

NAME:	Harshal Chawan
UID:	2021300019
SUBJECT	Design and Analysis of Algorithm
EXPERIMENT NO :	09
DATE OF PERFORMANCE	17/04/2023
DATE OF SUBMISSION	23/04/2023
AIM:	To implement Approximation algorithms (The vertex-cover problem).
PROBLEM STATEMENT 1:	Approximation algorithms (The vertex-cover problem)
ALGORITHM and THEORY:	<p>THEORY :</p> <p>A vertex cover of an undirected graph is a subset of its vertices such that for every edge (u, v) of the graph, either 'u' or 'v' is in the vertex cover. Although the name is Vertex Cover, the set covers all edges of the given graph. Given an undirected graph, the vertex cover problem is to find minimum size vertex cover.</p> <p>Vertex Cover Problem is a known NP Complete problem, i.e., there is no polynomial-time solution for this unless $P = NP$. There are approximate polynomial-time algorithms to solve the problem though. Following is a simple approximate algorithm adapted from CLRS book.</p>

	<p>ALGORITHM :</p> <ol style="list-style-type: none"> 1) Initialize the result as { } 2) Consider a set of all edges in given graph. Let the set be E. 3) Do following while E is not empty. <ol style="list-style-type: none"> ...a) Pick an arbitrary edge (u, v) from set E and add 'u' and 'v' to result. ...b) Remove all edges from E which are either incident on u or v. 4) Return result.
Program	<pre> #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"; } } </pre>

```

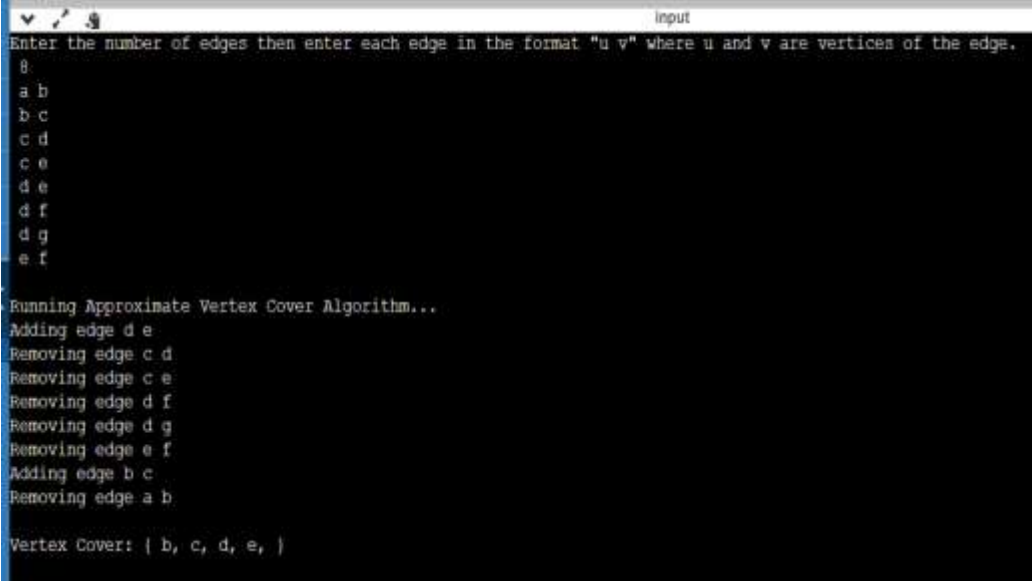
        for (int i = 0; i < edges_copy.size(); i++) {
            if (edges_copy[i].u == e.u || edges_copy[i].v == e.u ||
                edges_copy[i].u == e.v || edges_copy[i].v == e.v) {
                cout << "Removing edge " << edges_copy[i].u << " "
                    << edges_copy[i].v << "\n";
   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

// sample input
// 8
// a b
// b c
// c d
// c e
// d e

```

	<pre>// d f // d g // e f</pre>
OUTPUT:	 <pre> input 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 d c e d e d f d g e f Running Approximate Vertex Cover Algorithm... Adding edge d e Removing edge c d Removing edge c e Removing edge d f Removing edge d g Removing edge e f Adding edge b c Removing edge a b Vertex Cover: { b, c, d, e, } </pre>
CONCLUSION:	<p>I have successfully understood and implemented the concept of vertex cover problem through this experiment. I was also able to understand how to find minimum size vertex cover when an undirected graph is given.</p>

