

LAB 9

1) From a given vertex in a weighted connected graph, find shortest paths to other vertices using dijkstra's algorithm.

CODE:

```
#include<stdio.h>
#include<conio.h>
#define INFINITY 999
#define MAX 10

void dijkstra(int G[MAX][MAX],int n,int startnode);
int main()
{
    int G[MAX][MAX],i,j,n,u; printf("Enter
    no. of vertices:"); scanf("%d",&n);
    printf("\nEnter the adjacency
    matrix:\n"); for(i=0;i<n;i++)
    for(j=0;j<n;j++) scanf("%d",&G[i][j]);
    printf("\nEnter the starting node:");
    scanf("%d",&u); dijkstra(G,n,u); return
    0;
}
void dijkstra(int G[MAX][MAX],int n,int startnode)
{
    int cost[MAX][MAX],distance[MAX],pred[MAX];
    int
    visited[MAX],count,mindistance,nextnode,i,j;
    for(i=0;i<n;i++) for(j=0;j<n;j++) if(G[i][j]==0)
    cost[i][j]=INFINITY;
    else
    cost[i][j]=G[i][j];
    for(i=0;i<n;i++)
```

```

{
    distance[i]=cost[startnode][i]
    ; pred[i]=startnode;
    visited[i]=0;
}
distance[startnode]=0;
visited[startnode]=1;
count=1; while(count<n-
1)
{
    mindistance=INFINITY;
    for(i=0;i<n;i++)
    if(distance[i]<mindistance&&!visited[i]
    )
    {
        mindistance=distance[i];
        nextnode=i;
    }
    visited[nextnode]=1;
    for(i=0;i<n;i++)
    if(!visited[i])
    if(mindistance+cost[nextnode][i]<distance[i])
    {
        distance[i]=mindistance+cost[nextnode][i];
        pred[i]=nextnode;
    }
    count++;
}

for(i=0;i<n;i++)
if(i!=startnode)
{
    printf("\nDistance of node%d=%d",i,distance[i]); printf("\nPath=%d",i);
    j=i;
    do

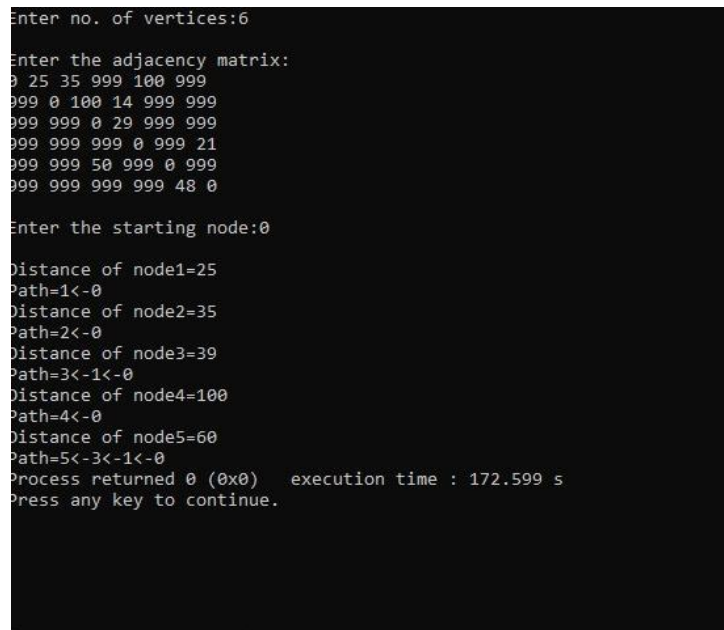
```

```

    { j=pred[j];
      printf("<-%d",j);
    }
    while(j!=startnode);
  }
}

```

OUTPUT:



```

Enter no. of vertices:6
Enter the adjacency matrix:
0 25 35 999 100 999
999 0 100 14 999 999
999 999 0 29 999 999
999 999 999 0 999 21
999 999 50 999 0 999
999 999 999 999 48 0

Enter the starting node:0

Distance of node1=25
Path=1<-0
Distance of node2=35
Path=2<-0
Distance of node3=39
Path=3<-1<-0
Distance of node4=100
Path=4<-0
Distance of node5=60
Path=5<-3<-1<-0
Process returned 0 (0x0)   execution time : 172.599 s
Press any key to continue.

```

2)Implement the “N-Queens” problem using Backtracking.

CODE:

```

#include<stdio.h>

#include<math.h>

int board[20],count;

int main()
{ int
n,i,j;
void queen(int row,int n);
printf("\n\nEnter no of Queens:");

```

```
scanf("%d",&n); queen(1,n);  
return 0;  
}
```

```
void print(int n)  
{ int  
i,j;  
printf("\n\nOutput %d:\n\n",++count);
```

```
for(i=1;i<=n;++i)  
    printf("\t%d",i);
```

```
for(i=1;i<=n;++i)  
{  
    printf("\n\n%d",i);  
    for(j=1;j<=n;++j)  
    {  
        if(board[i]==j)  
            printf("\tQ");  
        else  
            printf("\t-");  
    }  
}  
}
```

```
int place(int row,int column)  
{ int i;  
for(i=1;i<=row-1;++i)  
{  
    if(board[i]==column)  
        return 0;  
    else  
        if(abs(board[i]-column)==abs(i-row))  
            return 0;  
}  
}
```

```

return 1;
}

void queen(int row,int n)
{
int column;
for(column=1;column<=n;++column)
{
    if(place(row,column))
    {
        board[row]=column;
        if(row==n)
            print(n);
        else
            queen(row+1,n);
    }
}
}

```

OUTPUT:

```

Enter no of Queens:4

Output 1:
      1      2      3      4
1      -      Q      -      -
2      -      -      -      Q
3      Q      -      -      -
4      -      -      Q      -

Output 2:
      1      2      3      4
1      -      -      Q      -
2      Q      -      -      -
3      -      -      -      Q
4      -      Q      -      -
Process returned 0 (0x0)   execution time : 3.031 s
Press any key to continue.

```