Name	Parth Gandhi
UID No.	2021300033
Class & Division	S.E. COMPS A (BATCH B)
Experiment No.	9

Aim: To implement Approximation algorithms (The vertex-cover problem)

Theory:
Approximation algorithms are algorithms that find solutions to optimization problems that are not necessarily exact, but are guaranteed to be within a certain factor of the optimal solution. One classic example of an optimization problem is the vertex-cover problem, which asks for the smallest set of vertices that cover all edges in a graph.

The vertex-cover problem is known to be NP-hard, which means that there is no known polynomial-time algorithm that can solve it exactly. However, there are many approximation algorithms that can find near-optimal solutions in polynomial time.

One such algorithm is the greedy algorithm, which starts with an empty set and iteratively adds vertices to the set that cover the most uncovered edges. This algorithm is guaranteed to find a vertex cover that is no more than twice the size of the optimal vertex cover.

Another approximation algorithm is the randomized rounding algorithm, which uses linear programming to find a fractional vertex cover and then rounds the fractional values to obtain a set of vertices that approximately cover all edges. This algorithm is also guaranteed to find a vertex cover that is no more than twice the size of the optimal vertex cover.

Algorithm:
1. Define a structure for Edge, with two character variables representing the two vertices of an edge.
2. Define a function vertexCover which takes a vector of edges and returns a set of vertices that form a vertex cover for the given edges.
3. Initialize an empty set called cover to store the vertices of the vertex cover.
4. Make a copy of the vector of edges called edges_copy.
5. Seed the random number generator with the current time.
6. Repeat the following while edges_copy is not empty:
    a. Generate a random index i between 0 and edges_copy.size()-1.
    b. Extract the edge at index i from edges_copy and store it in a variable e.
    c. Remove the extracted edge from edges_copy.
    d. Add both vertices of e to the cover set.
    e. Print a message indicating that the edge has been added.
    f. Iterate through the edges in edges_copy and remove any edge that shares a vertex with e. Print a message indicating that the edge has been removed.
7. Return the cover set.
8. In main function, prompt the user to input the number of edges and each edge in the format "u v".
9. Create a vector of edges based on the user input.
10. Call the vertexCover function passing the vector of edges as input.
11. Print the resulting vertex cover in the format "{ v1, v2, ..., vn }".

Code:
```
#include <bits/stdc++.h>
using namespace std;
```

```
struct Edge {
    char u;
    char v;

    Edge() {
        u = 0;
        v = 0;
    }

    Edge(char u, char v) {
        this->u = u;
        this->v = v;
    }
};

set<char> vertexCover(vector<Edge>& edges) {
    set<char> cover;
    vector<Edge> edges_copy = edges;
    srand(time(NULL));

    while (!edges_copy.empty()) {
        int i = rand() % edges_copy.size();
        Edge e = edges_copy[i];
        edges_copy.erase(edges_copy.begin() + i);

        cover.insert(e.u);
        cover.insert(e.v);

        cout << "Adding edge " << e.u << " " << e.v << "\n";
        for (int i = 0; i < edges_copy.size(); i++) {
            if (edges_copy[i].u == e.u || edges_copy[i].v == e.u ||
                edges_copy[i].u == e.v || edges_copy[i].v == e.v) {
                cout << "Removing edge " << edges_copy[i].u << " "
                     << edges_copy[i].v << "\n";
                edges_copy.erase(edges_copy.begin() + i);
                i--;
            }
        }
    }
    return cover;
}

int main() {
    cout << "Enter the number of edges then enter each edge in the format \"u "
            "v\" where u and v are vertices of the edge.\n";
    int n;
    cin >> n;
    vector<Edge> edges(n);
    for (int i = 0; i < n; i++) {
        char u, v;
        cin >> u >> v;
        edges[i] = Edge(u, v);
    }
```

```
    cout << "\nRunning Approximate Vertex Cover Algorithm...\n";
    set<char> cover = vertexCover(edges);
    cout << "\nVertex Cover: { ";
    for (char v : cover) {
        cout << v << ", ";
    }
    cout << "}\n" << endl;
    return 0;
}
```

Output:

```
Enter the number of edges then enter each edge in the format "u v" where u and v are vertices of the edge.
8
a b
b c
c e
c d
e d
e f
d g
d f

Running Approximate Vertex Cover Algorithm...
Adding edge c d
Removing edge b c
Removing edge c e
Removing edge e d
Removing edge d g
Removing edge d f
Adding edge a b
Adding edge e f

Vertex Cover: { a, b, c, d, e, f, }
```

Conclusion: Successfully wrote a program to implement vertex cover problem using approximation algorithm.