AA_LAB-10_Assignment

CE_054

Aim :- Find out the Minimum vertex cover using different method of Approximation.

Program :-

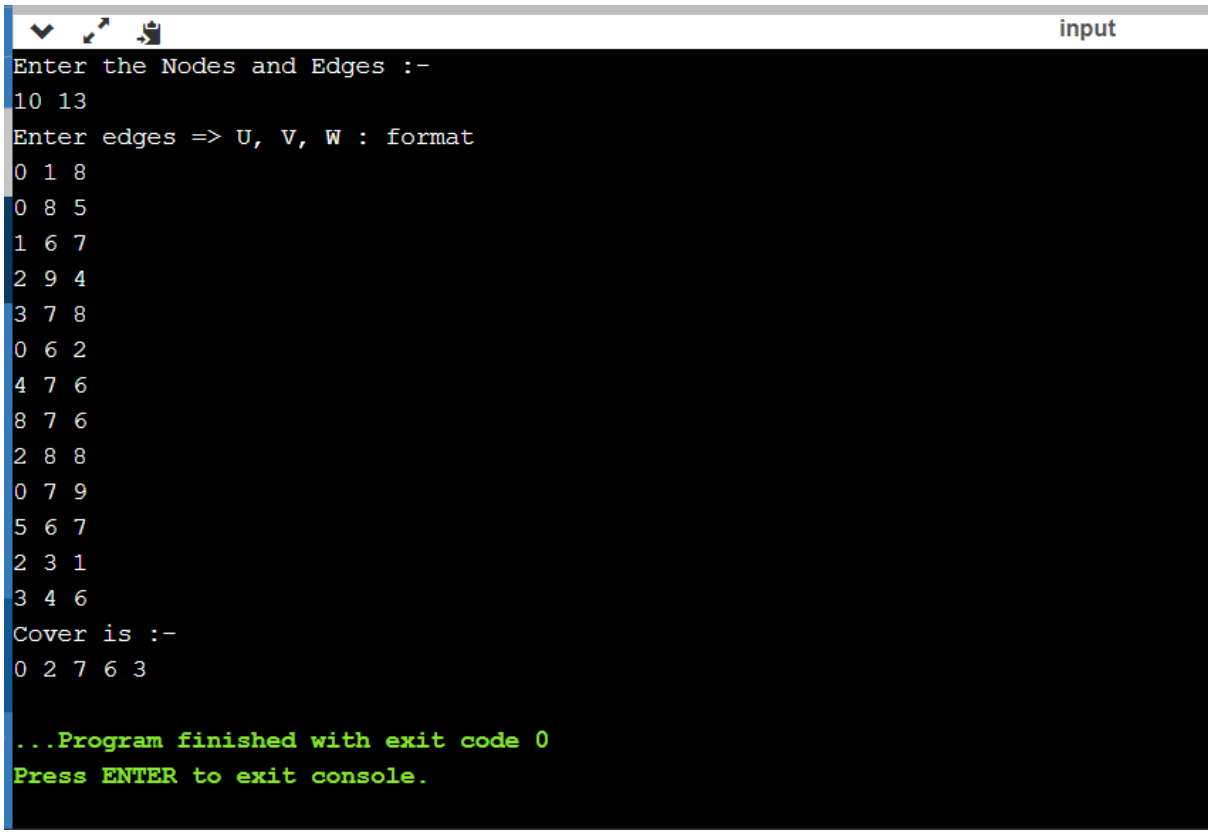
1. Approximation Algorithm for Vertex Cover using Greedy strategy (Select a vertex with the highest degree).

