ADA-LAB-8

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

b) Implement "N-Queens Problem" using Backtracking

CODE-

Djkstra's Algorithm-

```
#include <stdio.h>

#include <conio.h>

void dijkstras();

int c[10][10], n, src;

void printPath(int parent[], int node);

void main()

{

  int i, j;

  printf("\nEnter the no of vertices:\t");

  scanf("%d", &n);

  printf("\nEnter the cost matrix:\n");

  for (i = 1; i <= n; i++)

  {

    for (j = 1; j <= n; j++)

    {

      scanf("%d", &c[i][j]);

    }

  }

  printf("\nEnter the source node:\t");

  scanf("%d", &src);

  dijkstras();

  getch();

}

void dijkstras()

{

  int vis[10], dist[10], parent[10], u, j, count, min;
```

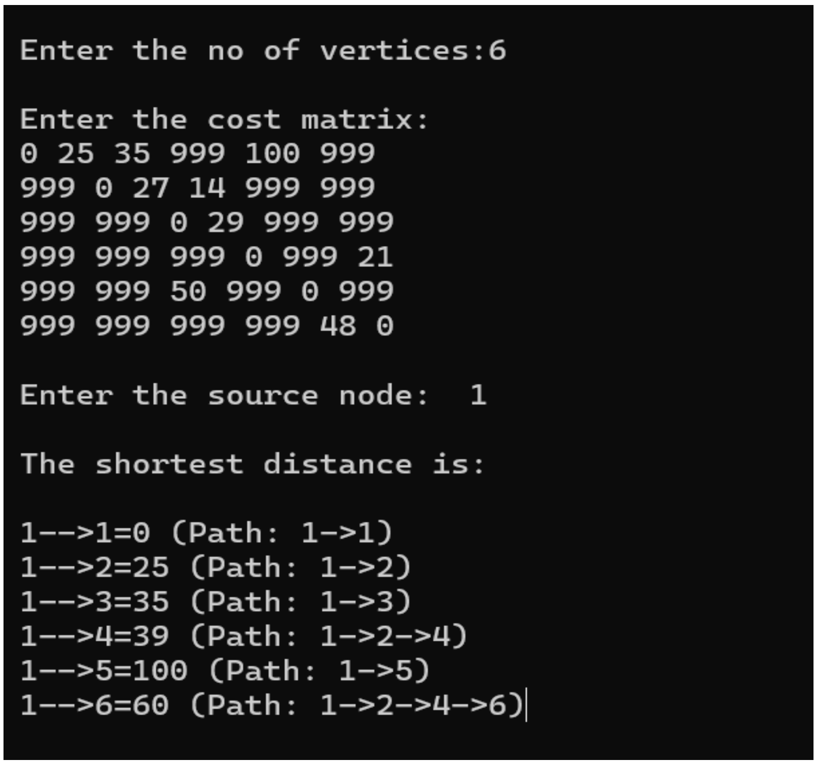
```
for (j = 1; j <= n; j++)
{
    dist[j] = c[src][j];
    parent[j] = src;
}
for (j = 1; j <= n; j++)
{
    vis[j] = 0;
}
dist[src] = 0;
vis[src] = 1;
count = 1;
while (count != n)
{
    min = 9999;
    for (j = 1; j <= n; j++)
    {
        if (dist[j] < min && vis[j] != 1)
        {
          min = dist[j];
            u = j;
        }
    }
    vis[u] = 1;
    count++;
    for (j = 1; j <= n; j++)
    {
        if (min + c[u][j] < dist[j] && vis[j] != 1)
        {
            dist[j] = min + c[u][j];
            parent[j] = u;
        }
        }

    }

    printf("\nThe shortest distance is:\n");

    for (j = 1; j <= n; j++)

    {

        printf("\n%d-->%d=%d (Path: %d", src, j, dist[j], src);

        printPath(parent, j);

        printf(")");

    }

}

void printPath(int parent[], int node)

{

    if (parent[node] == src)

    {

        printf("->%d", node);

        return;

    }

    printPath(parent, parent[node]);

    printf("->%d", node);

}
```

