Class 93

24 May 2023 11:09

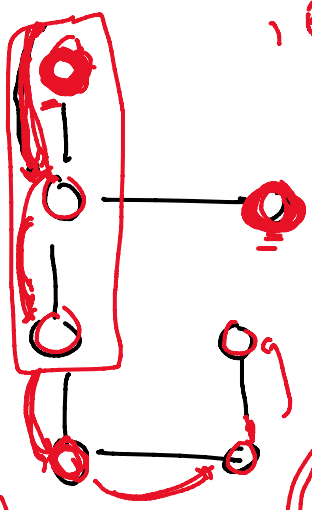
Class 93

Graphs

DFS / BFS → O(V+E)

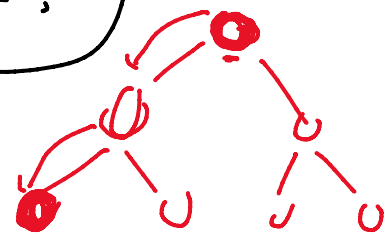
↳ visited array because of cycle

Back edge



Back edge → Edge connect Node to its ancestor

cycle ?



Ancestor

Node in the path from root to cur Node

cur Node

Backedge

Cycle Node

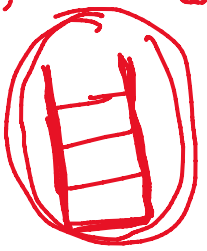
Ancestor Node

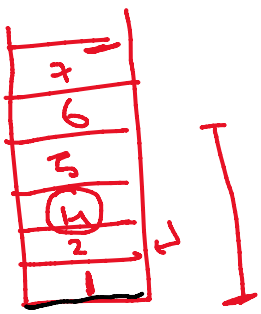
Cycle → Backedge

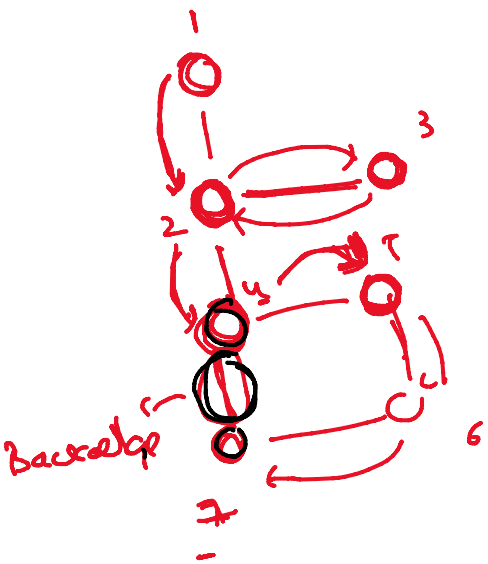
Ancestor Node

Backedge

Call stack Node



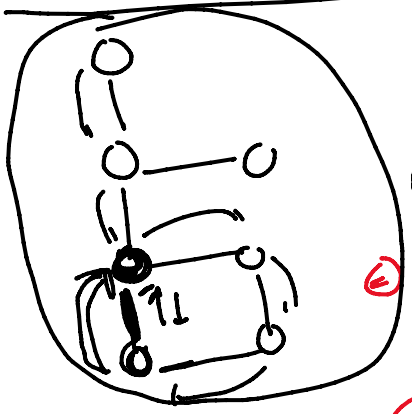




Backedge

undirected

Find cycle in a graph using DFS

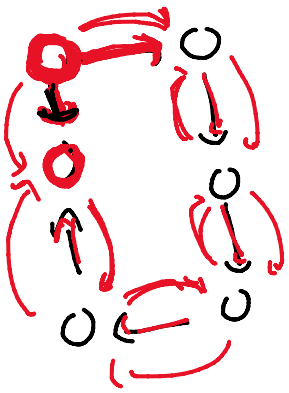


Visited

Ancestor Node

Visited

Ancestor Node

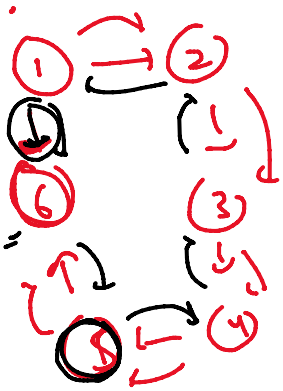


visited ? ✓

Ancestor Node ?

