DELHI TECHNOLOGICAL UNIVERSITY

GRAPH THEORY (MC-405)

Lab Project

COURSE SCHEDULING USING TOPOLOGICAL SORTING

SUBMITTED BY:

Aiman Siddiqua - 2K18/MC/008

Apoorva - 2K18/MC/019

SUBMITTED TO:

Dr. Sangita Kansal

Radhika Kavra

INDEX

PROBLEM STATEMENT **3**

DISCUSSION **4**

PROBLEM SOLUTION 4

TOPOLOGICAL SORTING 4

ALGORITHM 6

CODE **7**

OUTPUT **9**

PROBLEM STATEMENT

In a university, a student is allowed to choose whatever course they desire to take from the given catalogue. However, certain courses require you to have completed some prerequisite courses before you take them.

You are given a number of courses in a catalogue and a list of prerequisites for each course. Write a program that gives you the correct order in which the student must take the courses.

EXAMPLE

Consider a catalogue containing 5 courses with the following prerequisites:

- Course 1 must be taken before course 2 and course 3.
- Course 2 must be taken before course 3 and course 4.
- Course 3 must be taken before course 4 and course 5.

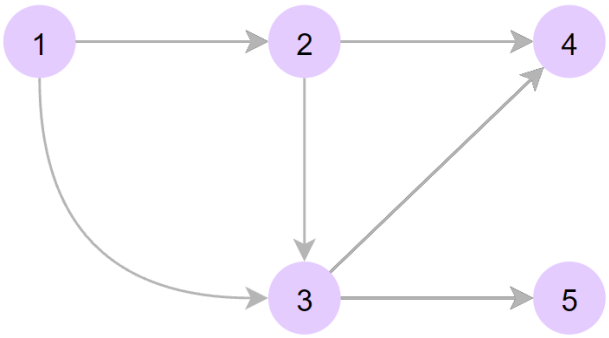


Fig. Course Prerequisite Graph

Course Schedule - 1 -> 2 -> 3 -> 4 -> 5

DISCUSSION

PROBLEM SOLUTION

We need to schedule the course in such a way that the prerequisites of each course are scheduled before that particular course.

Let us consider the courses as vertices and the prerequisites as directed edges of a graph. There exists an edge from A to B if Course A is a prerequisite of course B.

If there are no cycles in the directed graph then the ordering of vertices required in the problem can be achieved by performing a topological sorting on the graph.

TOPOLOGICAL SORTING

Topological Sort is a linear ordering of the vertices in such a way that if there is an edge in the directed acyclic graph going from vertex 'u' to vertex 'v', then 'u' comes before 'v' in the ordering.

It is important to note that topological sorting is not possible if the directed graph contains a cycle.
For example, for the given graph,

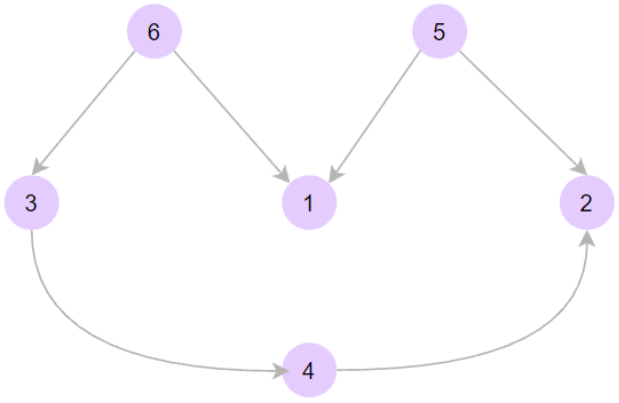


Fig 2. Example Graph for Topological Sorting

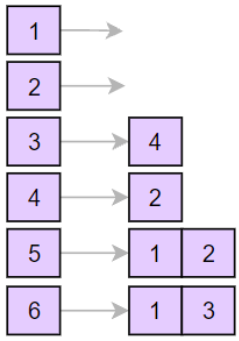


Fig 3. Adjacency for graph in Fig 2.

The topological sorting order can be 6 5 1 3 4 2 or 5 6 3 4 2 1.

Let us illustrate how this is done. We use the following two data structures.

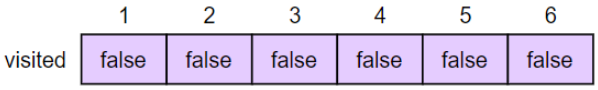


Fig 3. Boolean Array that keeps track of visited nodes.

