# VASAVI COLLEGE OF ENGINEERING

(AUTONOMOUS)
(Affiliated to Osmania University)
Hyderabad - 500 031.

DEPARTMENT OF : CSE

NAME OF THE LABORATORY : DAA LAB

Name K.S.I.SIVANI Roll No. -052 Page No. 16

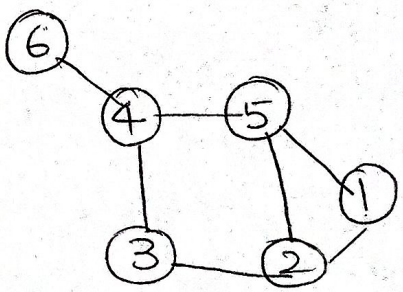
## PRELAB QUESTIONS-2

1) List the asymptotic notations used to represent time complexity of an algorithm.

→ Asymptotic notation is used to describe the running time of an algorithm: big O, big Theta($\Theta$) and big Omega($\Omega$)

2) List different methods to represent a graph. Represent the graph given below.

Graph can be represented in the form of adjacency list and adjacency matrix.




| | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| 1 | 0 | 1 | 0 | 0 | 1 | 0 |
| 2 | 1 | 0 | 1 | 0 | 1 | 0 |
| 3 | 0 | 1 | 0 | 1 | 0 | 0 |
| 4 | 0 | 0 | 1 | 0 | 1 | 1 |
| 5 | 1 | 1 | 0 | 1 | 0 | 0 |
| 6 | 0 | 0 | 0 | 1 | 0 | 0 |

3) Which graph traversal can be used to determine the connected components in a graph?

→ We can either use BFS (or) DFS to determine the connected components in a graph.

4) Write the algorithm to detect a cycle in a graph using BFS. Compute the time complexity:

```
int BFS(int k, int n)
 { for(i=1;i<=n;i++)  for i:= 1 to n
   { if(vis[i]==0)
     {BFS(i,n);
       count++; } }
     return count;
}
```

5) Write the algorithm to determine the degree of a graph. Compute the time complexity.

```
int degree(int a[m,10], int m, int x)
 { int j, c:=0;
   int y; y:=x-1;
```

```
for j:=0 to m-1
{ if (a[y][j]==1)
    { c++; }
  }
    return c;
}
```

## PRELAB PROGRAMS -2:

1) Implement DFS traversal by representing in Adjacency List:

```
#include <stdio.h>
#include <stdlib.h>
struct node
{ int vertex;
  struct node * next;};
struct node *createNode(int v);
struct Graph
{ int numVertices;
  int *visited;
  struct node **adjLists;};
```

```
void DFS(struct Graph *graph, int vertex)
{ struct node *adjlist = graph->adjLists[vertex];
  struct node *temp = adjlist;
  graph->visited[vertex] = 1;
  printf("Visited %d\n", vertex);
  while(temp!=NULL)
  {int connectedVertex = temp->vertex;
   if(graph->visited[connectedVertex]==0)
    {DFS(graph, connectedVertex);}
   temp = temp->next;}}
Struct node *createNode(int v)
{ struct node *newNode = malloc(sizeof(struct node));
  newNode->vertex = v;
  newNode->next = NULL;
  return newNode;}
struct Graph *createGraph(int vertices)
 { struct Graph * graph = malloc(sizeof(struct
                                        Graph));
   graph->numVertices = vertices;
```

```
graph->adjLists = malloc(vertices * sizeof(struct node*));
graph->visited = malloc(vertices * sizeof(int));
int i;
for(i=0; i<vertices; i++)
{ graph->adjLists[i] = NULL;
  graph->visited[i] = 0; }
return graph; }
void addEdge(struct Graph *graph, int src, int dest)
{ struct node *newNode = createNode(dest);
  newNode->next = graph->adjLists[src];
  graph->adjLists[src] = newNode;
  newNode = createNode(src);
  newNode->next = graph->adjLists[dest];
  graph->adjLists[dest] = newNode; }

void printGraph(struct Graph *graph)
{int v;
 for(v=0; v<graph->numVertices; v++)
{ struct node *temp = graph->adjLists[v];
  printf("\n Adjacency list of vertex %d\n", v);
```

* Output:

```
Adjacency list of vertex 0
2 -> 1 ->
Adjacency list of vertex 1
2 -> 0 ->
Adjacency list of vertex 2
3 -> 1 -> 0 ->
Adjacency list of vertex 3
2 ->
visited 2
visited 3
visited 1
visited 0
```



```
while (temp)
{ printf("%d->", temp->vertex);
  temp= temp-> next; }
}
int main()
{ struct Graph *graph = createGraph(4);
  addEdge(graph, 0, 1);
  addEdge(graph, 0, 2);
  addEdge(graph, 1, 2);
  addEdge(graph, 2, 3);
  printGraph(graph);
  DFS(graph, 2);
  return 0; }
```

2) Implement BFS by representing graph in Adjacency Matrix:

```
#include <stdio.h>
int n, i, j, queue[10], front = -1, rear = -1, visited[10];
int adj[10][10];
```

```
void bfs(int v)
{ for(i=1;i<=n; i++)
  { if (adj[v][i] && !visited[i])
     { queue[++rear]=i; }
     if (front<=rear)
       { visited[queue[front]]=1;
          bfs(queue[front++]); }}
}
void main()
{ int n;
  printf("Enter the no. of vertices: ");
  scanf("%d", &n);
  for(i=0; i<=n-1; i++)
   { queue[i] = -1;
      visited[i] = -1; }
  printf("Enter the graph data in matrix form:\n");
  for(i=0;i<=n-1;i++)
  { for(j=0;j<=n-1;j++)
    { scanf("%d", &adj[i]); }}
```

