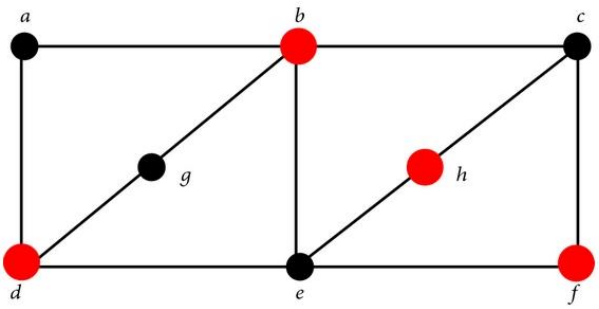
DAA Experiment-10
(Batch-A/A1)

Name	Ansari Mohammed Shanouf Valijan
UID Number	2021300004
Class	SY B.Tech Computer Engineering(Div-A)
Experiment Number	10
Date of Performance	23-04-23
Date of Submission	23-04-23

Aim:
To implement the approximation algorithm for solving Vertex-Cover Problem.

Theory:
The vertex cover problem is a well-known optimization problem in computer science that seeks to find the minimum size set of vertices that covers all the edges of a graph. In other words, it is the problem of finding a set of vertices in a graph such that every edge of the graph is incident to at least one vertex in the set. A vertex cover of a graph G is a subset of its vertices such that for every edge (u, v) in G, at least one of u or v is in the vertex cover. The size of the vertex cover is the number of vertices in the subset.

The vertex cover problem is NP-hard, meaning that there is no known efficient algorithm that can solve it in polynomial time. However, there are several algorithms that can solve it approximately or for special classes of graphs. One of the most commonly used algorithms for the vertex cover problem is the approximation algorithm that gives a solution within a factor of two of the optimal solution. The algorithm works by repeatedly selecting an arbitrary edge and adding its endpoints to the vertex cover, removing all the edges covered by these vertices, and repeating the process until all edges are covered.



The above image demonstrates a solution of the Vertex-Cover problem for the shown graph. The vertices in red are sufficient to cover/explore all the edges of the graph.

Algorithm:

[A] For solving Vertex-Cover problem-

I. Start.
II. Select an arbitrary edge of the graph.
III. Include the vertices forming this edge into the solution set.
IV. From the set of edges, remove all those edges that are connected to either of the above two vertices.
V. Repeat steps 2 to 4 till the set of edges gets empty.
VI. End.

Program:

```
//header files
#include<stdio.h>
#include<stdlib.h>
#include<stdbool.h>

//graph handling
int** graphData;
int* visited;
int* degree;
int numOfNodes;
int edgeCount=0;

//queue handling
int* nodesQueue;
int front=-1, rear=-1;
int soln[50];//to store the nodes which are to be included in the solution
int solnCount=0;

void enqueue(int elem){
    rear++;
    nodesQueue[rear]=elem;
}

int dequeue(){
    front++;
    return nodesQueue[front];
}

//function to find node with max degree
int getNodeWithMaxDeg(){
    int tempMax=0;
```

