Grafos

Ordenação Topológica

Prof. Edson Alves

Faculdade UnB Gama

Ordenação topológica

Ordenação topológica

Seja $G(V, E)$ um grafo direcionado com N vértices. Uma ordenação $O = \{ v_{i_1}, v_{i_2}, \ldots, v_{i_N} \}$ dos vértices de G é uma **ordenação topológica** se vale a seguinte afirmação: para quaisquer pares de vértices $u, v \in V$, se existe um caminho de u a v, então u antecede v na ordenação O.

Características da ordenação topológica

Características da ordenação topológica

- Grafos que possuem ciclos não possuem ordenações topológicas

Características da ordenação topológica

- Grafos que possuem ciclos não possuem ordenações topológicas

- Um grafo direcionado acíclico (DAG) contém, no mínimo, uma ordenação topológica

Características da ordenação topológica

- ⋆ **Grafos que possuem ciclos não possuem ordenações topológicas**

- ⋆ **Um grafo direcionado acíclico (DAG) contém, no mínimo, uma ordenação topológica**

- ⋆ **Ordenações topológicas estabelecem relações de prioridade: se a tarefa A é pré-requisito da tarefa B, então $A < B$ na ordenação**

Características da ordenação topológica

- ⋆ Grafos que possuem ciclos não possuem ordenações topológicas
- ⋆ Um grafo direcionado acíclico (DAG) contém, no mínimo, uma ordenação topológica
- ⋆ Ordenações topológicas estabelecem relações de prioridade: se a tarefa A é pré-requisito da tarefa B, então $A < B$ na ordenação
- ⋆ O algoritmo de Tarjan determina uma ordenação topológica em um DAG

Características da ordenação topológica

- ⋆ **Grafos que possuem ciclos não possuem ordenações topológicas**
- ⋆ **Um grafo direcionado acíclico (DAG) contém, no mínimo, uma ordenação topológica**
- ⋆ **Ordenações topológicas estabelecem relações de prioridade: se a tarefa A é pré-requisito da tarefa B, então $A < B$ na ordenação**
- ⋆ **O algoritmo de Tarjan determina uma ordenação topológica em um DAG**
- ⋆ **O algoritmo de Kahn também identifica uma ordenação topológica**

Proponente do algoritmo de Tarjan

Robert Endre Tarjan

(1976)

Características do algoritmo de Tarjan

Características do algoritmo de Tarjan

- **O algoritmo de Tarjan determina uma ordenação topológica em um DAG por meio de uma DFS modificada**

Características do algoritmo de Tarjan

- O algoritmo de Tarjan determina uma ordenação topológica em um DAG por meio de uma DFS modificada

- A ideia central é que, na árvore induzida pela DFS, as folhas devem aparecer após os nós intermediários na ordenação topológica

Características do algoritmo de Tarjan

- O algoritmo de Tarjan determina uma ordenação topológica em um DAG por meio de uma DFS modificada

- A ideia central é que, na árvore induzida pela DFS, as folhas devem aparecer após os nós intermediários na ordenação topológica

- Durante a travessia, cada vértice assume um dentre três estados: não encontrado (branco), encontrado (verde) e processado (azul)

Características do algoritmo de Tarjan

- O algoritmo de Tarjan determina uma ordenação topológica em um DAG por meio de uma DFS modificada

- A ideia central é que, na árvore induzida pela DFS, as folhas devem aparecer após os nós intermediários na ordenação topológica

- Durante a travessia, cada vértice assume um dentre três estados: não encontrado (branco), encontrado (verde) e processado (azul)

- Quando um vértice se torna processado, ele deve entrar no início da fila que conterá a ordenação topológica

Pseudocódigo

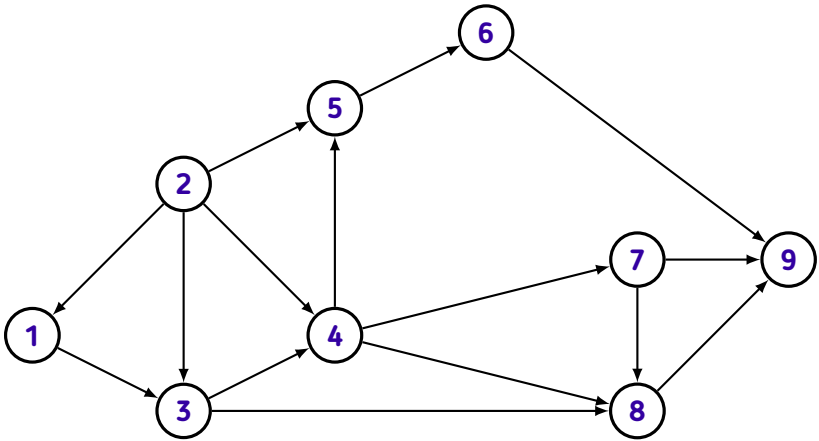
Pseudocódigo

Entrada: um grafo direcionado acíclico $G(V, E)$
Saída: uma ordenação topológica O de G

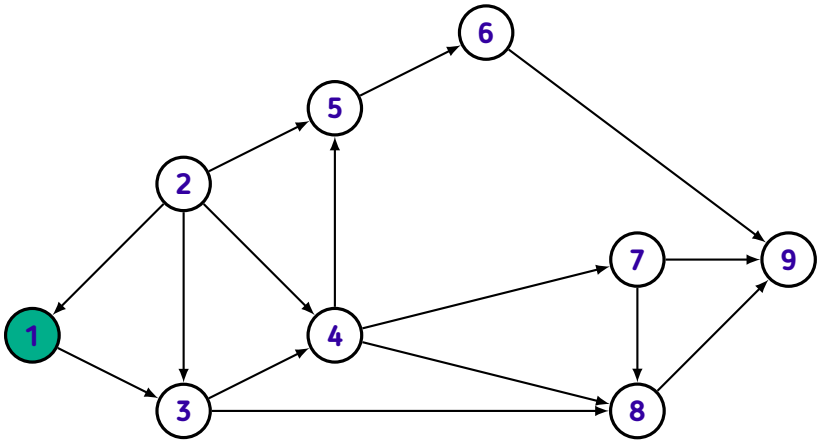
1. Marque todos os vértices $v \in V$ como não encontrado

2. Enquanto existir ao menos um vértice u não encontrado:
   - (a) Marque u como encontrado
   - (b) Prossiga a travessia em todos seus filhos de u
   - (c) Marque u como processado e insira u na frente da fila O

