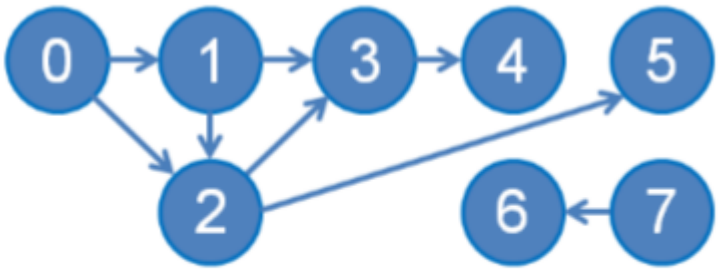
nhatnguyendrgs

Topo Sort

Trong khoa học máy tính, thứ tự tô pô của một đồ thị có hướng là một thứ tự sắp xếp của các đỉnh sao cho với mọi cung từ u đến v trong đồ thị, u luôn nằm trước v. Thuật toán để tìm thứ tự tô pô gọi là **thuật toán sắp xếp tô pô**. Thứ tự tô pô tồn tại khi và chỉ khi **đồ thị không có chu trình (viết tắt là DAG - tiếng Anh directed acyclic graph)**. Đồ thị có hướng không có chu trình luôn có ít nhất một thứ tự tô pô, và có thuật toán để tìm thứ tự tô pô trong thời gian tuyến tính... (Wikipedia).



Input:

```
8 8
0 1 0
0 2 0
1 2 0
1 3 0
2 3 0
2 5 0
3 4 0
```

Output:

```
Topological Sort (the input graph must be DAG)
 7 6 0 1 2 5 3 4
```

Code (tham khảo Competitive Programming):

```
// nhatnguyendrgs (c) 2015 - toposort.cpp

#include "iostream"
#include "stdio.h"
#include "stdlib.h"
#include "string"
#include "string.h"
#include "algorithm"
#include "math.h"
#include "vector"
#include "map"
#include "queue"
#include "stack"
#include "deque"
#include "set"
using namespace std;

typedef pair<int, int> ii;
typedef vector<int> vi;
typedef vector<ii> vii;
const int inf = 1e9;

#define DFS_WHITE -1 //    UNVISITED
#define DFS_BLACK 1  //    EXPLORED
#define DFS_GRAY 2   //    VISTED BUT NOT EXPLO
RED

vector<vii> AdjList;
int V, E, u, v, w;

void graphDirected(){
    scanf("%d %d", &V, &E);
    AdjList.assign(V + 4, vii());
```

```
    for (int i = 0; i < E; i++) {
        scanf("%d %d %d", &u, &v, &w);
        AdjList[u].push_back(ii(v, w));
    }
}

vi dfs_num;
vi topoSort;                // global vector to
 store the toposort in reverse order
bool cycle = false;             // check cycle
 in graph
void topo(int u) {              // change funct
ion name to differentiate with original dfs
    dfs_num[u] = DFS_GRAY;
    for (int j = 0; j < AdjList[u].size(); j++)
{
        ii v = AdjList[u][j];
        if (dfs_num[v.first] == DFS_WHITE)
            topo(v.first);
        else if (dfs_num[v.first] == DFS_GRAY)
            cycle = true;
    }
    topoSort.push_back(u);
    dfs_num[u] = DFS_BLACK;
}


int main(){
    printf("Topological Sort (the input graph m
ust be DAG)\n");
    graphDirected();
    topoSort.clear();
    dfs_num.assign(V+4, DFS_WHITE);
    for (int i = 0; i < V; i++)
        // this part is the same as finding CCs
    if (dfs_num[i] == DFS_WHITE)
        topo(i);
    if (cycle == true)
        printf(" Exist cycle.\n");
    else{
        reverse(topoSort.begin(), topoSort.end
());     // reverse topoSort
        for (int i = 0; i < (int)topoSort.size
(); i++) // or you can simply read
            printf(" %d", topoSort[i]);
        // the content of `topoSort' backwards
    }
    printf("\n");
```

```
    return 0;
}
```