Experiment 18 **Date: 15.12.2023**

BFS and DFS

Aim:

Program to implement BFS and DFS on a connected undirected graph

Algorithm:

main()

```
1. start
2. declare q[20],top=-1, front=-1,rear=-1,a[20][20],vis[20],stack[20] globally
3. declare n,i,s,ch,j
4. char c,dummy
5. for(i=1;i<=n;i++)
       for(j=1;j<=n;j++)
            read a[i][j]
6. display adjacency matrix
   for(i=1;i<=n;i++)
       for(j=1;j<=n;j++)
              display a[i][j]
7. while(c==y or c==Y)
       for(i=1;i<=n;i++)
           vis[i]=0
       display Menu 1.BFS 2.DFS
       read ch
       if(ch==1)
           bfs(s,n)
       if(ch==2)
           dfs(s,n)
       display "do you want to continue(y/n)
       read  dummy
       read   c
 8.stop
```

void bfs(s,n)

```
1. declare p,i
2. call enqueue(s)
3. vis[s]=1
4. p= call dequeue()
5. if(p!=0)
     Display p
6. while(p!=0)
      for(i=1;i<=n;i++)
           if((a[p][i]!=0)&&(vis[i]==0))
                 enqueue(i)
```

```
                vis[i]=1
          p=dequeue()
          if(p!=0)
              display p
 7. for(i=1;i<=n;i++)
      if(vis[i]==0)
           bfs(i,n)
 8. Exit
```

void enqueue(item)

```
1.if(rear==19)
     Display "QUEUE FULL"
   Else
       if(rear==-1)
            q[++rear]=item
            front=front+1
      else
           q[++rear]=item
 2. exit
```

int dequeue()

```
 1.declare k;
 2. if((front>rear)||(front==-1))
       return(0)
    else
      k=q[front++]
      return(k)
 3. exit
```

void dfs(s, n)

```
 1.declare i,k
 2.  push(s)
 3. vis[s]=1
 4. k=pop()
 5. if(k!=0)
       display k
6. while(k!=0)
      for(i=1;i<=n;i++)
          if((a[k][i]!=0)&&(vis[i]==0))
              push(i)
              vis[i]=1;
      k=pop();
      if(k!=0)
         display k
 7.for(i=1;i<=n;i++)
      if(vis[i]==0)
          dfs(i,n)
 8. exit
```

void push(item)

```
1.if(top==19)
     display"Stack overflow "
   else
     stack[++top]=item;
```

int pop()

```
1. declare k
2.if(top==-1)
       return(0)
  else
      k=stack[top--]
      return(k)
3. exit
```

Program

```
#include<stdio.h>
int q[20],top=-1,front=-1,rear=-1,a[20][20],vis[20],stack[20];
int dequeue();
void enqueue(int item);
void bfs(int s,int n);
void dfs(int s,int n);
void push(int item);
int pop();
void main(){
        int n,i,s,ch,j;
        char c,dummy;
        printf("ENTER THE NUMBER VERTICES ");
        scanf("%d",&n);
        for(i=1;i<=n;i++){
                for(j=1;j<=n;j++){
                        printf("ENTER 1 IF %d HAS A NODE WITH %d ELSE 0 ",i,j);
                        scanf("%d",&a[i][j]);
                        }
                        }
printf("THE ADJACENCY MATRIX IS\n");
for(i=1;i<=n;i++){
        for(j=1;j<=n;j++){
        printf(" %d",a[i][j]); }
        printf("\n"); }
do{
        for(i=1;i<=n;i++)
        vis[i]=0;
        printf("\nMENU");
        printf("\n1.B.F.S");
        printf("\n2.D.F.S");
        printf("\nENTER YOUR CHOICE");
        scanf("%d",&ch);
        printf("ENTER THE SOURCE VERTEX :");
        scanf("%d",&s);
        switch(ch){
                case 1:bfs(s,n);
                        break;
                case 2:
                        dfs(s,n);
                        break;
                        }
printf("DO U WANT TO CONTINUE(Y/N) ? ");
scanf("%c",&dummy);
scanf("%c",&c);
}while((c=='y')||(c=='Y'));
}

void bfs(int s,int n){
        int p,i;
        enqueue(s);
        vis[s]=1;
        p=dequeue();
        if(p!=0)
        printf(" %d",p);
while(p!=0){
        for(i=1;i<=n;i++)
        if((a[p][i]!=0)&&(vis[i]==0))
        {
        enqueue(i);
        vis[i]=1;
        }
        p=dequeue();
        if(p!=0)
        printf(" %d ",p);
        }
        for(i=1;i<=n;i++)
        if(vis[i]==0)
        bfs(i,n);
        }

void enqueue(int item){
        if(rear==19)
        printf("QUEUE FULL");
        else
        {
        if(rear==-1){
```

```
        q[++rear]=item;
        front++; }
        else{
        q[++rear]=item;
        }
}
int dequeue(){
        int k;
        if((front>rear)||(front==-1))
        return(0);
        else{
        k=q[front++];
        return(k);}
        }
void dfs(int s,int n){
        int i,k;
        push(s);
        vis[s]=1;
        k=pop();
        if(k!=0)
        printf(" %d ",k);
        while(k!=0){
        for(i=1;i<=n;i++)
        if((a[k][i]!=0)&&(vis[i]==0))
        {
        push(i);
        vis[i]=1;
        }
        k=pop();
        if(k!=0)
        printf(" %d ",k);
        }
        for(i=1;i<=n;i++)
        if(vis[i]==0)
        dfs(i,n);
        }
void push(int item){
if(top==19)
printf("Stack overflow ");
else
stack[++top]=item;
}
int pop()
{
int k;
if(top==-1)
return(0);
else{
        k=stack[top--];
        return(k);}
        }
```