Code :-

```c
// Author : Dhruv B Kakadiya
#include <stdio.h>
#include <stdlib.h>
int no_of_nodes;

int minimum_cover(int **adj_matrix, int deg_matrix[], int sol[]);
int find_max_index_graph(int deg_matrix[]);

int main()
{
    int nodes, edges, _, __, u_, v_, w_, max_flow;
    printf("enter number of nodes and edges\n");
    scanf("%d %d", &nodes, &edges);
    no_of_nodes = nodes;
    int *adj_matrix[nodes], deg_matrix[nodes], sol[nodes];
    for (_ = 0; _ < nodes; _++)
    {
        adj_matrix[_] = (int *)calloc(nodes, sizeof(int));
        deg_matrix[_] = 0;
    }
    printf("enter edges in form u v w ;such that 0<= u, 0<=v \n");
    for (_ = 0; _ < edges ; _++)
    {
        scanf("%d %d %d", &u_, &v_, &w_);
        /*undirected graph*/
        adj_matrix[u_][v_] = w_;
        adj_matrix[v_][u_] = w_;
        deg_matrix[u_]++;
        deg_matrix[v_]++;
    }
    int solLen = minimum_cover(adj_matrix, deg_matrix, sol);
    printf("Solution is:\n");
    for (_ = 0; _ < solLen; _++)
    {
        printf("%d ", sol[_]);
    }
```

```
    return 0;
}

int find_max_index_graph(int deg_matrix[])
{
    int _, max_deg, index_max_degree;
    max_deg = deg_matrix[0];
    for (_ = 1; _ < no_of_nodes; _++)
    {
        if (max_deg < deg_matrix[_])
        {
            max_deg = deg_matrix[_];
            index_max_degree = _;
        }
    }
    return index_max_degree;
}

int minimum_cover(int **adj_matrix, int deg_matrix[], int sol[])
{
    int _, __, index_max_degree, max_deg, v_, u_;
    int solTop = 0;
    index_max_degree = find_max_index_graph(deg_matrix);
    max_deg = deg_matrix[index_max_degree];

    while (max_deg > 0)
    {
        sol[solTop] = index_max_degree;
        solTop++;
        u_ = index_max_degree;
        deg_matrix[u_] -= max_deg;
        for (_ = 0; _ < no_of_nodes; _++)
        {
            v_ = _;
            if (adj_matrix[u_][v_] > 0)
            {
                adj_matrix[u_][v_] = 0;
                adj_matrix[v_][u_] = 0;
                deg_matrix[v_]--;
            }
        }
        /*finding next max degree*/
        index_max_degree = find_max_index_graph(deg_matrix);
        max_deg = deg_matrix[index_max_degree];
    }

    return solTop;
}
```

Output :-



2. Approximation Algorithm for Vertex Cover which Randomly selects edges. (Algorithm which has Approximation Factor 2)

Code :-

```
#Author : Dhruv B Kakadiya

def find_max(matrix):
    maximum_degree = 0
    maximum_Node = 0

    for i in range(len(matrix)):
        countD = matrix[i].count(1)
        if countD > maximum_degree:
            maximum_degree = countD
            maximum_Node = i
    return maximum_degree, maximum_Node
```

```
def removeEdge(matrix, n):
    for i in range(len(matrix)):
        for j in range(len(matrix)):
            if i == n or j == n:
                matrix[i][j] = 0

def anyEdgeLeft(matrix):
    left = False
    for i in range(len(matrix)):
        if matrix[i].count(1) > 0:
            left = True
    return left

if __name__ == "__main__":
    n = int(input("Enter number of vertex :- \n"))
    print("Enter matrix :- \n")
    matrix = []
    maximum_degree = 0
    maximum_Node = -1
    Ran_approx = []

    for i in range(n):
        temp = list(map(int, input().split()))
        matrix.append(temp)

    while anyEdgeLeft(matrix):
        maximum_degree, maximum_Node = find_max(matrix)
        Ran_approx.append(maximum_Node)
        removeEdge(matrix, maximum_Node)

    print("Min vertex cover:", Ran_approx)
```

Output :-

```
TERMINAL   PROBLEMS   OUTPUT   DEBUG CONSOLE
6
Enter matrix :-

0 1 0 1 0 1
1 0 1 0 1 0
0 0 0 1 1 1
1 0 1 1 0 0
0 0 1 0 0 0
1 0 0 0 0 0
Min vertex cover: [0, 2, 1, 3]

D:\clg2021\AA\LAb10>python second.py
Enter number of vertex :-
5
Enter matrix :-

0 0 1 1 0
1 1 0 0 0
1 0 1 0 1
0 1 0 1 0
0 0 0 1 1
Min vertex cover: [2, 1, 4, 0, 3]

D:\clg2021\AA\LAb10>
```