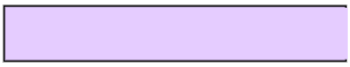
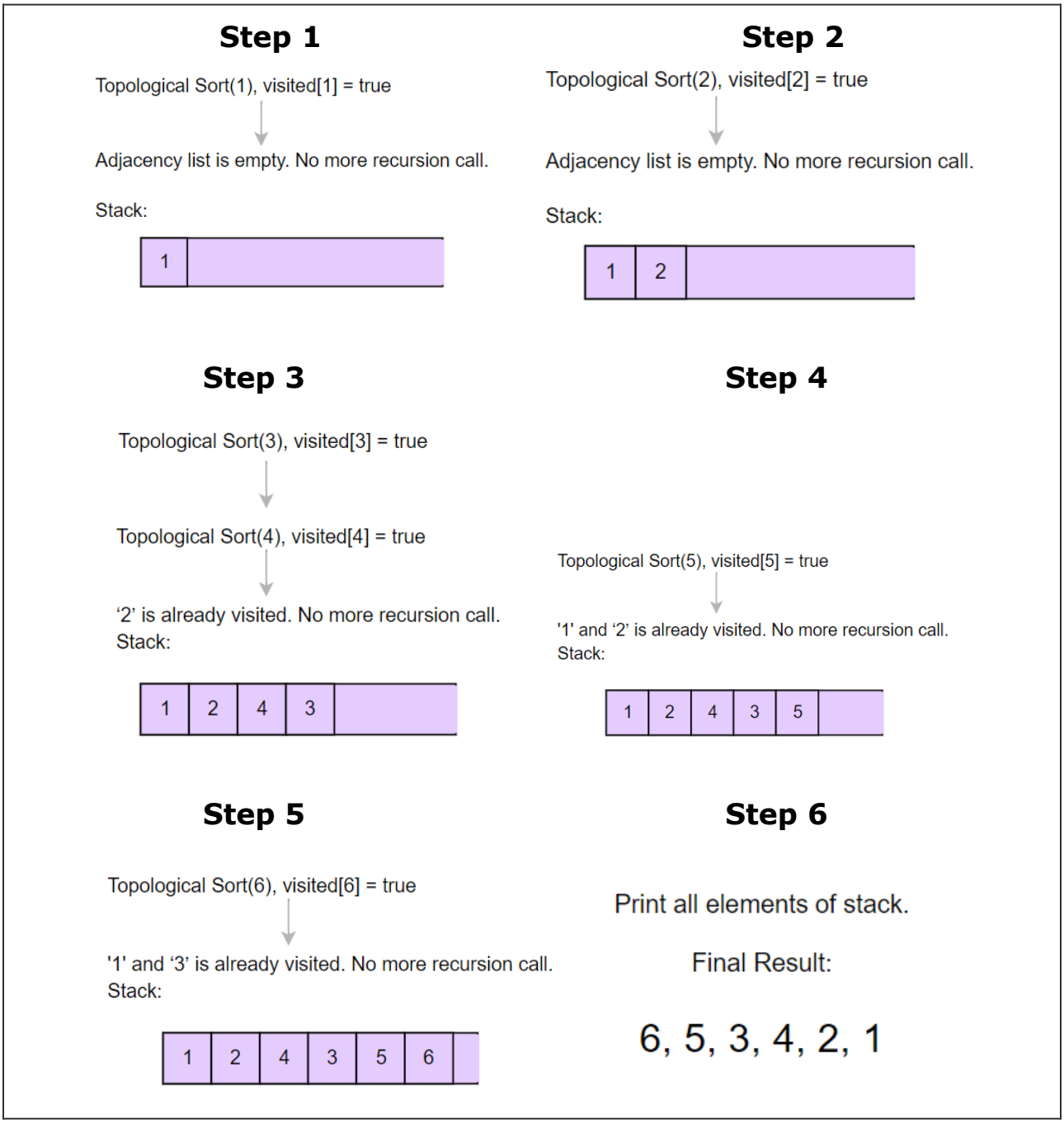


Fig 4. Empty Stack



Few important applications of the topological sort are:

- Scheduling jobs from the given dependencies among jobs
- Instruction Scheduling
- Determining the order of compilation tasks to perform in makefiles
- Data Serialization

ALGORITHM

1. Create a directed graph of size equal to the total number of courses in the catalogue.
2. There exists an edge from A to B if Course A is a prerequisite of course B.
3. Perform topological sorting.
4. If there exists a cycle in the graph, topological sorting returns false indicating that course scheduling is not possible with the given prerequisites.
5. Else, Print the topological sorting order.