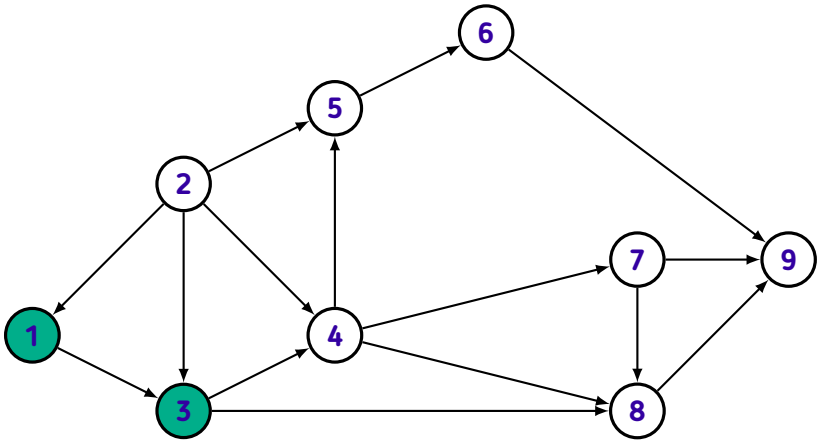
3. Retorne O



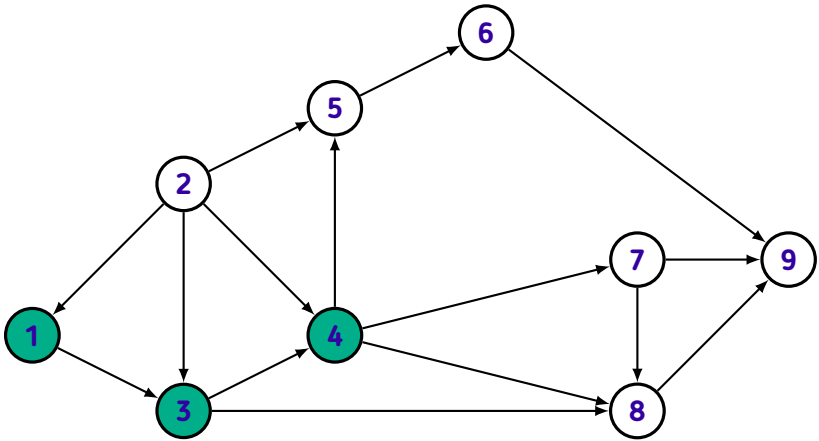
$O = \{\ \}$



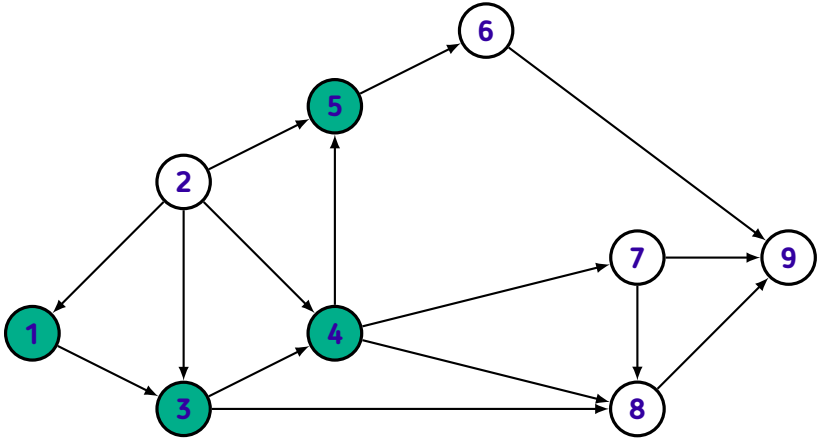
$O = \{\ \}$



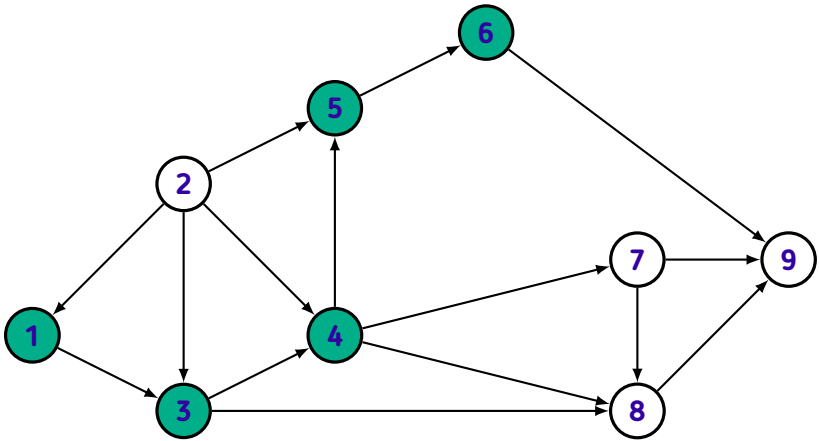
$O = \{\ \}$



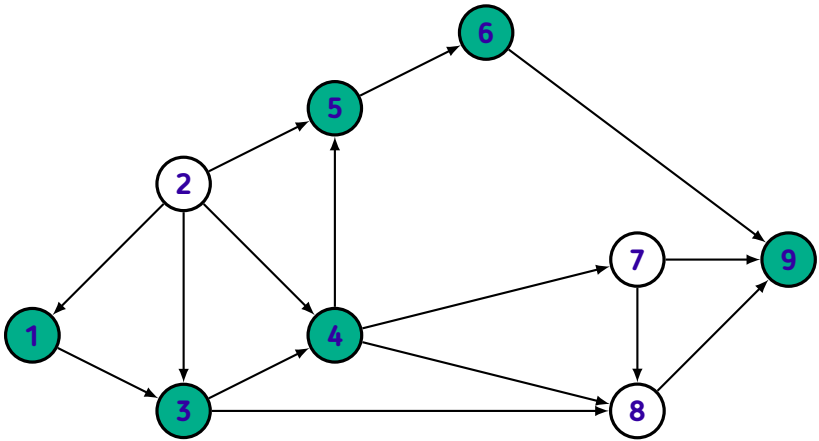
$O = \{\ \}$



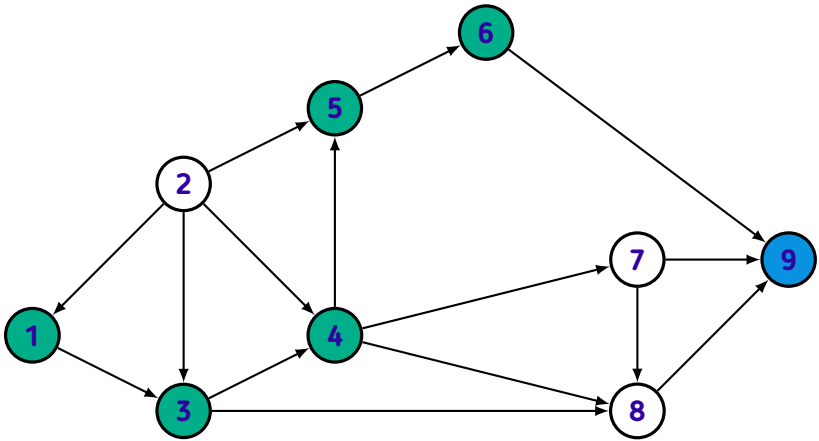
$O = \{\ \}$



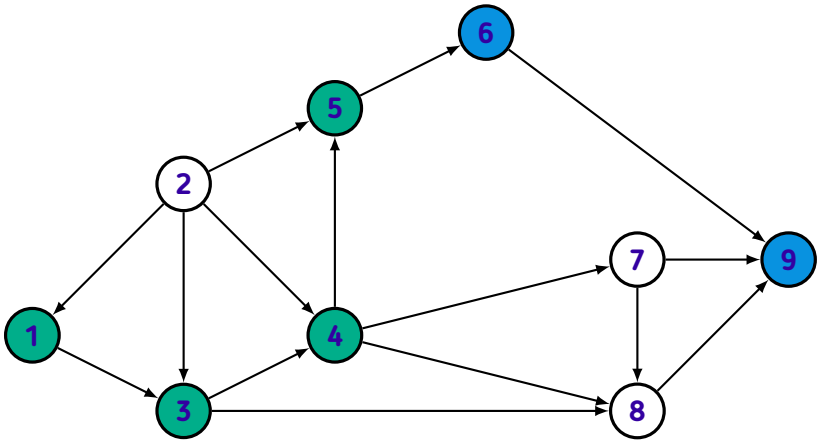
$O = \{\ \}$



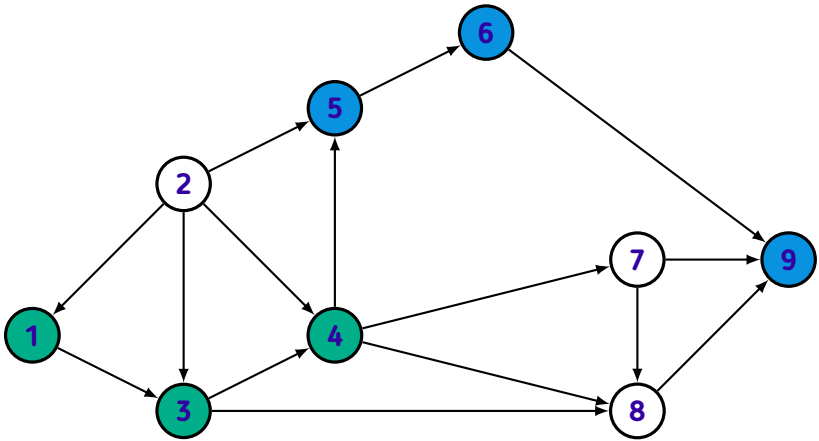
$O = \{\ \}$



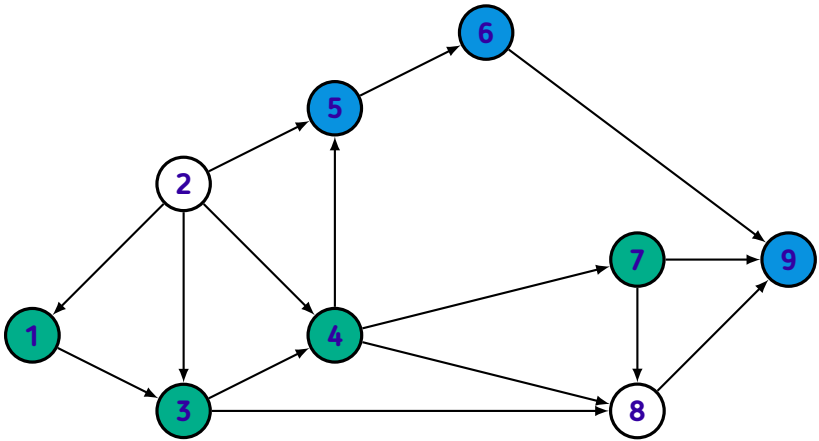
$O = \{ \mathbf{9} \}$



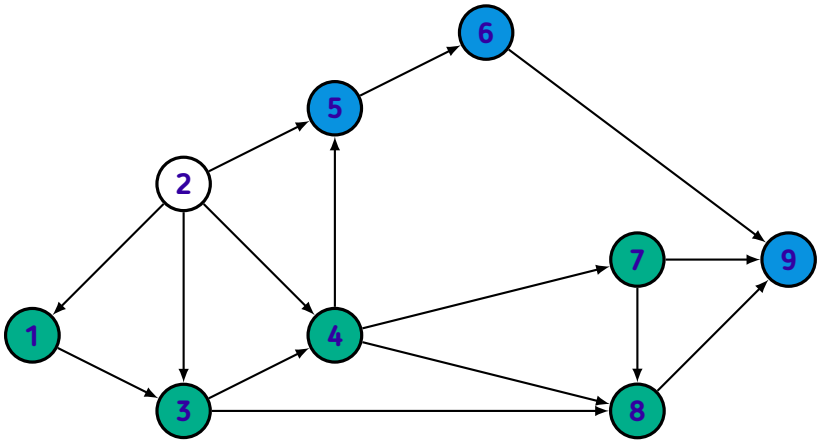
$O = \{$ **6, 9** $\}$



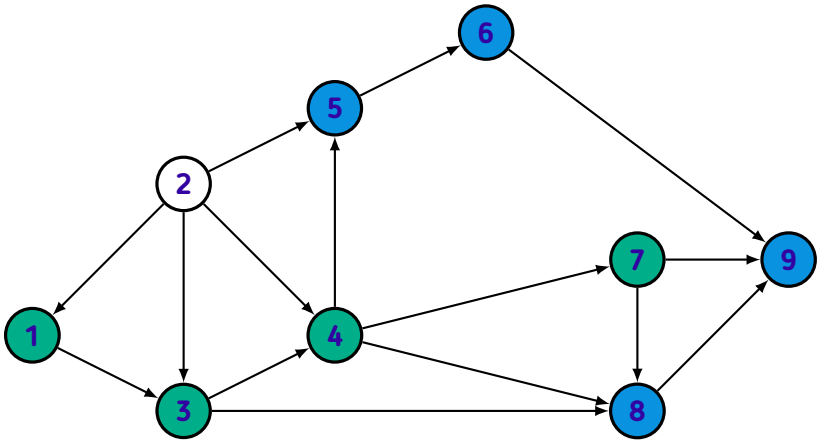
$O = \{$ **5, 6, 9** $\}$



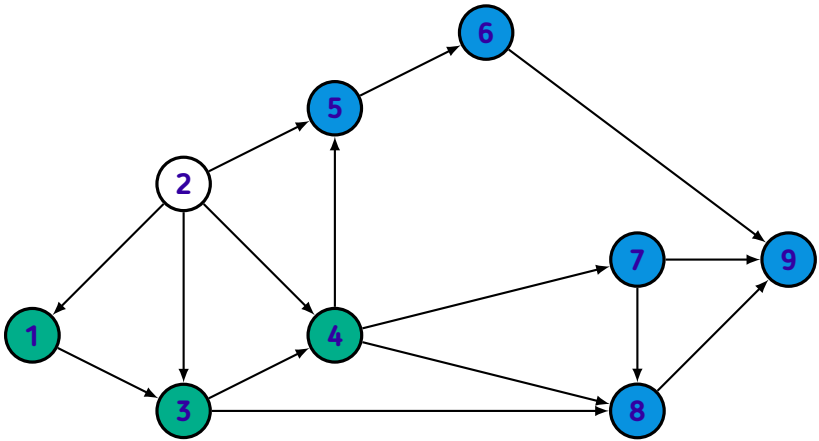
$O = \{$ **5, 6, 9** $\}$



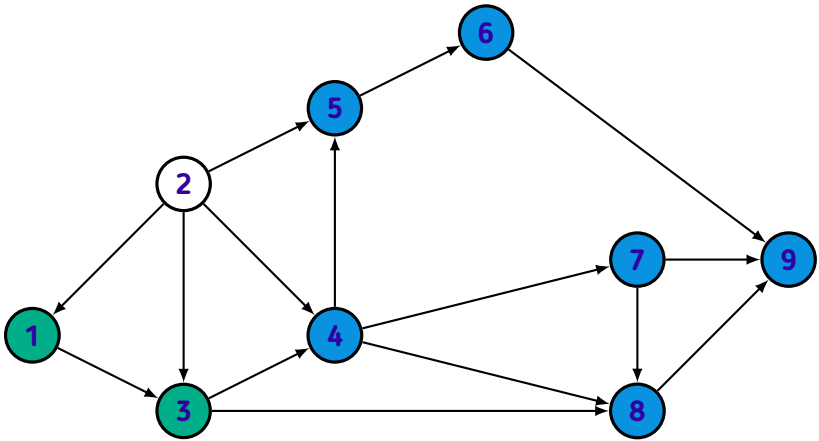
$O = \{$ **5, 6, 9** $\}$



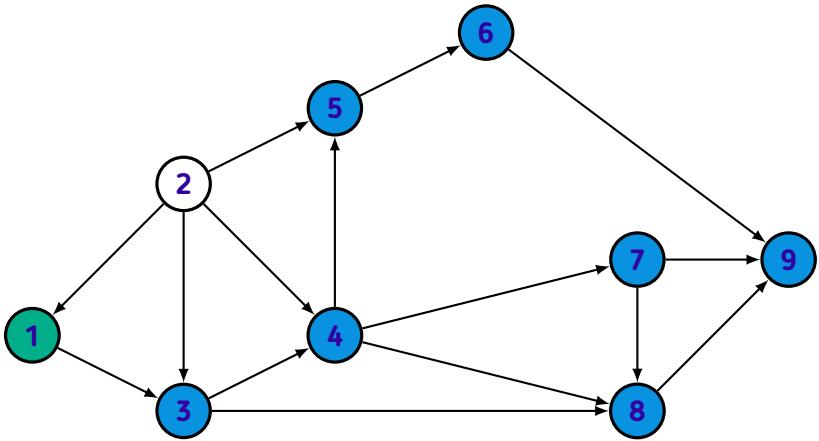
$O = \{$ **8, 5, 6, 9** $\}$



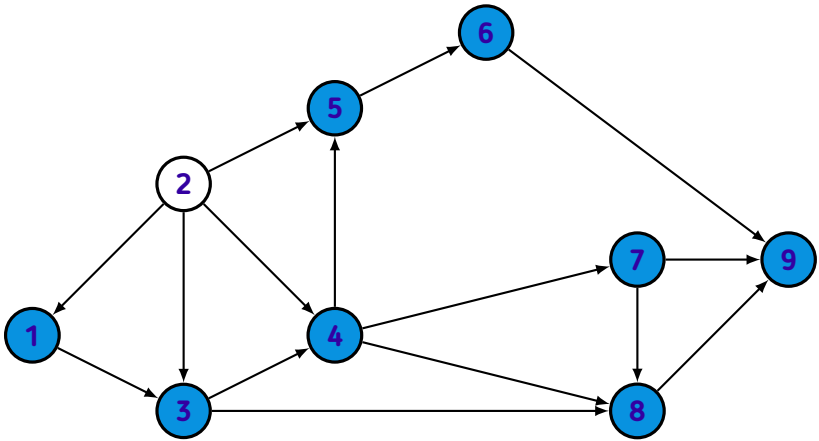
