Topological Sort

LeetCode 207 - Course Schedule *[medium]*

There are a total of **n** courses you have to take, labeled from **0** to **n - 1**.

Some courses may have prerequisites, for example to take course **0** you have to first take course **1**, which is expressed as a pair: **[0, 1]**

Given the total number of courses and a list of prerequisite pairs, is it possible for you to finish all courses?

Example 1:

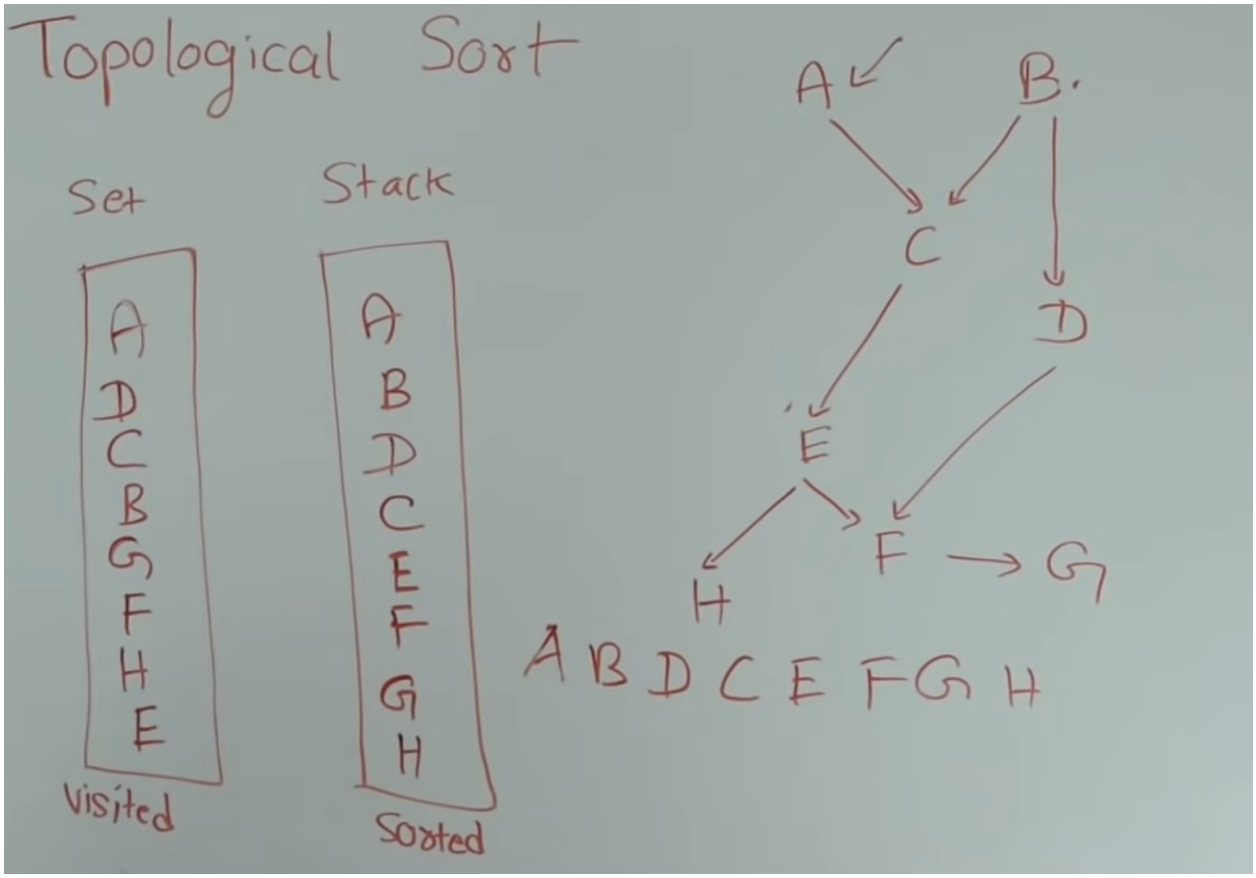
```
Input: 2, [[1, 0]]
Output: true
Explanation: There are a total of 2 courses to take.
             To take course 1 you should have finished course 0. So it is possib
```

Example 2:

```
Input: 2, [[1, 0], [0, 1]]
Output: false
Explanation: There are a total of 2 courses to take.
             To take course 1 you should have finished course 0, and to take cou
             you should also have finished course 1. So it is impossible.
```

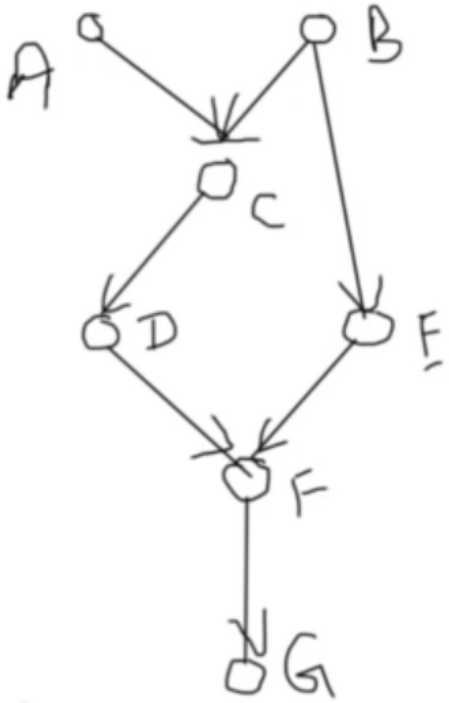
Note:

- The input **prerequisites** is a graph represented by a list of **edges**, not adjacency matrices. Read more about how a graph is represented.
- You may assume that there are no duplicate **edges** in the input


Topological Sort
Set
Stack
A
D
C
B
G
F
H
E
Visited
A
B
D
C
E
F
G
H
Sorted
A
B
C
D
E
F
G
H
A B D C E F G H

stack

visited



A B E C D F G

```java
public class TopologicalSort<T> {
    public Deque<Vertex<T>> topSort(Graph<T> graph) {
        Deque<Vertex<T>> stack = new ArrayDeque<>();
        Set<Vertex<T>> visited = new HashSet<>();
        for (Vertex<T> vertex : graph.getAllVertex()) {
            if (visited.contains(vertex)) {
                continue;
            }
            topSortUtil(vertex,stack,visited);
        }
        return stack;
    }

    private void topSortUtil(Vertex<T> vertex, Deque<Vertex<T>> stack,
            Set<Vertex<T>> visited) {
        visited.add(vertex);
        for(Vertex<T> childVertex : vertex.getAdjacentVertexes()){
            if(visited.contains(childVertex)){
                continue;
            }
            topSortUtil(childVertex,stack,visited);
        }
        stack.offerFirst(vertex);
    }

    public static void main(String args[]){
        Graph<Integer> graph = new Graph<>(true);
        graph.addEdge(1, 3);
        graph.addEdge(1, 2);
        graph.addEdge(3, 4);
        graph.addEdge(5, 6);
        graph.addEdge(6, 3);
        graph.addEdge(3, 8);
        graph.addEdge(8, 11);

        TopologicalSort<Integer> sort = new TopologicalSort<Integer>();
        Deque<Vertex<Integer>> result = sort.topSort(graph);
        while(!result.isEmpty()){
            System.out.println(result.poll());
        }
    }
}
```

LeetCode 210 - Course Schedule II [*medium*]

LeetCode 269 - Alien Dictionary [*hard*] [*premium*]

LeetCode 310 - Minimum Height Trees [*medium*]