CODE

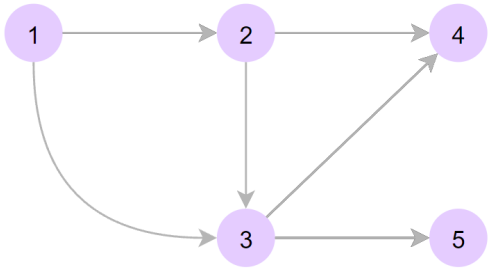
```cpp
#include<bits/stdc++.h>
using namespace std;

stack<int> courses;

bool explore(int u, list<int> adj[], bool visited[], bool rec[]){
    visited[u]=true;
    rec[u] = true;

    bool flag = false;

    for(auto i=adj[u].begin(); i!=adj[u].end(); i++){
        if(!visited[*i] && explore(*i, adj, visited, rec))
            flag = true;

        else if(rec[*i])
            flag = true;
    }

    rec[u] = false;
    courses.push(u);

    return flag;
}

bool topologicalSort(list<int> adj[], int n){
    bool visited[n];
    memset(visited, false, sizeof(visited));

    bool rec[n];
    memset(rec,false,sizeof(rec));

    stack<int> st;

    for(int i=0; i<n; i++){
        if(!visited[i])
            if(explore(i, adj, visited, rec))
                return false;
    }
    return true;
}
```

```
int main(){
    int n,x;

    cout<<"Enter the number of courses: ";
    cin>>n;

    cout<<"\nEnter the number of prerequisites: ";
    cin>>x;

    list<int> adj[n];

    cout<<"\nEnter prerequisites for each course"<<endl;
    for(int i=0;i<x;i++){
        int u,v;
        cin>>u>>v;

        adj[u-1].push_back(v-1);
    }

    if(topologicalSort(adj, n)){
        cout<<"\nCourse Schedule: ";

        while(!courses.empty()){
            cout<<courses.top()+1<<" ";
            courses.pop();
        }
        cout<<endl;
    }
    else
        cout<<"Course Scheduling is not possible"<<endl;

    return 0;
}
```

OUTPUT

Example 1:

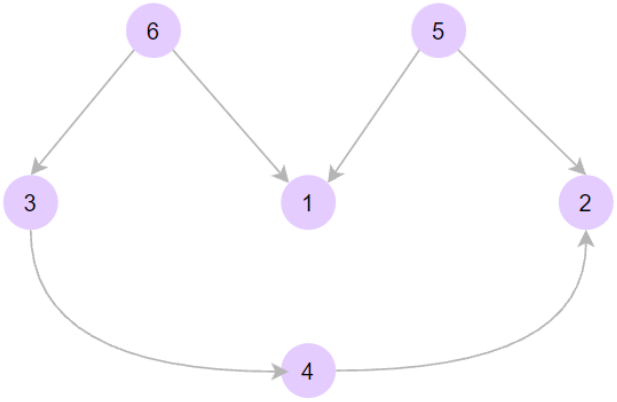
Graph:




```
Enter the number of courses: 5

Enter the number of prerequisites: 6

Enter prerequisites for each course
1 2
1 3
2 3
2 4
3 4
3 5

Course Schedule: 1 2 3 5 4
```

Example 2:

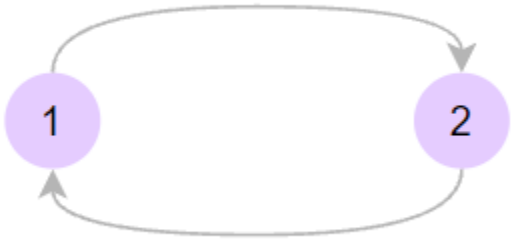
Graph:




```
Enter the number of courses: 6

Enter the number of prerequisites: 6

Enter prerequisites for each course
6 3
6 1
5 1
5 2
3 4
4 2

Course Schedule: 6 5 3 4 2 1
```

Example 3:

Graph:




```
Enter the number of courses: 2

Enter the number of prerequisites: 2

Enter prerequisites for each course
1 2
2 1

Course Scheduling is not possible
```