$O = \{$ **7, 8, 5, 6, 9** $\}$



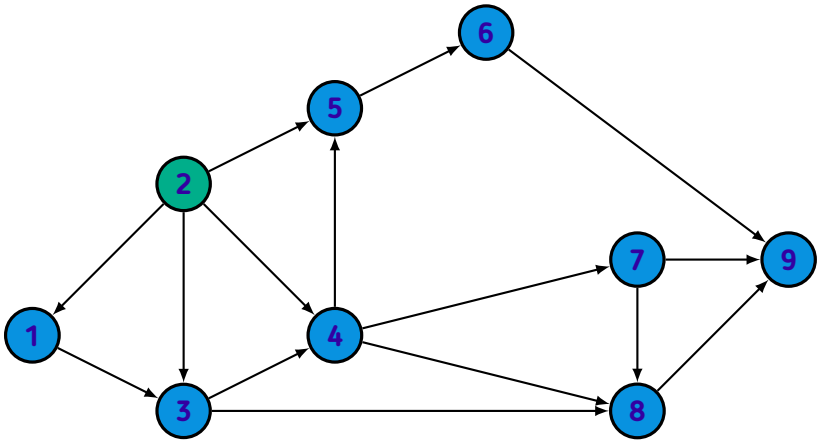
$O = \{$ **4, 7, 8, 5, 6, 9** $\}$



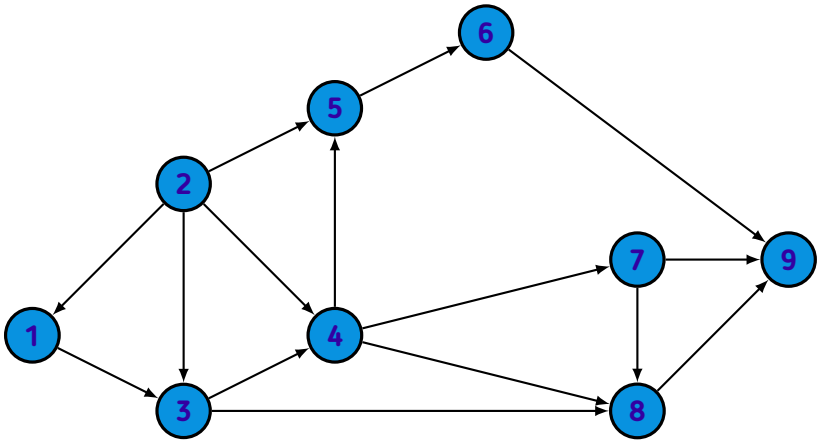
$O = \{$ **3, 4, 7, 8, 5, 6, 9** $\}$



$O = \{$ **1, 3, 4, 7, 8, 5, 6, 9** $\}$



$O = \{$ **1, 3, 4, 7, 8, 5, 6, 9** $\}$



$O = \{$ **2, 1, 3, 4, 7, 8, 5, 6, 9** $\}$

```
vector<int> topological_sort(int N)
{
    vector<int> o, state(N + 1, NOT_FOUND);


    for (int u = 1; u <= N; ++u)
        if (state[u] == NOT_FOUND and not dfs(u, o, state))
            return { };

    reverse(o.begin(), o.end());

    return o;
}
```

```cpp
bool dfs(int u, vector<int>& o, vector<int>& state)
{
    if (state[u] == PROCESSED)
        return true;

    if (state[u] == FOUND)
        return false;

    state[u] = FOUND;

    for (auto v : adj[u])
        if (not dfs(v, o, state))
            return false;

    state[u] = PROCESSED;