Output

```
mits@mits-Lenovo-S510:~/Desktop/s1mca$ gcc DFS.c
mits@mits-Lenovo-S510:~/Desktop/s1mca$ ./a.out
ENTER THE NUMBER VERTICES 4
ENTER 1 IF 1 HAS A NODE WITH 1 ELSE 0 0
ENTER 1 IF 1 HAS A NODE WITH 2 ELSE 0 1
ENTER 1 IF 1 HAS A NODE WITH 3 ELSE 0 1
ENTER 1 IF 1 HAS A NODE WITH 4 ELSE 0 1
ENTER 1 IF 2 HAS A NODE WITH 1 ELSE 0 1
ENTER 1 IF 2 HAS A NODE WITH 2 ELSE 0 0
ENTER 1 IF 2 HAS A NODE WITH 3 ELSE 0 0
ENTER 1 IF 2 HAS A NODE WITH 4 ELSE 0 0
ENTER 1 IF 3 HAS A NODE WITH 1 ELSE 0 1
ENTER 1 IF 3 HAS A NODE WITH 2 ELSE 0 0
ENTER 1 IF 3 HAS A NODE WITH 3 ELSE 0 0
ENTER 1 IF 3 HAS A NODE WITH 4 ELSE 0 0
ENTER 1 IF 4 HAS A NODE WITH 1 ELSE 0 1
ENTER 1 IF 4 HAS A NODE WITH 2 ELSE 0 0
ENTER 1 IF 4 HAS A NODE WITH 3 ELSE 0 0
ENTER 1 IF 4 HAS A NODE WITH 4 ELSE 0 0
THE ADJACENCY MATRIX IS
 0 1 1 1
 1 0 0 0
 1 0 0 0
 1 0 0 0

MENU
1.B.F.S
2.D.F.S
ENTER YOUR CHOICE1
ENTER THE SOURCE VERTEX :3
 3 1  2  4 DO U WANT TO CONTINUE(Y/N) ? y

MENU
1.B.F.S
2.D.F.S
ENTER YOUR CHOICE2
ENTER THE SOURCE VERTEX :2
 2  1  4  3 DO U WANT TO CONTINUE(Y/N) ? n
mits@mits-Lenovo-S510:~/Desktop/s1mca$
```

Experiment 19 Date: 20.12.2023

Prims's Algorithm

Aim

Program to implement Prim's algorithm for finding the minimum cost spanning tree.

