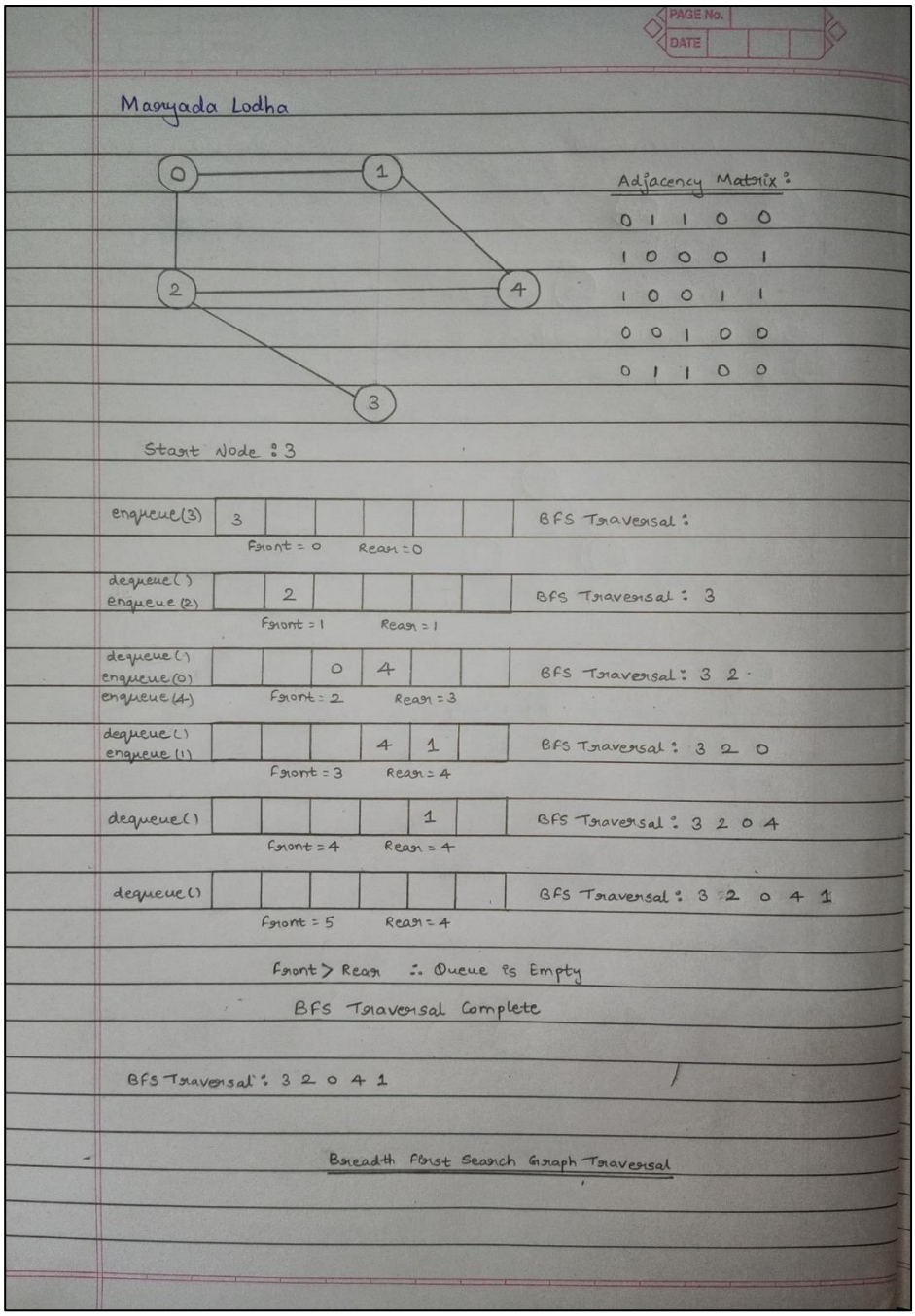
NAME :	Maryada Lodha
EXPERIMENT NO :	08
DATE :	05-12-2022

AIM :	Implementation of Breadth First Search in Graph

ALGORITHM :	int main() 1] Read the number of vertices from the user 2] Initialize the two-dimensional adjacency matrix with 0 3] Initial the visited array with 0 4] Read the edge information of the vertices from the user with the help of start and end node 5] Display the adjacency matrix 6] Read the Start Node from the user 7] Enqueue Start Node and print Start Node 8] Mark visited[Start Node] as 1 9] Repeat Step 10 until the queue is empty (FRONT > REAR) 10] Call the dequeue function, assign the return value to the variable Vertex and repeat step 11 for each vertex 11] Enqueue and print the connected and unvisited vertex of the Vertex and mark them as 1 in the visited array 12] End void enqueue(int a) 1] If REAR = = MAX – 1, then Print “Queue is Full” Go to Step 2