Output:

```
Enter the no. of vertices :4
Enter graph data in matrix form:
0 1 1 0
1 0 0 1
1 0 0 1
0 1 1 0
Enter starting vertex : 1
Node which is reachable are:
0 1 2 3
```

VASAVI COLLEGE OF ENGINEERING
(AUTONOMOUS)
(Affiliated to Osmania University)
Hyderabad - 500 031.
DEPARTMENT OF : CSE
NAME OF THE LABORATORY : DAA LAB
Name K.S.I.SIVANI Roll No. -052 Page No. 23

```
printf("Enter the starting vertex:");
scanf("%d",&v);
bfs(v);
printf("Node which is reachable are:\n");
for(i=0;i<=n-1;i++)
{ if(visited[i])
  {printf("%d\t",i);}
  else
  {printf("BFS is not possible");}}
return 0;
}
```

3) Implement connected components with appropriate graph traversal.

```
#include <stdio.h>
#include <stdlib.h>
int q[20], front=rear=-1;
int a[10][10], vis[10];
```

```
void add (int item)
{ if (rear ==19)
  { printf(" Queue Full"); }
  else
  { if(rear ==-1)
     { q[++rear]=item;
       front++; }
     else
     { q[++rear] =item; }
  }
}
int delete()
{ int k;
  if((front >rear) || (front ==-1))
     { return 0; }
  else
  { k = q[front++] ; return k; }}
```

```
int BFS (int s, int n)
{ int p, i, count=0;
  add(s);
  p=delete();
  if (p!=0)
  { printf("%d", p);
    while (p!=0)
    { for(i=0; i<=n; i++)
      { if(a[p][i]!=0) && (vis[i]==0))
        { add[i]; vis[i]=1; }
        p=delete();
      if(p!=0)
      { printf("%d", p); }
    }
  for(i=1; i<=n; i++)
    { if(vis[i]==0)
      { BFS(i, n);
        count++; }}}} return count; }
```

Output:

Enter the no. of vertices: 4

Enter 1 if there is an edge else 0: 0

1
0
1
1
0
1
0
0
1
0
1
1
0
1
0

Enter the source vertex: 1

1324 No. of islands: 1

**VASAVI COLLEGE OF ENGINEERING**

(AUTONOMOUS)
(Affiliated to Osmania University)
Hyderabad - 500 031.

DEPARTMENT OF : CSE

NAME OF THE LABORATORY : DAA LAB

Name K.S.T.SIVANI Roll No. -052 Page No. 26

```
int main()
{ int m,n,i,j;
  printf("Enter the no. of vertices:");
  scanf("%d",&n);
  printf("Enter 1 if there is an edge else 0:");
  for(i=0;i<n;i++)
  {for(j=0;j<n;j++)
   {scanf("%d",&a[i][j]);}}
printf("Enter the source vertex:");
scanf("%d",&m);
printf("No. of islands: %d",BFS(m,n));
 return 0;
 }
```

## LAB PROGRAMS - 2:

### 1) Topological Sort:

```
#include <stdio.h>
int vis[10] ={0};
int a[10][10] = {0};
void fun1(int n)
{ int i,j,count;
for(i=0;i<n;i++)
{ count=0;
 if(vis[i]==0)
 { for(j=0;j<n;j++)
{ ~~for~~ if((a[j][i]==1)&&(vis[j]!=1)) {
 { count+=1;}}
 if(count==0)
 {printf("%d",i+1);
  vis[i]=1;}}
} return; }
```

Output:

Enter no. of vertices: 5

Enter adjacency matrix: 0

1
0
1
0
0
0
1
1
0
0
0
0
0
0
0
0
1
0
1
0
0
0
0
0
1

1 2 4 5 3

**VASAVI COLLEGE OF ENGINEERING**

(AUTONOMOUS)
(Affiliated to Osmania University)
Hyderabad - 500 031.

DEPARTMENT OF : CSE

NAME OF THE LABORATORY : DAA LAB

Name K.S.I.SIVANI Roll No. -052 Page No. 28

```
int main()
{ int n,i,j;
  printf("Enter no. of vertices");
  scanf("%d", &n);
  printf("Enter adjacency matrix:");
  for(i=0;i<n;i++)
  {for(j=0;j<n;j++)
   { scanf("%d", &a[i][j]);}}
  fun1(n);
  for(i=0;i<n;i++)
  { if(vis[i]==0)
   {fun1(n);}}
    return 0;}
```

# VASAVI COLLEGE OF ENGINEERING

(AUTONOMOUS)
(Affiliated to Osmania University)
Hyderabad - 500 031.

DEPARTMENT OF : CSE

NAME OF THE LABORATORY : DAALAB

Name K.S.T.SIVANI Roll No. -052 Page No. 29

2) Implement a program to find minimum & maximum element using divide and conquer.

```
#include<stdio.h>
int min,max,min1,max1;
int a[]={17,5,-6,8,12,43};
int minmax(int i,int j)
{int mid=(i+j)/2;
 if(i==j-1)
 {if(a[i]>a[j])
   {max1=a[i];
    min1=a[j];}
  else
   {max1=a[j];
    min1=a[i];}
 }
else if(i==j)
{min1=max1=a[i];}
else
{minmax(i,mid);
 minmax(mid+1,j);}
```

```
if(max<max1)
{  max=max1;}
if (min>min1)
{ min=min1;}}
int main()
{minmax(0,5);
 printf("%d %d",min,max);
 return 0;}.
```

Output:

-6 43.