23 Dijkstra's Algorithm.

```
#include <stdio.h>
#include <conio.h>
#define infinity 9999
#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 ("\n Enter the adjacency Matrix:\n");
  for (i=0; i<n; i++);
  for (j=0; j<n; j++)
   Scanf ("%d", &G[i][j]);
  Printf (" Enter 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[i]: cost[startnode][i];
  pred[i] = startnode;
   visited[i]=0;
  }
distance[startnode]=0;
visited[startnode] = 1;
  count=1
while (count<n-1)
for (i=0; i<n; i++)
 if (i != startnode)
{
 Printf("\n Distance of node %d = %d",
        i, distance[i]);
 Printf("\n Path = %d", i);
 j=i;
 do
{
 j = Pred[j];
  Printf("<-%d", j);
} while (j != startnode);
}
```

Enter no of Vertices. - 6

	a	b	c	d	e	f
a	0	25	35	0	100	0
b	0	0	27	14	0	0
c	0	0	0	29	0	0
d	0	0	0	0	0	21
e	0	0	50	0	0	0
f	0	0	6	0	48	0

Enter the Starting Node. - 0

Dist of node 1 = 25
Path = 1<-0
Dist of node 2 = 35
Path = 2<-0
Dist of node 3 = 34
Path = 3<-1<-0
Dist of node 4 = 100.
Path = <=-0
Dist of node 5 = 60.
Path = 5<-3<-1<-0.

123 N-Queen Algorithm.

```
#include <stdio.h>
#include <math.h>

int board[20], count;
int main()
{
  int n, i, j;
void queen (int row, int n);

Printf ("- N Queen Problem Using Backtracking -");
Printf ("\n\n Enter no of queen:");
Scanf ("%d", &n);
  queen(1, n);
    return 0;
  }
void Print (int n)
{
  int i, j;
  Printf ("\n\n Solution %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");
        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 the no of queens - 4

Sol-1

	1	2	3	4
1	-	Q	-	-
2	-	-	-	Q
3	Q	-	-	-
4	-	-	Q	-

Sol-2

	1	2	3	4
1	-	-	Q	-
2	Q	-	-	-
3	-	-	-	Q
4	-	Q	-	-

9/8/23