[u])
            rev[v].emplace_back(u);

    vector<vector<int>> cs;
    visited.reset();

    for (auto u : order) {
        if (visited[u])
            continue;

        cs.emplace_back(vector<int>());
        dfs_cc(u, cs.back());
    }

    return cs;
}
```

```
void dfs_cc(int u, vector<int>& cc)
{
    if (visited[u])
        return;

    visited[u] = true;
    cc.emplace_back(u);

    for (auto v : rev[u])
        dfs_cc(v, cc);
}
```

Referências

1. **HALIM**, **Felix**; **HALIM**, **Steve**. ***Competitive Programming 3*, 2010.**

2. **LAAKSONEN**, **Antti**. ***Competitive Programmer's Handbook*, 2018.**