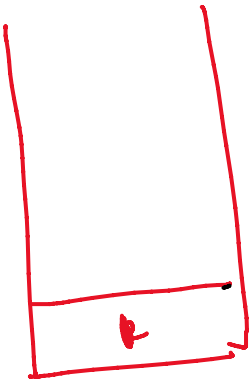
Stack ?

Cycle ? X



Cycle

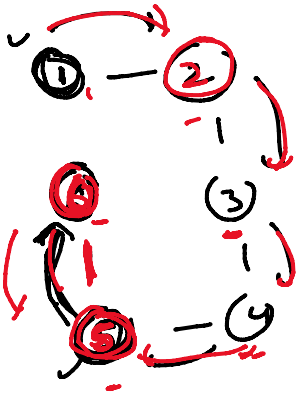
Backedge

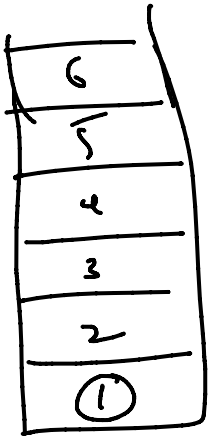


X

visited ✓

ancestor Node ? ✓





Backedge

1 visited ? ✓

ancestor ? ✓

vis[node] ✓

vis[i] ?

Cycle in undirected graph

vis[node] ✓

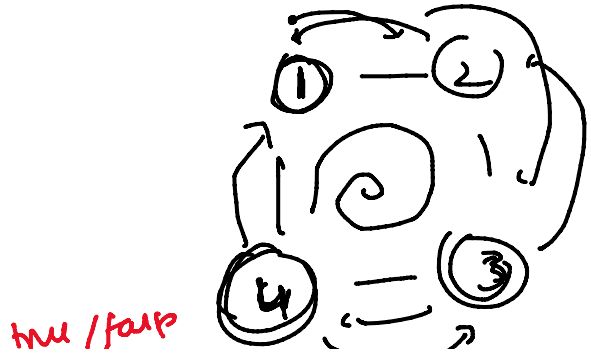
node != par

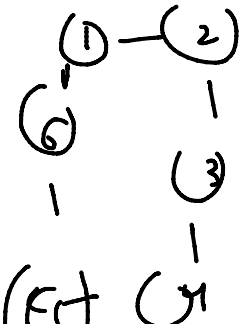
```
vector<int>parent(n);
bool dfs ( int i, vector<vector<int>> adj, vector<int>&vis,
                                                  int par)
{
    vis[i] = 1;
    parent[i] = par;
    for( auto j : adj[i] )
    {
        if ( !vis[j] )
            return dfs (j, adj, vis, i);
        else if ( j != par )
            return true;
    }
    return false;
}
```

