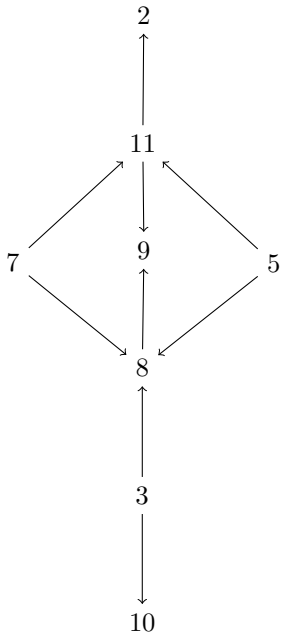
Programação Competitiva

Isaías Castro

Maio 2025

1 Ordenação Topológica

Dado um grafo dirigido acíclico (DAG), uma ordenação topológica de seus vértices é uma permutação dos vértices do grafo na qual cada vértice aparece antes dos vértices para os quais ele mande um arco.
Ex.:



7, 5, 3, 11, 8, 2, 9, 10 (cima para base, esquerda para direita
3, 5, 7, 8, 10, 11, 2, 9 (menor índice primeiro)
7, 5, 11, 3, 10, 8, 9, 2 (maior índice primeiro)
7, 3, 5, 11, 8, 2, 9, 10 (maior grau de saída)

1.1 Algoritmo de Khan

Entrada: um DAG $\vec{G}$
Saída: uma ordenação topológica dos vértices de $\vec{G}$

```
L = {}
S = {vértices de G que têm grau de entrada zero}
Enquanto S não for vazia:
    Remova de S obtendo v
    Insira v em L
    Para cada arco partindo de v:
        seja u o destino do arco
        Decremente o grau de entrada de u
        Se o grau de entrada de u for zero:
            Insira u em S
Devolva L
```

1.2 Implementação

1. Primeiro vamos criar uma função que retorne uma lista com o grau de entrada de cada vértice do grafo.

```cpp
vector<int> CalcDegree(vector<vector<int>>& G){
    int n = G.size();

    vector<int> Degree (n, 0);

    for(int i = 0; i < n; ++i){
        for(int x : G[i]){
            Degree[x]++;
        }
    }

    return Degree;
}
```

2. Agora vamos precisar criar uma lista que contenha apenas os vértices que tenham grau de entrada zero, nesse caso nossa função retornará uma fila.

```cpp
queue<int> _S(vector<int>& Degree){
    int n = Degree.size();
    queue<int> S;

    for(int i = 0; i < n; ++i){
        if(Degree[i] == 0) S.push(i);
    }

    return S;
}
```

3. Como já temos a nossa lista S, podemos já implementar o algoritmo em si.

```cpp
vector<int> Khan(vector<vector<int>>& G){
    int n = G.size();
    vector<int> L;
    vector<int> Degree = CalcDegree(G);
    queue<int> S = _S(Degree);

    while(!S.empty()){
        int v = S.front();
        S.pop();
        L.push_back(v);

        for(int x : G[v]){
            Degree[x]--;
            if(Degree[x] == 0) S.push(x);
        }
    }

    return L;
}
```