    o.emplace_back(u);

    return true;
}
```

Algoritmo de Kahn

Algoritmo de Kahn

- O algoritmo de Kahn encontra uma ordenação topológica em um DAG

Algoritmo de Kahn

- O algoritmo de Kahn encontra uma ordenação topológica em um DAG
- Foi proposto por Arthur B. Kahn em 1962

Algoritmo de Kahn

- O algoritmo de Kahn encontra uma ordenação topológica em um DAG
- Foi proposto por Arthur B. Kahn em 1962
- A ideia central é que os vértices com grau de entrada menor aparecem antes dos vértices com grau de entrada maior na ordenação topológica

Algoritmo de Kahn

- O algoritmo de Kahn encontra uma ordenação topológica em um DAG
- Foi proposto por Arthur B. Kahn em 1962
- A ideia central é que os vértices com grau de entrada menor aparecem antes dos vértices com grau de entrada maior na ordenação topológica
- Para identificar tais vértices, é utilizada uma BFS modificada

Algoritmo de Kahn

- ⋆ **O algoritmo de Kahn encontra uma ordenação topológica em um DAG**
- ⋆ **Foi proposto por Arthur B. Kahn em 1962**
- ⋆ **A ideia central é que os vértices com grau de entrada menor aparecem antes dos vértices com grau de entrada maior na ordenação topológica**
- ⋆ **Para identificar tais vértices, é utilizada uma BFS modificada**
- ⋆ **Complexidade:** $O(E \log V)$

Pseudocódigo

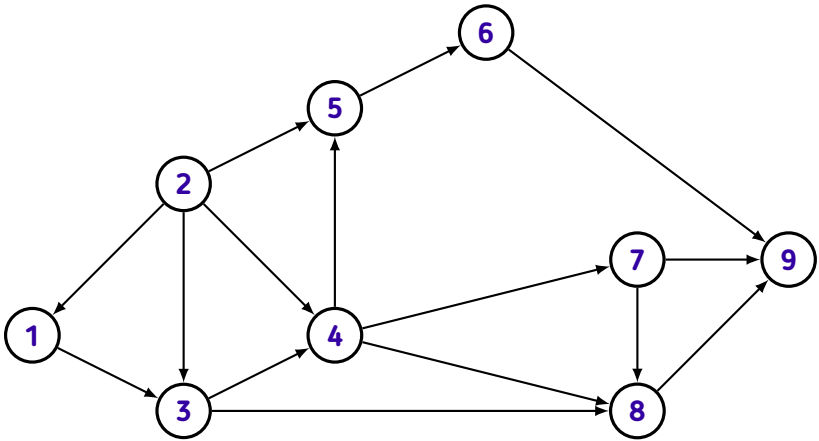
Pseudocódigo

Entrada: um grafo direcionado acíclico $G(V, E)$
Saída: uma ordenação topológica O de G

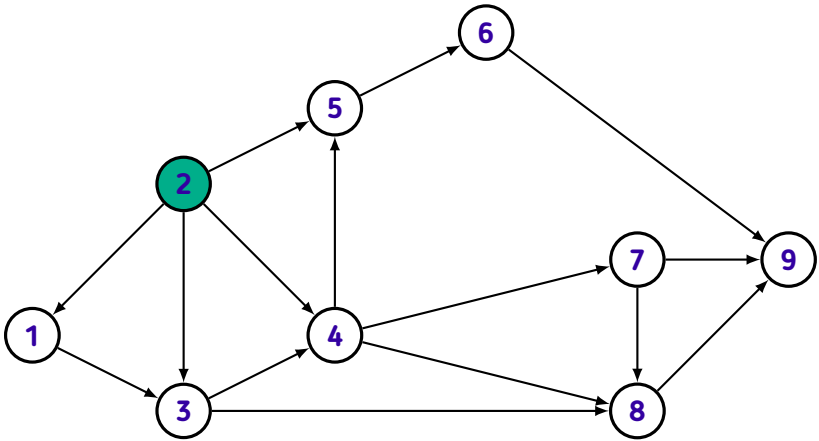
1. Insira, em uma fila Q, todos os vértices com grau de entrada igual a zero

2. Enquanto Q não estiver vazia:
   - (a) Extraia o primeiro elemento u da fila e o insira em O
   - (b) Exclua u e todas as suas arestas que parte de u do grafo G
   - (b) Insira em Q os vértices com grau de entrada igual a zero