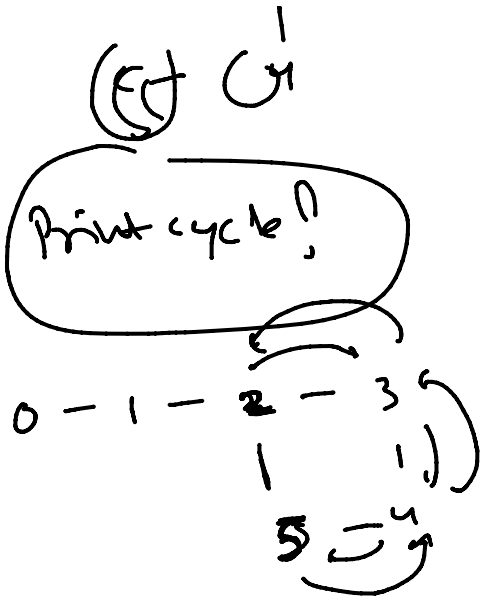
// Node Parent

BackedgeX

tree = graph w/o backedge



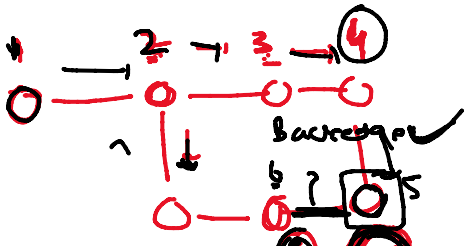




Cycle Detection in Undirected using BFS

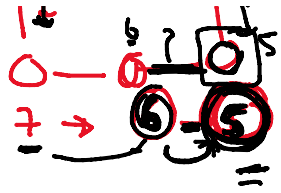
$O(V+E)$

```
vector<int> vis(n);
queue<int> q;
vector<int> parent(n);
q.push(0);
parent[0] = -1;
vis[0] = 1;
while(!q.empty())
{
  auto tp = q.front();
  q.pop();
  for(auto i: adj[tp])
  {
    if(!vis[i]) X
    {
      q.push(i);
      vis[i] = 1;
```



```
q.push(i);
vis[i] = 1;
par[i] = tp;
}
else if (i != par[tp])
{
    // found cycle
    //print cycle
```

1 → -1
2 → 1
3 → 2
7 → 2
4 → 3
6 → 7
5 → 4



Undirected graph → using BFS, DFS

[Directed graph, Detect cycle using DFS

```
vector<int> vis(n);
vector<int> callStack(n);
```

```
bool cycle(vector<vector<int>> adj, int node)
{
    vis[node] = 1
    callStack[node] = 1;

    for (auto i : adj[node])
```

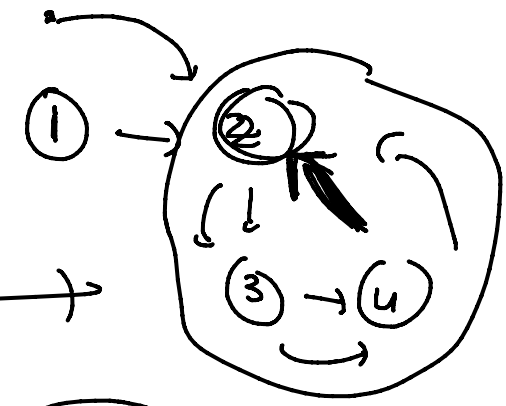
```
if ( !vis[i] )
{
    bool check = cycle ( adj, i);
    if (check)
    return true;
}
else if ( callstack[i] == 1 )
{ // cycle
    return true;
}
```

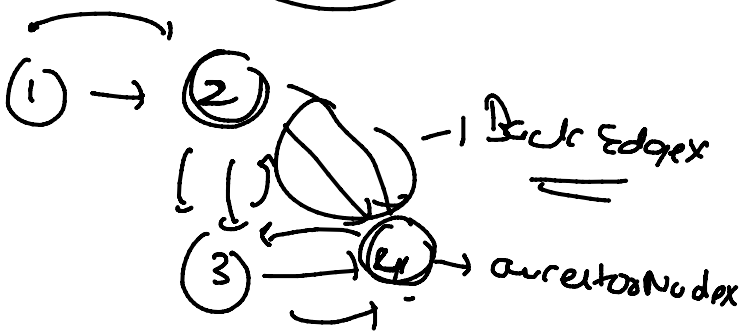
BFS

Topological sort

callstack[node] = 0;

1 →	!, !
2 →	1, 1
3 →	1, 1
4 →	1, 1





1 →	1, 1
2 →	1, 1
3 →	1, 0
4 →	1, 0 ✗