

NAME:	Deepanshu Aggarwal
UID:	2021300002
BRANCH:	Computer engineering
BATCH:	A
SUBJECT:	DAA
EXPT NO:	10
AIM:	Approximate solution for vertex cover problem.
THEORY:	<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.</p> <p>Naive Approach:</p> <p>Consider all the subset of vertices one by one and find out whether it covers all edges of the graph. For eg. in a graph consisting only 3 vertices the set consisting of the combination of vertices are: {0,1,2}, {0,1}, {0,2}, {1,2}, {0,1,2} . Using each element of this set check whether these vertices cover all the edges of the graph. Hence update the optimal answer. And hence print the subset having minimum number of vertices which also covers all the edges of the graph.</p>
PROGRAM:	<pre> #include<iostream> #include <list> using namespace std; class Graph { int V; </pre>

```

    list<int> *adj;
public:
    Graph(int V);
    void addEdge(int v, int w);
    void printVertexCover();
};

Graph::Graph(int V)
{
    this->V = V;
    adj = new list<int>[V];
}

void Graph::addEdge(int v, int w)
{
    adj[v].push_back(w);
    adj[w].push_back(v);
}

void Graph::printVertexCover()
{
    bool visited[V];
    for (int i=0; i<V; i++)
        visited[i] = false;

    list<int>::iterator i;

    for (int u=0; u<V; u++)
    {
        if (visited[u] == false)
        {
            for (i= adj[u].begin(); i !=
adj[u].end(); ++i)

```

```

        {
            int v = *i;
            if (visited[v] == false)
            {
                visited[v] = true;
                visited[u] = true;
                break;
            }
        }
    }
}

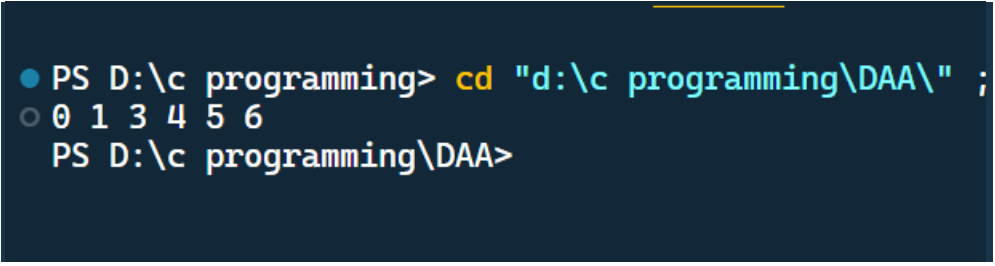
// Print the vertex cover
for (int i=0; i<V; i++)
    if (visited[i])
        cout << i << " ";
}

int main()
{
    Graph g(7);
    g.addEdge(0, 1);
    g.addEdge(0, 2);
    g.addEdge(1, 3);
    g.addEdge(3, 4);
    g.addEdge(4, 5);
    g.addEdge(5, 6);

    g.printVertexCover();

    return 0;
}

```

RESULT:	 <pre>PS D:\c programming> cd "d:\c programming\DAA\" ; 0 1 3 4 5 6 PS D:\c programming\DAA></pre>
CONCLUSION:	From this experiment, I learnt the concept of approximate value and how I can apply that approach in solving the vertex cover problem.