N-Queens-

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

int x[20]; // Solution array to store column index of queens
int count = 0;

int place(int k, int i) {
    for (int j = 1; j <= k - 1; j++) {
        if (x[j] == i || abs(x[j] - i) == abs(j - k)) {
            return 0;
        }
    }
    return 1;
}
void nqueens(int k, int n) {
    for (int i = 1; i <= n; i++) {
        if (place(k, i)) {
            x[k] = i;
            if (k == n) {
                count++;
                printf("Solution %d:\n", count);
                for (int j = 1; j <= n; j++) {
                    for (int l = 1; l <= n; l++) {
                        if (x[j] == l) {
                            printf("Q ");
                        } else {
                            printf("0 ");
                        }
                    }
                    printf("\n");
                }
                printf("\n");
            } else {
                nqueens(k + 1, n);
            }
        }
    }
}

int main() {
    int n;
    printf("Enter the number of queens: ");
    scanf("%d", &n);

    if (n <= 0) {
        printf("Invalid input.\n");
        return 1;
    }

    nqueens(1, n);

    if (count == 0) {
        printf("No solutions found for %d queens.\n", n);
    } else {
        printf("Total solutions: %d\n", count);
    }

    return 0;
}
```

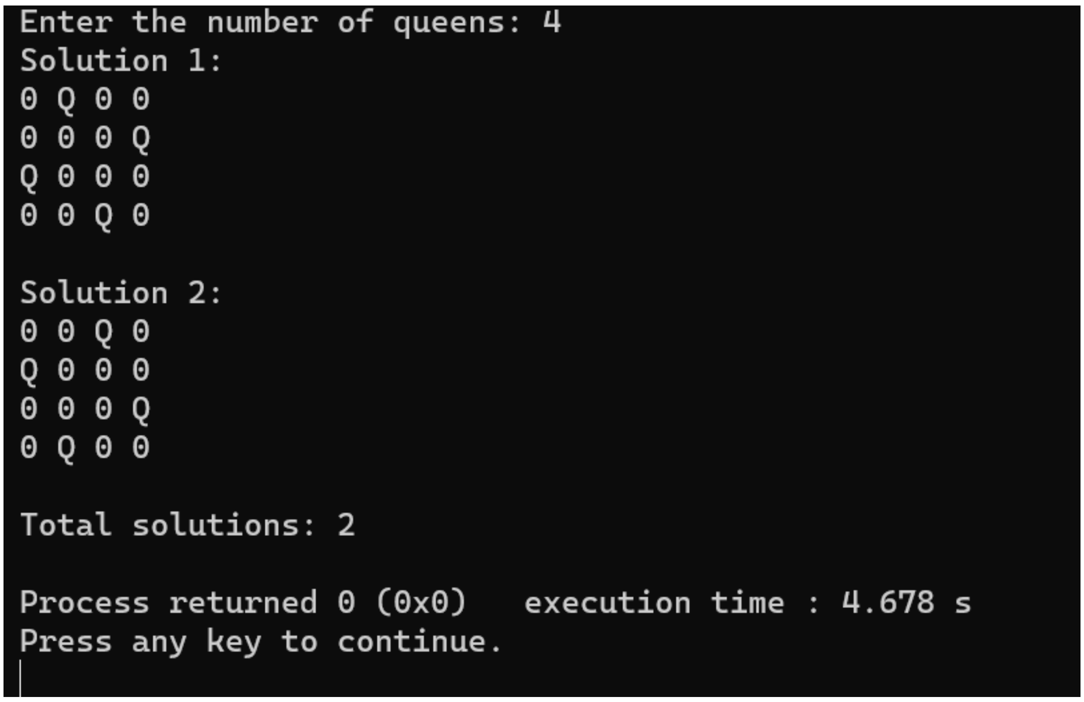
OUTPUT-

Djkstra’s Alogirthm-



```
Enter the no of vertices:6

Enter the cost matrix:
0 25 35 999 100 999
999 0 27 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 source node:  1

The shortest distance is:

1-->1=0 (Path: 1->1)
1-->2=25 (Path: 1->2)
1-->3=35 (Path: 1->3)
1-->4=39 (Path: 1->2->4)
1-->5=100 (Path: 1->5)
1-->6=60 (Path: 1->2->4->6)
```

N-Queens-



```
Enter the number of queens: 4
Solution 1:
0 Q 0 0
0 0 0 Q
Q 0 0 0
0 0 Q 0

Solution 2:
0 0 Q 0
Q 0 0 0
0 0 0 Q
0 Q 0 0

Total solutions: 2

Process returned 0 (0x0)   execution time : 4.678 s
Press any key to continue.
```