Algorithm:

```
1. Start
2. Declare globally  a,b,u,v,i,ne=1,visited[10],cost[10][10]
```

Main()

```
1. Read no of nodes
2. for(i=1;i<=n;i++)
     for(j=1;j<=n;j++)
        read the cost
        if(cost[i][j]==0)
       cost[i][j]=999
3.   set visited[1]=1

4.   while(ne<n)
        for(i=1,min=999;i<=n;i++)
            for(j=1;j<=n;j++)
                if(cost[i][j]<min)
                 if(visited[i]!=0)
                    min=cost[i][j];
                      a=u=i;
                      b=v=j;

        if(visited[u]==0 || visited[v]==0)
            display edge and cost
             mincost+=min
             visited[b]=1
           cost[a][b]=cost[b][a]=999;
5.   display minimum cost
```

Program

```
#include<stdio.h>

int a,b,u,v,n,i,j,ne=1;
int visited[10]={0},min,mincost=0,cost[10][10];
void main()
{

 printf("Enter the number of nodes:");
 scanf("%d",&n);

 for(i=1;i<=n;i++)
  for(j=1;j<=n;j++)
  {
   printf("ENTER THE COST OF %d & %d : ",i,j);
   scanf("%d",&cost[i][j]);
   if(cost[i][j]==0)
    cost[i][j]=999;
  }

 visited[1]=1;
 printf("\n");
 while(ne<n)
 {
  for(i=1,min=999;i<=n;i++)
   for(j=1;j<=n;j++)
    if(cost[i][j]<min)
     if(visited[i]!=0)
     {
      min=cost[i][j];
      a=u=i;
      b=v=j;
     }
  if(visited[u]==0 || visited[v]==0)
  {
   printf("\n Edge %d:(%d %d) cost:%d",ne++,a,b,min);
   mincost+=min;
   visited[b]=1;
  }
  cost[a][b]=cost[b][a]=999;
 }
 printf("\n Minimun cost=%d\n",mincost);

}
```

<u>Output</u>

```
mits@mits-Lenovo-S510:~/Desktop/s1mca$ gcc prims.c
mits@mits-Lenovo-S510:~/Desktop/s1mca$ ./a.out
Enter the number of nodes:5
ENTER THE COST OF 1 & 1 : 0
ENTER THE COST OF 1 & 2 : 2
ENTER THE COST OF 1 & 3 : 0
ENTER THE COST OF 1 & 4 : 0
ENTER THE COST OF 1 & 5 : 10
ENTER THE COST OF 2 & 1 : 2
ENTER THE COST OF 2 & 2 : 0
ENTER THE COST OF 2 & 3 : 3
ENTER THE COST OF 2 & 4 : 1
ENTER THE COST OF 2 & 5 : 0
ENTER THE COST OF 3 & 1 : 0
ENTER THE COST OF 3 & 2 : 3
ENTER THE COST OF 3 & 3 : 0
ENTER THE COST OF 3 & 4 : 7
ENTER THE COST OF 3 & 5 : 0
ENTER THE COST OF 4 & 1 : 0
ENTER THE COST OF 4 & 2 : 1
ENTER THE COST OF 4 & 3 : 7
ENTER THE COST OF 4 & 4 : 0
ENTER THE COST OF 4 & 5 : 4
ENTER THE COST OF 5 & 1 : 10
ENTER THE COST OF 5 & 2 : 0
ENTER THE COST OF 5 & 3 : 0
ENTER THE COST OF 5 & 4 : 4
ENTER THE COST OF 5 & 5 : 0


 Edge 1:(1 2) cost:2
 Edge 2:(2 4) cost:1
 Edge 3:(2 3) cost:3
 Edge 4:(4 5) cost:4
 Minimun cost=10
mits@mits-Lenovo-S510:~/Desktop/s1mca$
```

Experiment 20 Date: 21.12.2023

Kruskal's Algorithm

Aim

Program to implement kruskal's algorithm

Algorithm:

```
1.  Start
2.  Declare globally  i,j,k,a,b,u,v,ne=1,min,mincost=0,cost[9][9],parent[9]
```

Main()

```
1. Read no of vertices
2.for(i=1;i<=n;i++)
3.  for(j=1;j<=n;j++)
      cost[i][j]
       if cost[i][j]==0
       cost[i][j]=999
4.while(ne<n)
        for(i=1,min=999;i<=n;i++)
           for(j=1;j<=n;j++)
                if(cost[i][j]<min)
                    min=cost[i][j];
                     a=u=i;
                     b=v=j;
5.  u=find(u)
6.  v=find(v)
7.  if(uni(u,v))
       Display edge
       mincost +=min
   cost[a][b]=cost[b][a]=999
8. print Minimum cost
```