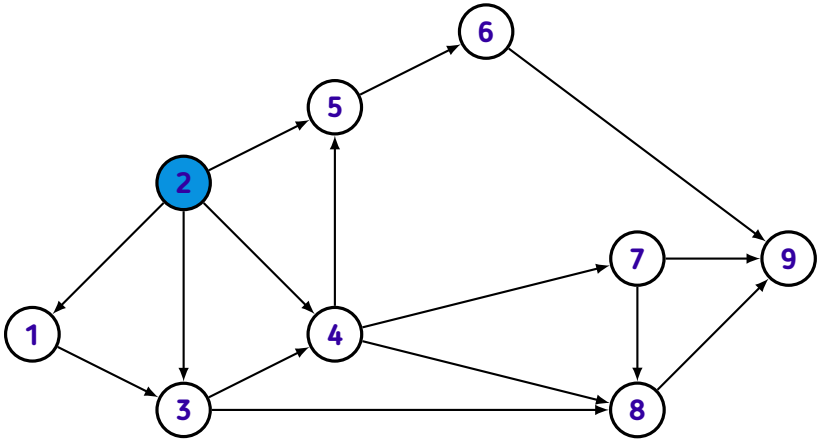
3. Retorne O



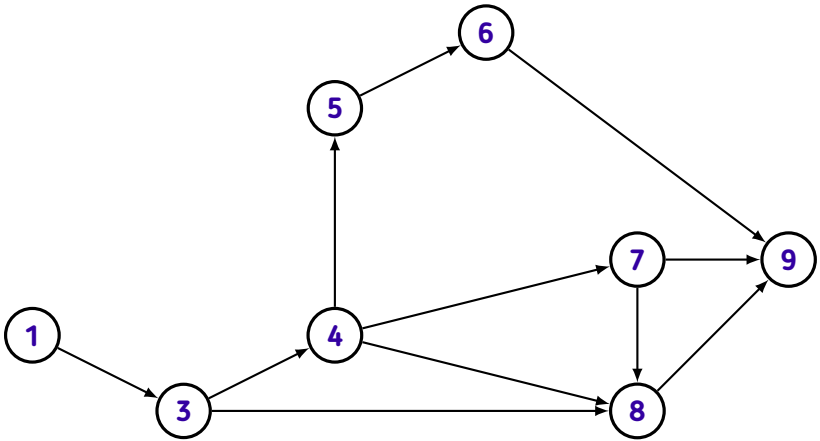
$Q = \{\ \},\ \ O = \{\ \}$



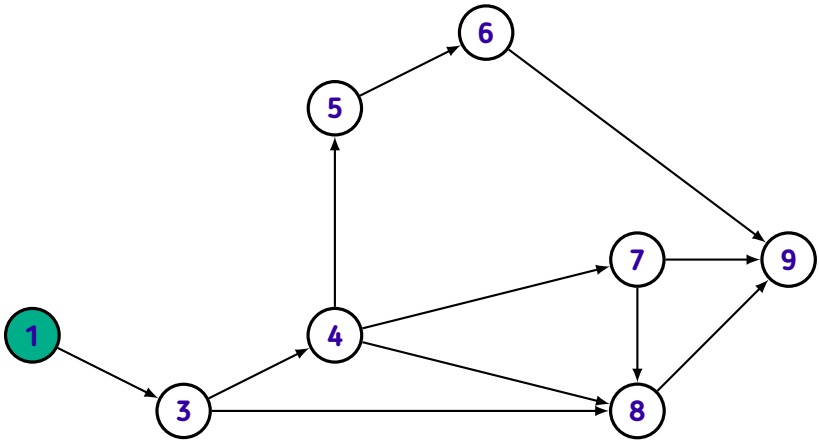
$Q = \{\ \mathbf{2}\ \},\quad O = \{\ \}$



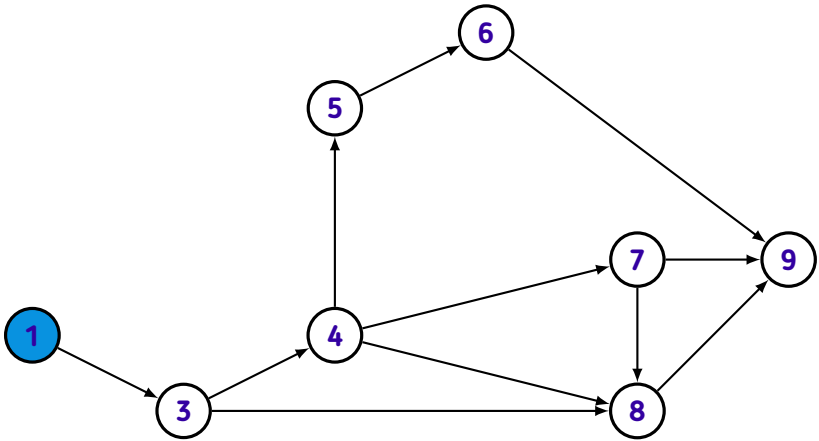
$Q = \{\ \},\ \ O = \{\ \mathbf{2}\ \}$



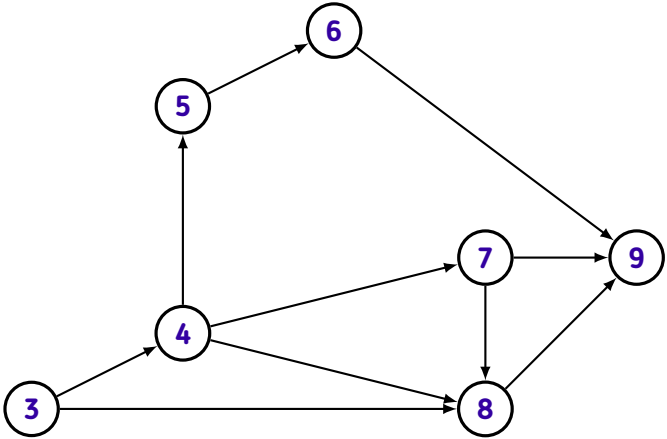
$Q = \{\ \},\ \ O = \{\ \mathbf{2}\ \}$



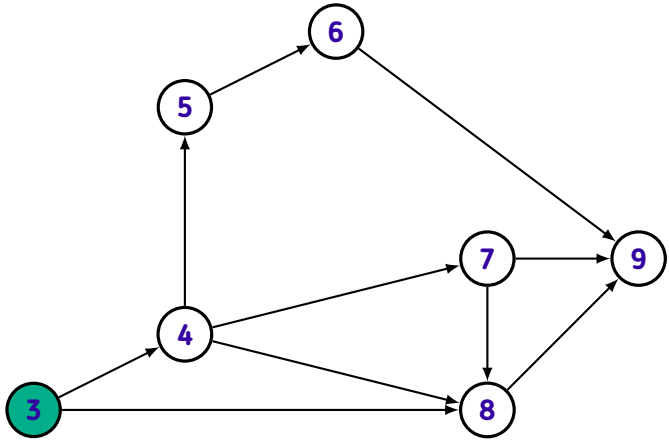
$Q = \{\ \mathbf{1}\ \},\quad O = \{\ \mathbf{2}\ \}$