	Else If FRONT = = -1 and REAR = = -1, then Set FRONT = REAR = 0 Set Queue[REAR] = a Else Set REAR = REAR + 1 Set Queue[REAR] = a [End of If] 2] End int dequeue(void) 1] If FRONT = = -1 or FRONT > REAR, then Print "Queue is Empty" Else Set Value = Queue[FRONT] Set FRONT = FRONT + 1 [End of If] 2] Return Value

PROBLEM SOLVING :



PAGE No.
DATE

Maryada Lodha

Adjacency Matrix :

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

Start Node : 3

enqueue(3)	3						BFS Traversal :
	Front = 0			Rear = 0			
dequeue() enqueue(2)		2					BFS Traversal : 3
		Front = 1			Rear = 1		
dequeue() enqueue(0) enqueue(4)			0	4			BFS Traversal : 3 2
			Front = 2		Rear = 3		
dequeue() enqueue(1)				4	1		BFS Traversal : 3 2 0
			Front = 3		Rear = 4		
dequeue()					1		BFS Traversal : 3 2 0 4
				Front = 4		Rear = 4	
dequeue()							BFS Traversal : 3 2 0 4 1
				Front = 5		Rear = 4	

Front > Rear ∴ Queue is Empty

BFS Traversal Complete

BFS Traversal : 3 2 0 4 1

Breadth First Search Graph Traversal

CODE :

```
#include <stdio.h>

#define max 10


int front=-1,rear=-1;

int queue[max];


void enqueue(int a);

int dequeue(void);


int main()

{

   int n,i,j,edge,startNode,vertex,choice;

   printf("\nEnter the Number of Vertices : ");

   scanf("%d",&n);

   int matrix[n][n];

   int visited[n];

   for(i=0;i<n;i++)

   {

      visited[i]=0;

      for(j=0;j<n;j++)

      {

         matrix[i][j]=0;

      }

   }
```

```
do
{
   printf("\nEnter Start and End of Edge : \n");
   scanf("%d%d",&i,&j);
   matrix[i][j]=1;
   printf("Add Edge? (0/1) : ");
   scanf("%d",&choice);
}while(choice==1);

printf("\nAdjacency Matrix : \n");
for(i=0;i<n;i++)
{
   for(j=0;j<n;j++)
   {
      printf("%d  ",matrix[i][j]);
   }
   printf("\n");
}

printf("\nEnter Start Node : ");
scanf("%d",&startNode);
printf("\nBreadth First Search Graph Traversal : ");
printf("%d  ",startNode);
enqueue(startNode);
visited[startNode]=1;
```

```
   while(front<=rear)
   {
      vertex=dequeue();
      for(i=0;i<n;i++)
      {
         if(matrix[vertex][i]==1 && visited[i]==0)
         {
            printf("%d  ",i);
            enqueue(i);
            visited[i]=1;
         }
      }
   }
   return 0;
}

void enqueue(int a)
{
   if(rear==max-1)
   {
      printf("\nQueue is Full");
   }
   else if(front==-1 && rear==-1)
   {
```

```
        front=0;
        rear=0;
        queue[rear]=a;
    }
    else
    {
        rear=rear+1;
        queue[rear]=a;
    }
}

int dequeue(void)
{
    int val;
    if(front==-1 || front>rear)
    {
        printf("\nQueue is Empty");
    }
    else
    {
        val=queue[front];
        front=front+1;
        return val;
    }
}
```

OUTPUT :

```
Enter the Number of Vertices : 5

Enter Start and End of Edge :
0
1
Add Edge? (0/1) : 1

Enter Start and End of Edge :
1
0
Add Edge? (0/1) : 1

Enter Start and End of Edge :
0
2
Add Edge? (0/1) : 1

Enter Start and End of Edge :
2
0
Add Edge? (0/1) : 1

Enter Start and End of Edge :
1
4
Add Edge? (0/1) : 1

Enter Start and End of Edge :
4
1
Add Edge? (0/1) : 1
```

```
Enter Start and End of Edge :
2
4
Add Edge? (0/1) : 1

Enter Start and End of Edge :
4
2
Add Edge? (0/1) : 1

Enter Start and End of Edge :
2
3
Add Edge? (0/1) : 1

Enter Start and End of Edge :
3
2
Add Edge? (0/1) : 0

Adjacency Matrix :
0  1  1  0  0
1  0  0  0  1
1  0  0  1  1
0  0  1  0  0
0  1  1  0  0

Enter Start Node : 3

Breadth First Search Graph Traversal : 3  2  0  4  1
```