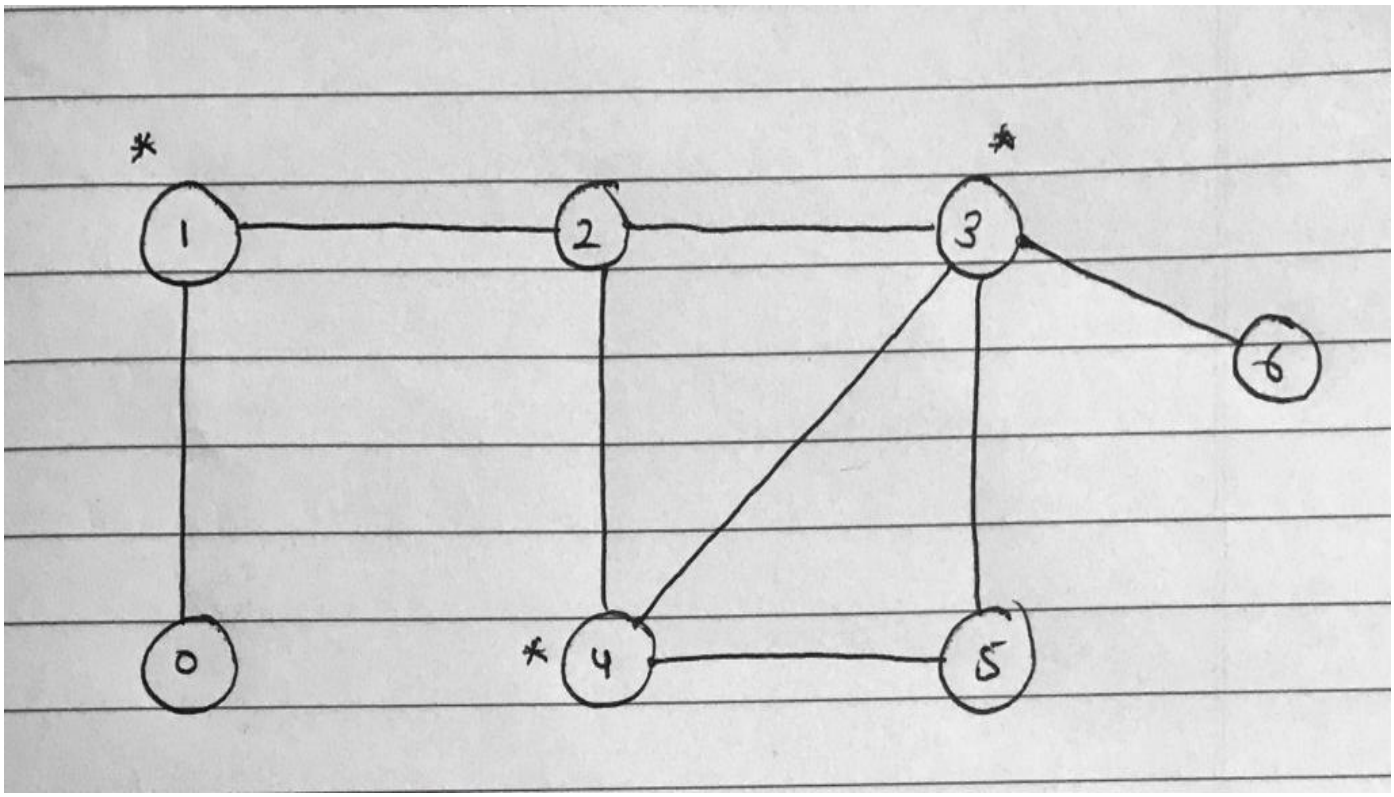
```
    for(int i=1; i<numOfNodes; i++){
        if(degree[i]>degree[tempMax])
            tempMax=i;
    }
    return tempMax;
}

//function to get approx answer of vertex cover problem
void approxVertexCover(){
    int currNode;

    while(edgeCount!=0){
        currNode=getNodeWithMaxDeg();

        printf("Current node added to the solution- %d\n",currNode);
        soln[solnCount++]=currNode;
        edgeCount-=2*degree[currNode];
        degree[currNode]=0;

        printf("Edges covered- {");
        for(int i=0; i<numOfNodes; i++){

            if(graphData[currNode][i]==1){
                degree[i]-=1;
                graphData[currNode][i]=0;
                graphData[i][currNode]=0;

                if(i>currNode)
                    printf("(%d,%d)",currNode,i);
                else
                    printf("(%d,%d)",i,currNode);

            }

        }

        printf("}\nEdges remaining- {");
        for(int i=0; i<numOfNodes; i++){
            for(int j=i+1; j<numOfNodes; j++){
                if(graphData[i][j]==1)
                    printf("(%d,%d)",i,j);
            }
        }
        printf("}\n\n");
    }

}

//main function
void main(){
    int node, degCount;

    printf("\nEnter the number of nodes in the graph -----> ");
    scanf("%d",&numOfNodes);
```

```
    graphData=(int**)malloc(numOfNodes*sizeof(int*));
    visited=(int*)malloc(numOfNodes*sizeof(int));
    degree=(int*)malloc(numOfNodes*sizeof(int));
    nodesQueue=(int*)malloc(numOfNodes*sizeof(int));

    for(int i=0; i<numOfNodes; i++){
        visited[i]=0;
        graphData[i]=(int*)malloc(numOfNodes*sizeof(int));
        for(int j=0; j<numOfNodes; j++){
            graphData[i][j]=0;
        }
    }

    printf("Nodes 0 to %d were initialized...\n",numOfNodes-1);

    printf("Enter the information regarding graph connectivity-\n\n");
    for(int i=0; i<numOfNodes; i++){
        degCount=0;
        printf("Enter the nodes connected to %d(-1 to terminate)- ",i);
        while(true){
            scanf("%d",&node);
            if(node==-1){
                break;
            }
            degCount++;
            graphData[i][node]=1;
        }
        edgeCount+=degCount;
        degree[i]=degCount;
    }

    //solving the vertex cover problem to get approx solution
    printf("\nFollowing data was obtained while solving the vertex cover problem-\n\n");
    approxVertexCover();
    printf("Hence, to cover all edges of the graph, following nodes should be explored-
{");
    for(int i=0; i<solnCount-1; i++)
        printf("%d, ",soln[i]);
    printf("%d}",soln[solnCount-1]);
    printf("\n\n");

    free(visited);
    free(nodesQueue);
    free(degree);

    for(int i=0; i<numOfNodes; i++){
        free(graphData[i]);
    }

    free(graphData);


}
```