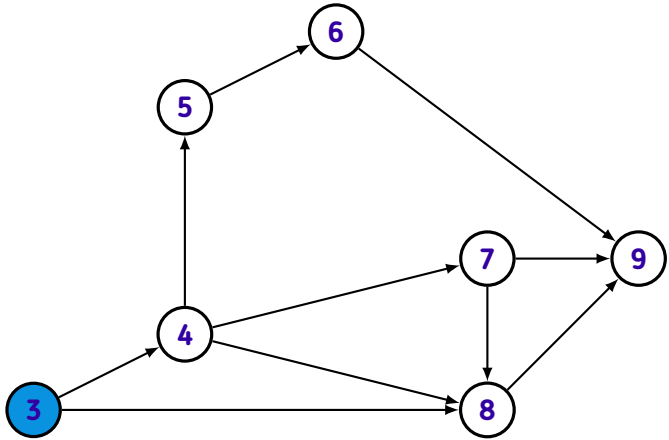
$Q = \{\ \}, \quad O = \{\ \mathbf{2, 1}\ \}$



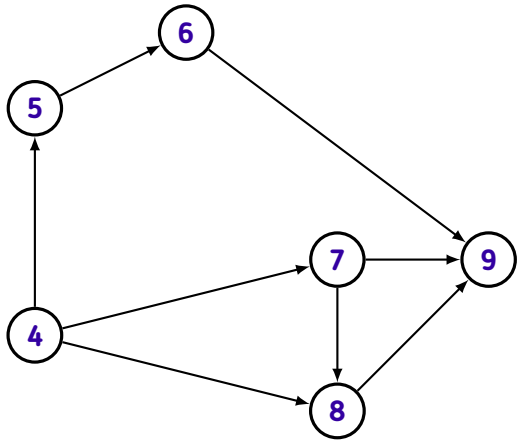
$Q = \{\ \}, \quad O = \{\ \mathbf{2, 1}\ \}$



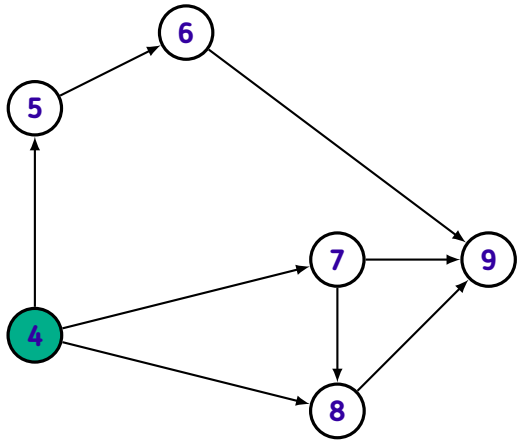
$Q = \{\ \mathbf{3}\ \},\quad O = \{\ \mathbf{2, 1}\ \}$



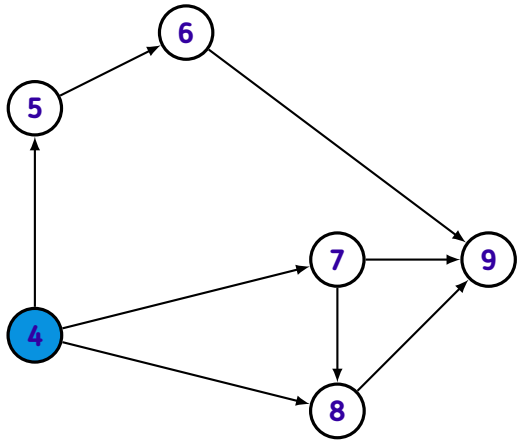
$Q = \{\ \}, \quad O = \{\ \mathbf{2, 1, 3}\ \}$



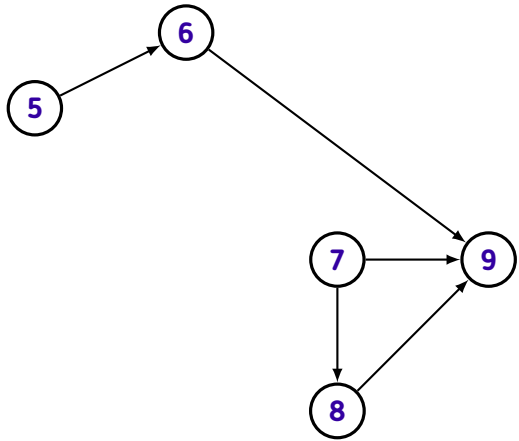
$Q = \{\ \}, \quad O = \{$ **2, 1, 3** $\}$



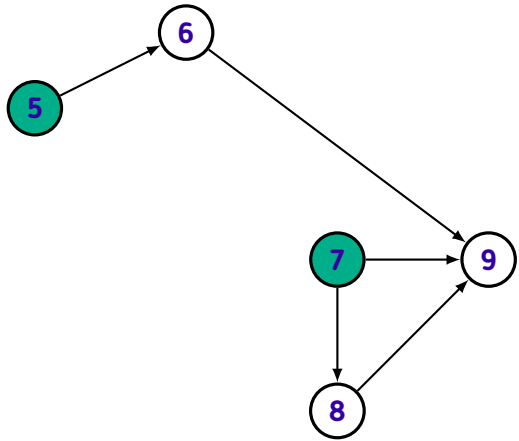
$Q = \{\ \mathbf{4}\ \},\quad O = \{\ \mathbf{2, 1, 3}\ \}$



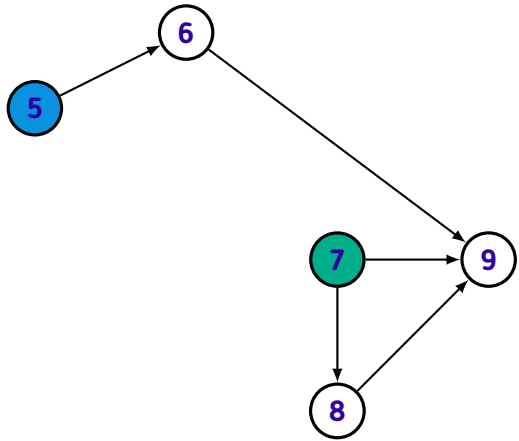
$Q = \{\ \},\quad O = \{$ **2, 1, 3, 4** $\}$