int find(int i)

```
1.  while(parent[i])
2.    i=parent[i]
3.   return i
```

int uni(int i,int j)

```
1.  if(i!=j)
2.  parent[j]=i
3.  return 1
```

Program

```
#include<stdio.h>
#include<stdlib.h>
int i,j,k,a,b,u,v,n,ne=1;
int min,mincost=0,cost[9][9],parent[9];
int find(int);
int uni(int,int);
void main(){
 printf("\n\tImplementation of Kruskal's algorithm\n");
 printf("\nEnter the no. of vertices:");
 scanf("%d",&n);
 printf("\nEnter the cost adjacency matrix\n");
 for(i=1;i<=n;i++){
  for(j=1;j<=n;j++){
   scanf("%d",&cost[i][j]);
   if(cost[i][j]==0)
    cost[i][j]=999; }
 }
 printf("\nThe edges of Minimum Cost Spanning Tree are\n");
 while(ne<n){
  for(i=1,min=999;i<=n;i++){
   for(j=1;j<=n;j++){
    if(cost[i][j]<min){
     min=cost[i][j];
     a=u=i;
     b=v=j;}
            }
            }
  u=find(u);
  v=find(v);
  if(uni(u,v)){
   printf("\n%d edge (%d,%d) =%d\n",ne++,a,b,min);
   mincost +=min; }
  cost[a][b]=cost[b][a]=999;
 }
 printf("\n\tMinimum cost = %d\n",mincost);}
int find(int i){
 while(parent[i])
  i=parent[i];
 return i;}
int uni(int i,int j){
            if(i!=j){
             parent[j]=i;
             return 1;}
 return 0;}
```

Output

```
mits@mits-Lenovo-S510:~/Desktop/s1mca$ gcc kruskals.c
mits@mits-Lenovo-S510:~/Desktop/s1mca$ ./a.out

  Implementation of Kruskal's algorithm

Enter the no. of vertices:5

Enter the cost adjacency matrix
0 0 3 0 0
0 0 10 4 0
3 10 0 2 6
0 4 2 0 1
0 0 6 1 0

The edges of Minimum Cost Spanning Tree are

1 edge (4,5) =1

2 edge (3,4) =2

3 edge (1,3) =3

4 edge (2,4) =4

  Minimum cost = 10
mits@mits-Lenovo-S510:~/Desktop/s1mca$
```

Experiment 21 Date: 04.01.2024

Disjoint set operations

Aim

Program to perform disjoint set operations create union and find.

Algorithm

main()

```
1. declare i
2. numElements-6
3. unionSets(0, 1);
4, unionSets(1, 2);
5. unionSets (3, 4).
6. unionSets (4.5);
7. unionSets (2, 4);
8. set i 0, i<numElements
9. Print find(i)
```

void initSets()

```
1.declare i
2. for (i=0; i<numElements; i++)
       sets[i].parent=i;
       sets[i].rank=0;
3.exit
```

int find(element)

```
1.if (sets[element].parent!=element)
       sets[element).parent=find(sets[element].parent)
2.return sets[element].parent;
3.exit
```

void union Sets(element1, element2)

```
1.declare set1=find(element1), set2=find(element2);
2.if (set1 != set2)
      if (sets[set1].rank>sets[set2].rank)
            sets[set2].parent=set
     else if (sets[set1].rank < sets[set2].rank)
           sets[set1].parent =set2
```

```
    else
        sets[set1].rank++
3.exit
```

void displaySets()

```
1.declare i;
  2.for (i=0; i<numElements; i++)
          Display  i
  3.for (i=0; i<numElements; i++)
        Display sets[i].parent
  4.for (i=0; i<numElements; i++)
        Display sets[i].rank
```

Program

```
#include <stdio.h>
#include <stdlib.h>
#define MAX_ELEMENTS 1000
typedef struct Set{
int parent;
int rank;
}Set;
Set sets[MAX_ELEMENTS];
int numElements;
void initSets() {
  int i;
  for (i=0; i<numElements; i++) {
  sets[i].parent=i;
  sets[i].rank=0;
  }
}
int find(int element) {
if (sets[element].parent!=element) {
sets[element].parent=find(sets[element].parent);
}
return sets[element].parent;
}
void unionSets(int element1, int element2)
{
    int set1=find(element1);
    int set2=find(element2);
   if (set1 != set2)
    {
      if (sets[set1].rank>sets[set2].rank){
          sets[set2].parent=set1;
       }
     else if(sets[set1].rank < sets[set2].rank)
      {
        sets[set1].parent =set2;
      }
     else {
         sets[set2].parent =set1;
         sets[set1].rank++;
      }
   }
}
void displaySets() {
     int i;
     printf("\nElement:\t");
  for (i=0; i<numElements; i++)
   {
       printf("%d\t",i);
   }
printf("\nParent:\t");
for (i=0; i<numElements; i++) {
   printf("%d\t", sets[i].parent);
}
printf("\nRank:\t");
for (i=0; i<numElements; i++) {
printf("%d\t", sets[i].rank);
}
printf("\n\n");
}
int main(){
int i;
numElements = 6;
initSets();
displaySets();
unionSets(0, 1);
unionSets(1, 2);
unionSets (3, 4);
unionSets (4, 5);
unionSets (2, 4);
displaySets();
for (i=0; i<numElements; i++) {
    printf("%d",find(i));
}return 0;
}
```