Implementation:

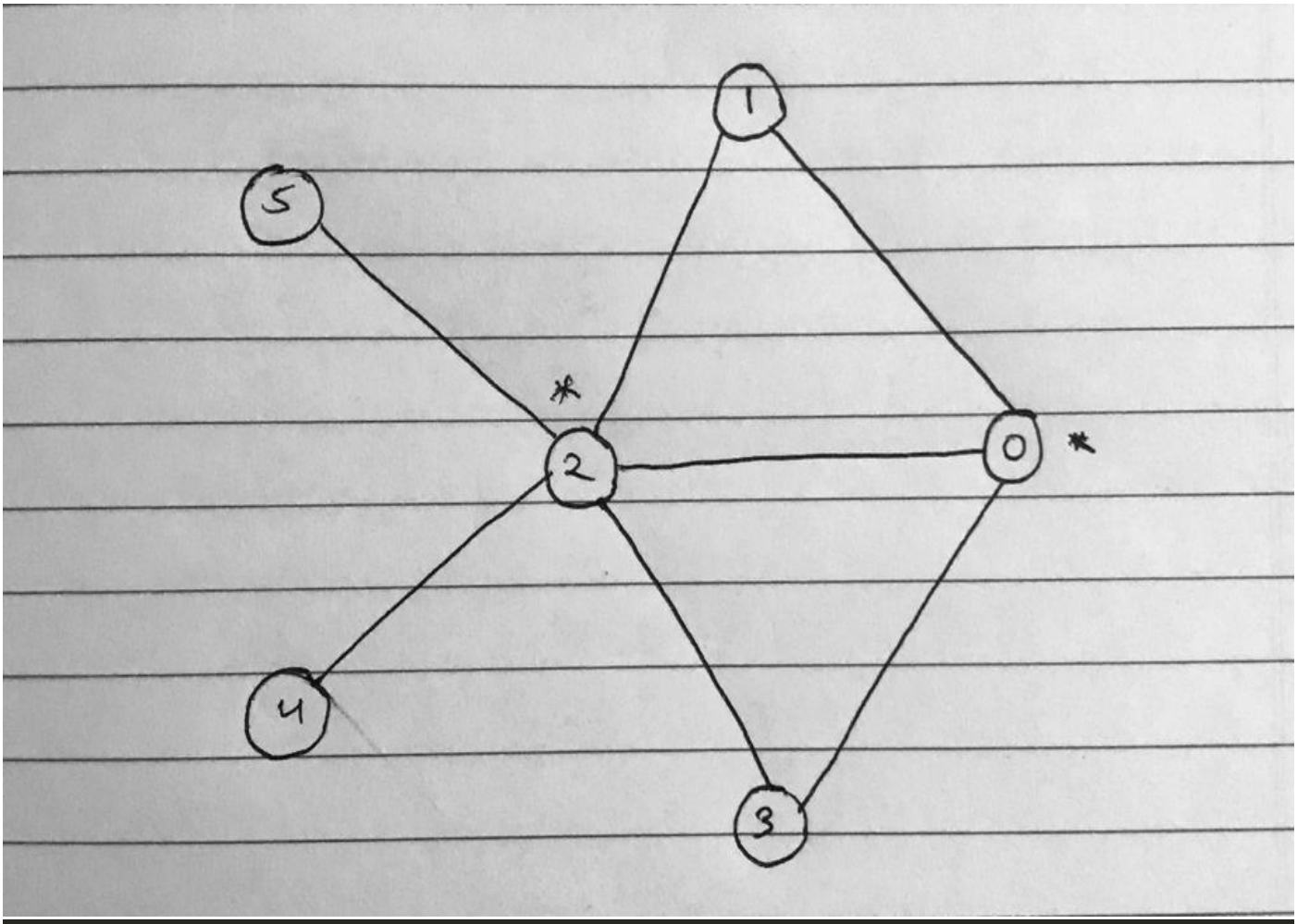
Note that the asterisk next to a vertex indicates, that vertex is a part of the solution.