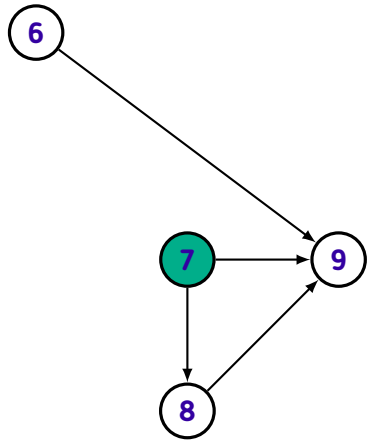
$Q = \{\ \},\quad O = \{$ **2, 1, 3, 4** $\}$



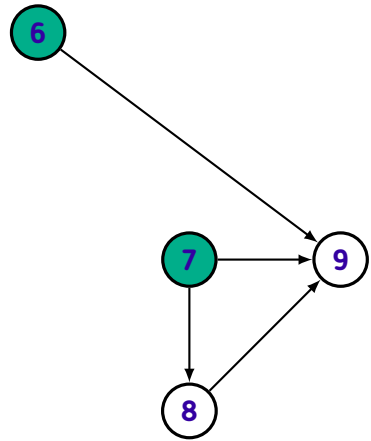
$Q = \{$ **5, 7** $\}, \quad O = \{$ **2, 1, 3, 4** $\}$



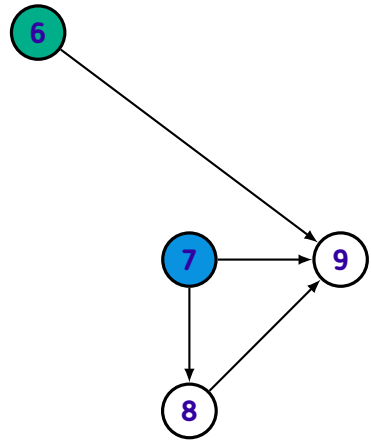
$Q = \{\ \mathbf{7}\ \},\quad O = \{\ \mathbf{2, 1, 3, 4, 5}\ \}$



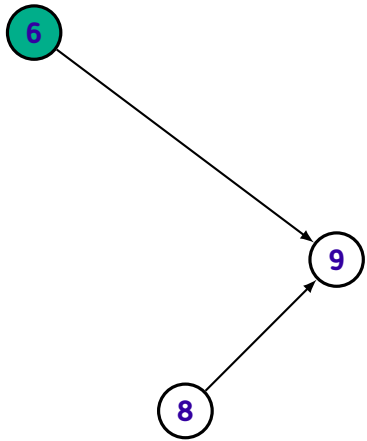
$Q = \{$ **7** $\}, \quad O = \{$ **2, 1, 3, 4, 5** $\}$



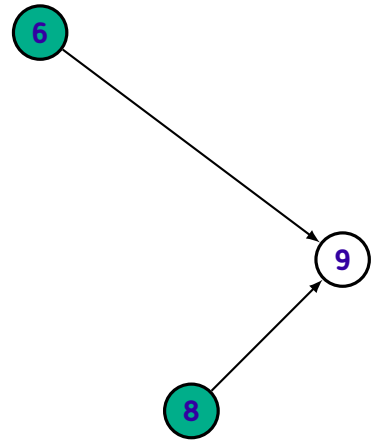
$Q = \{$ **7, 6** $\}, \quad O = \{$ **2, 1, 3, 4, 5** $\}$



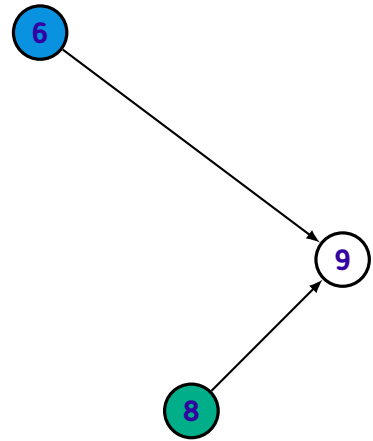
$Q = \{$ **6** $\}, \quad O = \{$ **2, 1, 3, 4, 5, 7** $\}$



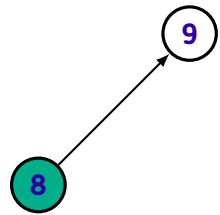
$Q = \{$ **6** $\}, \quad O = \{$ **2, 1, 3, 4, 5, 7** $\}$



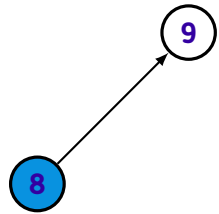
$Q = \{$ **6, 8** $\}, \quad O = \{$ **2, 1, 3, 4, 5, 7** $\}$



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



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



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

9

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

$Q = \{$ **9** $\}, \quad O = \{$ **2, 1, 3, 4, 5, 7, 6, 8** $\}$

9

$Q = \{\ \},\quad O = \{$ **2, 1, 3, 4, 5, 7, 6, 8, 9** $\}$

$Q = \{\ \},\quad O = \{\ \mathbf{2, 1, 3, 4, 5, 7, 6, 8, 9}\ \}$

```
vector<int> topological_sort(int N)
{
    vector<int> o;
    queue<int> q;

    for (int u = 1; u <= N; ++u)
        if (in[u].empty())
            q.push(u);

    while (not q.empty())
    {
        auto u = q.front();
        q.pop();

        o.emplace_back(u);
```

```
        for (auto v : out[u])
        {
            in[v].erase(u);

            if (in[v].empty())
                q.push(v);
        }
    }

    return (int) o.size() == N ? o : vector<int> { };
}
```

Problemas sugeridos

1. **Codeforces 510C – Fox and Names**
2. **OJ 11060 – Beverages**
3. **SPOJ TOPOSORT – Topological Sorting**
4. **Timus 1280 – Topological Sorting**

Referências

1. **ACM Digital Library**. *Topological sorting of large networks*, **A. B. Kahn, 1962.**
2. **HALIM**, **Felix**; **HALIM**, **Steve**. *Competitive Programming 3*, **2010.**
3. **LAAKSONEN**, **Antti**. *Competitive Programmer's Handbook*, **2018.**
4. **Wikipédia**. *Robert Tarjan*, **acesso em 08/09/2021.**
5. **Wikipédia**. *Topological sorting*, **acesso em 08/09/2021.**