Cycle Detection in Graphs

Two Main Cases:

- Undirected Graph
- Directed Graph

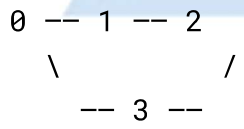
Cycle Detection in Undirected Graphs

Theoretical Concepts

What is a Cycle in an Undirected Graph?

A cycle in an undirected graph is a path that starts and ends at the same vertex, **without repeating any edge.**

Example Graph:



This graph has a cycle: `0 → 1 → 2 → 3 → 0`

Method 1: DFS (Depth-First Search)

concept:

- Visit all vertices using DFS.
- Keep track of the parent node.

- If a visited node is encountered that is **not the parent**, a cycle exists.

C++ Implementation – DFS Approach

```
#include <iostream>
#include <vector>
using namespace std;

bool dfsCycle(int node, int parent, vector<bool>& visited, vector<int>
adj[]) {
    visited[node] = true;

    for (int neighbor : adj[node]) {
        if (!visited[neighbor]) {
            if (dfsCycle(neighbor, node, visited, adj))
                return true;
        }
        else if (neighbor != parent) {
            return true; // Back edge detected
        }
    }

    return false;
}

bool isCyclicUndirected(int V, vector<int> adj[]) {
    vector<bool> visited(V, false);

    for (int i = 0; i < V; ++i) {
        if (!visited[i]) {
            if (dfsCycle(i, -1, visited, adj))
                return true;
```

```
        }
    }
    return false;
}
```

Method 2: Union-Find (Disjoint Set)

concept:

- For each edge, check if both vertices belong to the same set.
- If yes → cycle exists.
- Otherwise, union the sets.

C++ Implementation – Union-Find

```
int findParent(int x, vector<int>& parent) {
    if (parent[x] == -1) return x;
    return parent[x] = findParent(parent[x], parent);
}

bool unionFindCycle(int V, vector<pair<int, int>>& edges) {
    vector<int> parent(V, -1);

    for (auto edge : edges) {
        int u = findParent(edge.first, parent);
        int v = findParent(edge.second, parent);

        if (u == v) return true;  // Cycle found

        parent[u] = v;  // Union
```

```
    }
    return false;
}
```

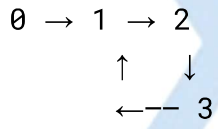
Cycle Detection in Directed Graphs

Theoretical Concepts

What is a Cycle in Directed Graph?

A cycle exists if there's a **path** from a vertex back to itself following edge directions.

Example:



Cycle: 1 → 2 → 3 → 1

Method: DFS + Recursion Stack

concept:

- Use DFS and keep track of recursion stack.
- If a node is visited and in recursion stack → **cycle exists.**

C++ Implementation – DFS + Rec Stack

```
bool dfsDirected(int node, vector<int> adj[], vector<bool>& visited,
vector<bool>& recStack) {
    visited[node] = true;
    recStack[node] = true;

    for (int neighbor : adj[node]) {
        if (!visited[neighbor]) {
            if (dfsDirected(neighbor, adj, visited, recStack))
                return true;
        } else if (recStack[neighbor]) {
            return true; // Back edge → cycle
        }
    }

    recStack[node] = false;
    return false;
}

bool isCyclicDirected(int V, vector<int> adj[]) {
    vector<bool> visited(V, false);
    vector<bool> recStack(V, false);

    for (int i = 0; i < V; ++i) {
        if (!visited[i]) {
            if (dfsDirected(i, adj, visited, recStack))
                return true;
        }
    }
    return false;
}
```