Example-1-




```
Enter the number of nodes in the graph -----> 7
Nodes 0 to 6 were initialized...
Enter the information regarding graph connectivity-

Enter the nodes connected to 0(-1 to terminate)- 1 -1
Enter the nodes connected to 1(-1 to terminate)- 0 2 -1
Enter the nodes connected to 2(-1 to terminate)- 1 3 4 -1
Enter the nodes connected to 3(-1 to terminate)- 2 4 5 6 -1
Enter the nodes connected to 4(-1 to terminate)- 2 3 5 -1
Enter the nodes connected to 5(-1 to terminate)- 3 4 -1
Enter the nodes connected to 6(-1 to terminate)- 3 -1

Following data was obtained while solving the vertex cover problem-

Current node added to the solution- 3
Edges covered- {(2,3)(3,4)(3,5)(3,6)}
Edges remaining- {(0,1)(1,2)(2,4)(4,5)}

Current node added to the solution- 1
Edges covered- {(0,1)(1,2)}
Edges remaining- {(2,4)(4,5)}

Current node added to the solution- 4
Edges covered- {(2,4)(4,5)}
Edges remaining- {}

Hence, to cover all edges of the graph, following nodes should be explored- {3, 1, 4}
```

Example-2-




```
Enter the number of nodes in the graph -----> 6
Nodes 0 to 5 were initialized...
Enter the information regarding graph connectivity-

Enter the nodes connected to 0(-1 to terminate)- 1 2 3 -1
Enter the nodes connected to 1(-1 to terminate)- 0 2 -1
Enter the nodes connected to 2(-1 to terminate)- 0 1 3 4 5 -1
Enter the nodes connected to 3(-1 to terminate)- 0 2 -1
Enter the nodes connected to 4(-1 to terminate)- 2 -1
Enter the nodes connected to 5(-1 to terminate)- 2 -1

Following data was obtained while solving the vertex cover problem-

Current node added to the solution- 2
Edges covered- {(0,2)(1,2)(2,3)(2,4)(2,5)}
Edges remaining- {(0,1)(0,3)}

Current node added to the solution- 0
Edges covered- {(0,1)(0,3)}
Edges remaining- {}

Hence, to cover all edges of the graph, following nodes should be explored- {2, 0}
```

Following example was solved to understand the algorithm-

Example -2-

Edges: {(0,1), (0,2), (0,3), (1,2), (2,3), (2,5), (2,4)}
Soln: { }

Selecting an arbitrary edge, say (0,2):

hence, In Solution,

we add vertices 2 & 0

∴ Solution: {2,0}

Further, we remove all the edges that are either connected to (2) or (0) or both.

i.e,
we remove (0,1), (0,2), (0,3), (1,2), (2,3), (2,5) and (2,4) from the set of edges

Thus, we obtain,

Edges: { }

Solution: {2,0}

Since, set of Edges is empty, problem is solved.

Hence, to cover all the vertices of the graph 2 & 0 nodes must be explored.

Teacher's Signature:

Inference:

While trying to write a program for this problem using the approximation algorithm, I realized that we can also solve it focusing on the degree of each vertex rather than selecting an arbitrary edge. Basically, we move to the vertex having maximum degree in an attempt to explore maximum number of edges. Further, we remove the explored edges from the set and repeat the above steps till all the edges have been explored. This gives a better solution to the problem.

Conclusion:

By performing this experiment, I was able to understand the approximation algorithm for Vertex-Cover problem. I was also able to implement the same and verify the results.