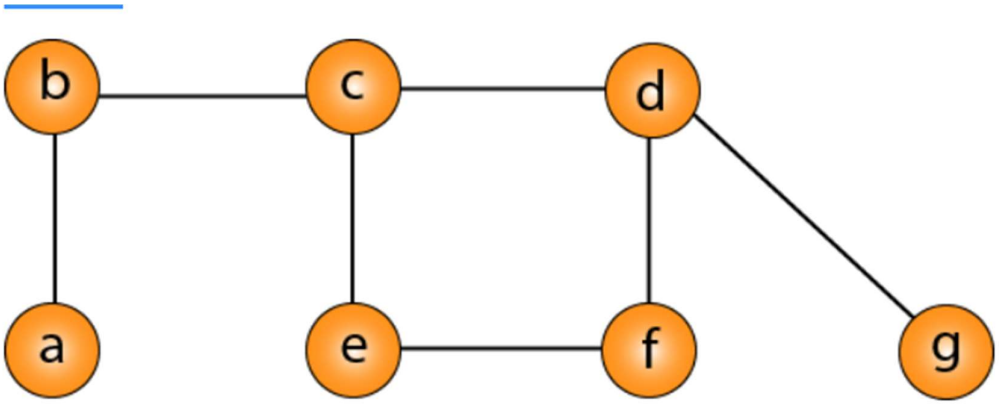
Experiment 10
DAA(Approximation Algorithms)

Name: Shaan Agarwal
UID: 2021300001
Subject: Design and Analysis Of Algorithms Lab
Date of Performance: 20th April, 2023
Date of Submission: 23rd April, 2023

Aim: To implement vertex cover problem using Approximation Algorithm

A Vertex Cover of a graph G is a set of vertices such that each edge in G is incident to at least one of these vertices.

The decision vertex-cover problem was proven NPC. Now, we want to solve the optimal version of the vertex cover problem, i.e., we want to find a minimum size vertex cover of a given graph. We call such vertex cover an optimal vertex cover C*.



Algorithm

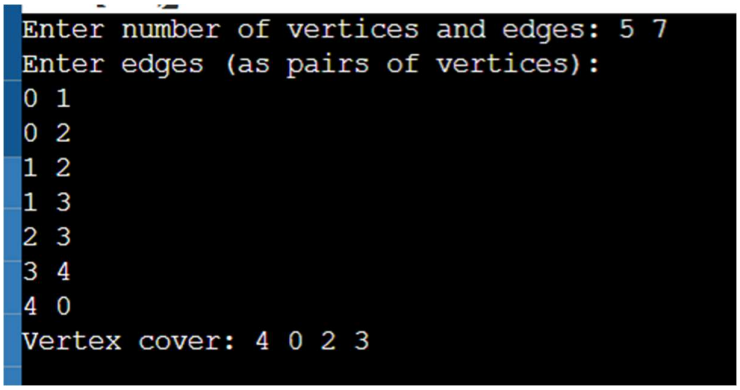
```
Approx-Vertex-Cover (G = (V, E))
{
     C = empty-set;
   E'= E;
   While E' is not empty do
    {
   Let (u, v) be any edge in E': (*)
   Add u and v to C;
   Remove from E' all edges incident to
     u or v;
    }
   Return C;
}
```

Program

```
#include <iostream>
#include <vector>
#include <algorithm>

using namespace std;

vector<int> vertexCover(int V, vector<vector<int>>& edges) {
    vector<int> cover;

    while (!edges.empty()) {
        sort(edges.begin(), edges.end(), [&](const vector<int>& a, const
vector<int>& b) {
            return count(cover.begin(), cover.end(), a[0]) +
count(cover.begin(), cover.end(), a[1]) <
                   count(cover.begin(), cover.end(), b[0]) +
count(cover.begin(), cover.end(), b[1]);
        });
```

```
        vector<int>& edge = edges.back();

        if (count(cover.begin(), cover.end(), edge[0]) == 0) {
            cover.push_back(edge[0]);
        }
        if (count(cover.begin(), cover.end(), edge[1]) == 0) {
            cover.push_back(edge[1]);
        }

        for (auto it = edges.begin(); it != edges.end();) {
            if (count(cover.begin(), cover.end(), (*it)[0]) > 0 ||
                count(cover.begin(), cover.end(), (*it)[1]) > 0) {
                it = edges.erase(it);
            } else {
                ++it;
            }
        }
    }

    return cover;
}

int main() {
    int V, E;
    cout << "Enter number of vertices and edges: ";
    cin >> V >> E;

    vector<vector<int>> edges(E, vector<int>(2));
    cout << "Enter edges (as pairs of vertices):" << endl;
    for (int i = 0; i < E; i++) {
        cin >> edges[i][0] >> edges[i][1];
    }

    vector<int> cover = vertexCover(V, edges);

    cout << "Vertex cover: ";
    for (int i : cover) {
        cout << i << " ";
    }
    cout << endl;

    return 0;
}
```

Output



Conclusion: After performing the above experiment, I have understood the vertex cover problem, and have implemented the same using approximation algorithm.