Output

```
Element     0    1    2    3    4    5
Parent      0    1    2    3    4    5
Rank        0    0    0    0    0    0


Element     0    1    2    3    4    5
Parent      0    1    2    3    3    3
Rank        2    0    0    1    0    0


The representative element of element 0 is 0
The representative element of element 1 is 0
The representative element of element 2 is 0
The representative element of element 3 is 0
The representative element of element 4 is 0
The representative element of element 5 is 0
```

Experiment 22 Date: 05.01.2024

Dijkstras Algorithm

Aim:

Program for single source shortest path algorithm using Dijkstras algorithm

Algorithm

minDistance(int,bool)

1. Start
2. Set v=0,v<V
3. If(sptSet[v]==false&&dist[v]<=min)
4. Min=dist[v],min_index=v
5. Return min_index
6. Stop

printSolution(dist[])

1. Start
2. Set i=0,i<V
3. Print I,dist[i]
4. Stop

Dijkstra(graph[V][V],src)

1. Start
2. Declare dist[V]
3. Declare sptSet[V]
4. Declare(i=0;i<V;i++)
5. Set dist[i]=INT_MAX,sptSet[i]=false
6. Set dist[src]=0
7. Set count=0,count<V-1
8. Set u=minDistance(dist,sptSet)
9. Set sptSet[u]=true
10. Set v=0,v<V
11. If !sptSet[v]&&graph[u][v]
12. Set dist[u]!=INT_MAX
13. Set dist[u]+graph[u][v]<dist[v]
14. Set dist[v]=dist[u]+graph[u][v]
15. Print solution(dist)
16. Stop

Program:

```
#include <limits.h>
#include <stdbool.h>
#include <stdio.h>
#define V 9
int minDistance(int dist[], bool sptSet[]){
int min INT MAX, min_index;
for (int v = 0; v <V; v++)
if (sptSet[v] false && dist[v] <= min)
min dist[v], min_index = v; return min index;
void printSolution(int dist[]){
printf("Vertex \t\t Distance from Source\n");
for (int i=0; i<V; i++)
printf("%d \t\t\t\t %d\n", i, dist[i]);
}
void dijkstra(int graph[V][V], int src){
int dist[V];
bool sptSet[V];
for (int i=0; i<V; i++)
dist[i] = INT_MAX, sptSet[i] = false;
dist[src] = 0;
for (int count = 0; count <V-1; count++) {
int u minDistance(dist, sptSet); sptSet[u] = true; for (int v = 0; v <V; v++)
if (!sptSet[v] && graph[u][v] && dist[u] != INT_MAX && dist[u] + graph[u][v] < dist[v])
dist[v] dist[u]+ graph[u][v];
printSolution(dist);
}
int main(){
int graph[V][V]= {(0,4,0,0,0,0,0,8, 0),
(4,0,8, 0, 0, 0, 0, 11,0),
( 0, 8, 0, 7, 0, 4, 0, 0, 2),
(0, 0, 7, 0, 9, 14,0,0,0)
(0, 0, 0, 9, 0, 10, 0, 0,0,0}, 0, 0)
(0, 0, 4, 14, 10, 0, 2, 0, 0 },
(0, 0, 0, 0, 0, 2, 0, 1,6),
{ 8, 11, 0, 0, 0, 0, 1, 1,0,7), 0, 7),
{ 0, 0, 2, 0, 0, 0, 6, 7, 0 } };
dijkstra(graph, 0);
return 0;
}
```

Output

```
mits@mits-Lenovo-S510:~/Desktop/s1mca$ gcc dijkstras.c
mits@mits-Lenovo-S510:~/Desktop/s1mca$ gcc ./a.out
Vertex          Distance from Source

0                0

1                4

2                12

3                19

4                21

5                11

6                 9

7                 8

8                 14
```