Topological Sort

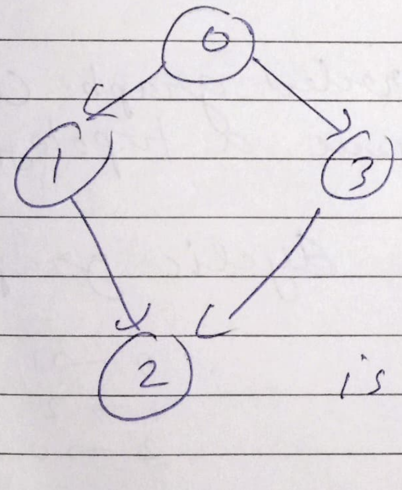
Applied on Directed Acyclic Graph....

Direction   no cycle

What is it?

It is a linear ordering of Vertices such that if there is a edge from u to v then u must appears before v.

eg.



Adj List

0 → 1, 3
1 → 2
2 → [ ]
3 → 2.

is (0 1 3 2) — ①
a topological sort?

Lets check

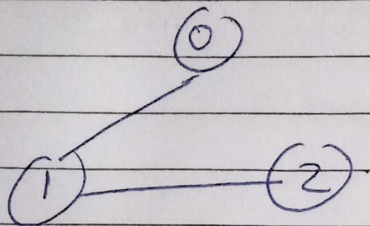
there edge b/w 0 → 1, 0 is befor 1 in ①
" " " 0 → 3, 0 ——— 3 in ①
" " " 1 → 2 1 " " 2 in ①
" " " 3 → 2 3 is befor 2 in ①

∴ 0, 1, 3, 2 is a valid topological Sort

Q- Why topological Sort is done only for 'directed' 'acyclic' graphs ?

C-1

A- Let take a undirected graph.



0 → 1
1 → 0, 2
2 → 1.

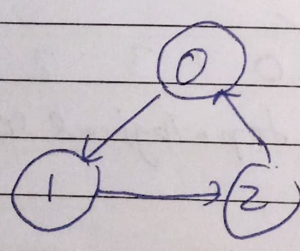
0 1 2. is a TS ?

0 has neigh 1 0 is occuring before 1
1 has neig 0 1 is occurin before 0 ?
No ✗

∴ directed graph can not have a topological Sort

C-2

Lets take cyclic graph (directed)

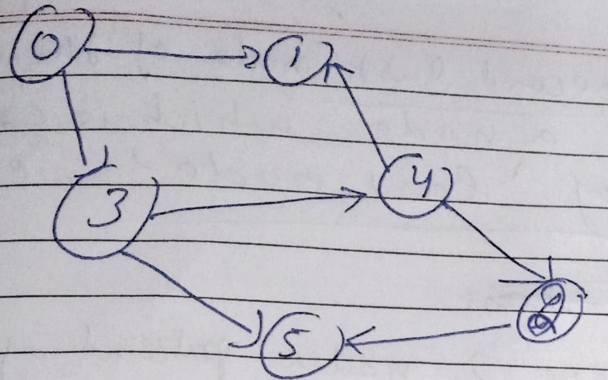


0 → 1
1 → 2
2 → 0.

0 1 2. is TS ?

0 to 1 there an edge. 0 occurs before 1 ✓
1 to 2 " " " 1 " 2 ✓
2 to 0 " " " 2 " " 0 ? No

∴ Cyclic graph can not have T.S.



0 → [1, 3]
1 → [ ]
2 → [5]
3 → [4, 5]
4 → [1, 2]
5 → [ ]

~~This~~ a an acyclic directed Graph (DAG)

One of Its topological sort order will be.

~~0 1 2 3 4 1 2 5 ?~~

0 3 4 1 2 5

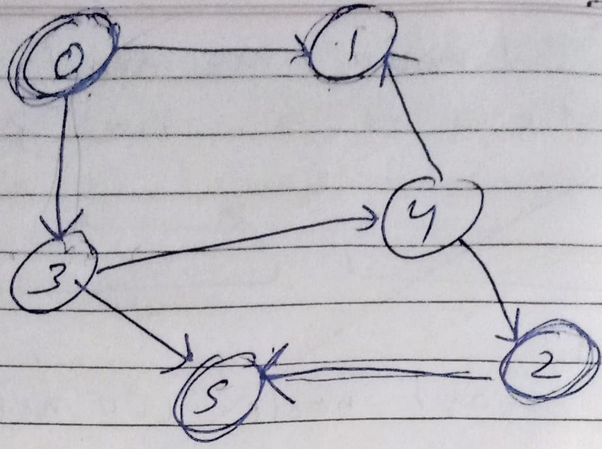
this is a valid T.S
but why ?

5 is at last in T.S order. means there is no node to which 5 needs to be ahead. i.e no node to which we can go from 5, 5 is dead end. i.e where we cant move further,

∴ the last node in Topological order will be something where rec calls will end. & return. so store it at end.

last node.

dry run.



Adj List

0 → 1, 3
1 →
2 → 5
3 → 4, 5
4 → 1, 2
5 →

Visited

0	1	2	3	4	5
T	T	T	T	T	T

0
3
4
2
5
1

stk.

5 - no parent

1 - no parent

DFS (0)

→ DFS (1)

→ empty

→ DFS (3).

→ DFS (4)

→ visited. DFS (1).

→ DFS (2).

→ DFS (5).

traverse the stk now.

Valid T.S ✓

0 3 4 2 5 1.

means whenever we reached to a node from where we can't go ahead i.e a node that has no childs!.

Topological order. 1

last.

So then we make dfs calls in path

0 → 3 → 4 → 2 → 5 ← 5 has No point

Topological order. 5 1

now from 2 we can't go anywhere else means we found another node.

2 5 1

now 4 doesn't have any other node.

4 2 5 1

~~now 4 then 3~~

3 4 2 5 1

now 0

0, 3, 4, 2, 5,

So why stack is used?

because in topological order we put the node with no parent at the end.

so if we need a Last in First o structure so that when we fetch the elements from stack

the ~~most~~ no parent nodes come at last — — —

Topological Sort Code

```
void TopoDFS( &AdjList, &visited, &Stack, src).
{
    visited[src] = true;
    for( int neigh : AdjList[src])
    {
        if(!visited[neigh])
        {
            TopoDFS(Adj, visit, stk, neigh)
            ~~stk.push(neigh)~~
        }.
    }.
    stk.push(src);
        When a nodes call returns
        Store that node in Stack.

    T : O(V+E)
    S : 3(O(V+E)).
}.
main()
{
    // AdjList create karo

    // do call for dfs (for all comp of graph
    for(int i=0; i<verticesSize; i++)
    {
        if(!visited[i])
            TopoDFS(AdjList, visited, stack, i);
                                          *
    // fetch the stk & return ele
}                                  stk LIFO store h.
```

Thats
it.