BHARATIYA VIDYA BHAVAN'S
SARDAR PATEL INSTITUTE OF TECHNOLOGY
Bhavan's Campus, Munshi Nagar, Andheri (West), Mumbai – 400058-India
Department of Computer Engineering

Name	Balla Mahadev Shrikrishna
UID no.	2023300010
Experiment No.	10

AIM:	To implement a vertex cover approximation algorithm.
Program 1	
PROBLEM STATEMENT :	**Details –** An Approximate Algorithm is a way of approaching NP-COMPLETENESS for the optimization problem. This technique does not guarantee the best solution. The goal of an approximation algorithm is to come as close as possible to the optimum value in a reasonable amount of time which is at the most polynomial time. Such algorithms are called approximation algorithms or heuristic algorithms. An algorithms for a problem has an approximation ratio of ρ (n) if for any input of size n the cost C of the solution produced by the algorithm is within a factor of ρ (n) of the cost C* of an optimal solution. Max(C/C*, C*/C) <= ρ (n) We have an algorithm ρ (n)-approximation if it achieves an approximation ratio of ρ (n) For maximization problem 0 < C <= C* For minimization problem 0 < C* <= C For the vertex cover problem, the optimization problem is to find the vertex cover with fewest vertices, and the approximation problem is to find the vertex cover with few vertices. Formally, 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 vertex-cover problem was proven by the 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*. **Input** – Graph G(V,E) of at least 10 vertices. **Output** – Approximate Vertex Cover for the given graph. **Submission –** 1) C/C++ source code of implementation 2) Verified output for the written source code with multiple inputs 3) One page report of Exp. 10

ALGORITHM:

1) Input: An undirected graph with V vertices and E edges.
2) Initialization: Mark all vertices as not in cover (inCover array).

3) Greedy Selection:
    a) While there are uncovered edges, pick an arbitrary edge (u, v) where neither u nor v is in the cover.
    b) Add both u and v to the cover.
4) Termination: When all edges are covered, output the selected vertices.

PROGRAM:

```
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>

#define MAX_VERTICES 100

int main() {
        int V, E;
        int graph[MAX_VERTICES][MAX_VERTICES] = {0};
        bool visited[MAX_VERTICES] = {false};
        bool inCover[MAX_VERTICES] = {false};

        // Get input
        printf("Enter the number of vertices: ");
        scanf("%d", &V);

        printf("Enter the number of edges: ");
        scanf("%d", &E);

        printf("Enter %d edges (format: u v):\n", E);
        for (int i = 0; i < E; i++) {
        int u, v;
        scanf("%d %d", &u, &v);

        // Build undirected graph
        graph[u][v] = 1;
        graph[v][u] = 1;
        }
```

```
    // Greedy approach: pick an uncovered edge and add both endpoints
to cover
    bool edgesRemain = true;
    while (edgesRemain) {
    edgesRemain = false;

    // Find an edge that is not covered
    for (int u = 0; u < V && !edgesRemain; u++) {
    for (int v = 0; v < V && !edgesRemain; v++) {
        if (graph[u][v] && !inCover[u] && !inCover[v]) {
        // Add both endpoints to cover
        inCover[u] = true;
        inCover[v] = true;
        edgesRemain = true;
        }
    }
    }
    }

    // Print result
    printf("Marked vertices: ");
    for (int i = 0; i < V; i++) {
    if (inCover[i]) {
    printf("%d ", i);
    }
    }
    printf("\n");

    return 0;
}
```

RESULT:

```
PS C:\Mahadev\SE\Sem4\DAA\Lab\Lab Sessions\Exp10> gcc vertex_cover.c
PS C:\Mahadev\SE\Sem4\DAA\Lab\Lab Sessions\Exp10> ./a.exe
Enter the number of vertices: 7
Enter the number of edges: 7
Enter 7 edges (format: u v):
0 1
0 3
1 2
1 4
2 5
4 5
2 6
Marked vertices: 0 1 2 5
PS C:\Mahadev\SE\Sem4\DAA\Lab\Lab Sessions\Exp10> ./a.exe
Enter the number of vertices: 10
Enter the number of edges: 12
Enter 12 edges (format: u v):
0 1
0 2
3 4
4 5
2 3
3 4
4 6
3 5
2 5
1 4
2 4
5 6
Marked vertices: 0 1 2 3 4 5
PS C:\Mahadev\SE\Sem4\DAA\Lab\Lab Sessions\Exp10>
```

CONCLUSION:	Name: Balla Mahadev Shrikrishna UID: 2023300010 Div: A Batch: A Exp-10 * The vertex cover problem involves selecting the smallest set of vertices in a graph such that every edge is incident to at least one vertex in the set. * Implementat^{n} · Graph Representat^{n}: Adjacency Matrix · Cover Tracking: Boolean array to mark included vertices. · Edge Check: Nested loops iterate until no uncovered edges remain. * Time Complexity & Space Complexity · T.C. = $O(V^2)$, i.e. for each edge check all possible vertex pairs. · S.C. = $O(V^2) + O(V)$ for adj. matrix & visited/cover arrays. * Although o/p is not wanted to be optimal, the sol^{n} is within a factor of 2 of the minimum, making it practically applicable for real-world prblms.