**GRAPH**

**Q1)adjacency list representation**

void addEdge(vector<int> adj[], int u, int v)

{

adj[u].push\_back(v);

adj[v].push\_back(u);

}

void printGraph(vector<int> adj[], int V)

{

for (int i = 0; i < V; i++)

{

for (int x : adj[i])

cout << x <<" ";

cout<<"\n";

}

}

int main()

{

int V = 4;

vector<int> adj[V];

addEdge(adj, 0, 1);

addEdge(adj, 0, 2);

addEdge(adj, 1, 2);

addEdge(adj, 1, 3);

printGraph(adj, V);

return 0;

}  
**note:you can use the unordered\_map too**

**Q2) BFS DFS   
try it yourself using unordered\_map**

**Q3) Detect cycle in an undirected graph**

**DFS APPROACH:**

bool isCycleUtil(vector<int> g[], int start, vector<bool>& visited, int parent) {

visited[start] = true;

vector<int>::iterator it;

for(it = g[start].begin(); it != g[start].end(); it++) {

if(\*it == parent)

continue;

if(visited[\*it] == true)

return true;

if(isCycleUtil(g, \*it, visited, start))

return true;

}

return false;

}

bool isCyclic(vector<int> g[], int V) {

vector<bool> visited(V, false);

for(int i = 0; i<V; i++) {

if(visited[i] == false && isCycleUtil(g, i, visited, -1))

return true;

}

return false;

}

**BFS APPROACH**

class Solution {

public:

bool isCycleBFS(vector<int> g[], int V, int start, vector<bool>& visited) {

queue<pair<int, int>> que;

que.push({start, -1});

visited[start] = true;

while(!que.empty()) {

int curr = que.front().first;

int parent = que.front().second;

que.pop();

for(auto x : g[curr]) {

if(visited[x] == false) {

que.push({x, curr});

visited[x] = true;

}

else if(x != parent)

return true;

}

}

return false;

}

// Function to detect cycle in an undirected graph.

bool isCycle(int V, vector<int> adj[]) {

vector<bool> visited(V, false);

for(int i = 0; i<V; i++) {

if(!visited[i] && isCycleBFS(adj, V, i, visited)) {

return true;

}

}

return false;

}

};

**Q4) undirected graph  
DFS:**

**BFS:**

**Q5)**

**Q6)**

**Q7)**

**Q8)**

**Q9)**

**Q10)**

**Q11)**

**Q12)**

**Q13)**

**Q14)**

**Q15)**

**Q16)**

**Q17)**

**Q18)**

**Q19)**

**Q20)**

**Q21)**

**Q22)**

**Q23)**

**Q24)**

**Q25)**

**Q26)**

**Q27)**

**Q28)**

**Q29)**

**Q30)**

**Q31)**

**Q32)**

**Q33)**

**Q34)**