**Practical 5**

**AIM:** Implement Shortest Path Routing Algorithm (Dijkstra Algorithm).

**THEORY:**

Dijkstra's algorithm is an algorithm for finding the shortest paths between nodes in a graph, which may represent, for example, road networks. It was conceived by computer scientist Edsger W. Dijkstra in 1956 and published three years later.

The algorithm exists in many variants. Dijkstra's original algorithm found the shortest path between two given nodes, but a more common variant fixes a single node as the "source" node and finds shortest paths from the source to all other nodes in the graph, producing a shortest-path tree. For a given source node in the graph, the algorithm finds the shortest path between that node and every other. It can also be used for finding the shortest paths from a single node to a single destination node by stopping the algorithm once the shortest path to the destination node has been determined. For example, if the nodes of the graph represent cities and edge path costs represent driving distances between pairs of cities connected by a direct road, Dijkstra's algorithm can be used to find the shortest route between one city and all other cities. A widely used application of shortest path algorithms is network routing protocols, most notably IS-IS (Intermediate System to Intermediate System) and Open Shortest Path First (OSPF).

The Dijkstra algorithm uses labels that are positive integers or real numbers, which are totally ordered. It can be generalized to use any labels that are partially ordered, provided the subsequent labels are monotonically non-decreasing. This generalization is called the generic Dijkstra shortest-path algorithm.

**CODE:**

close all;

matrix = [[0 7 7 9 Inf Inf Inf Inf],

[7 0 8 3 Inf Inf 1 Inf],

[7 8 0 Inf 1 Inf 4 Inf],

[9 3 Inf 0 Inf Inf Inf 6],

[Inf Inf 1 Inf 0 Inf 1 7],

[Inf Inf Inf Inf Inf 0 Inf 7],

[Inf 1 4 Inf 1 Inf 0 6],

[Inf Inf Inf 6 7 7 6 0]];

[path,cost]= dijkstra(matrix, 2);

display(path);

display(cost);

function [path,cost]= dijkstra(matrix, source)

nodes=length(matrix);

cost(1:nodes)=Inf;

known(1:nodes)=0;

path(1:nodes)=-1;

cost(source)=0;

curr=source;

while sum(known)<nodes

known(curr)=1;

for i=1:nodes

if (matrix(curr,i)~=Inf | matrix(curr,i)~=0) & (known(i)==0)

if(cost(i)>cost(curr)+matrix(curr,i))

cost(i)=matrix(curr,i)+cost(curr);

path(i)=curr;

end

end

end

minValue=Inf; %minimum cost from non visited nodes

minVertex=Inf; %vertex with minimum cost

for i=1:nodes

if known(i)==0

if cost(i)<minValue

minValue=cost(i);

minVertex=i;

end

end

end

curr=minVertex;

end

end

**GRAPH:**

**Diagram

Description automatically generated**

**RESULT:**

Calendar

Description automatically generated with medium confidence

**CONCLUSION:** Thus, we implemented Dijkstra Algorithm to find the shortest path with source vertex 2, given 7 node undirected graph. The path and minimum cost to reach respective nodes can be found using this algorithm.

**Practical 6**

**AIM:** Implement Symmetric Key Ciphering and Deciphering using Classical Ciphers.

**CODE:**

Text

Description automatically generated

close all;

clear all;

file=fileread('test.txt');

n=strlength(file);

for i=1:1:n

encript(i)=bitxor(int32(file(i)),i); %Xoring to encript data

end

encripted\_data=char(encript);

display(encripted\_data);

n=strlength(encripted\_data);

for i=1:1:n

decript(i)=bitxor(int32(encripted\_data(i)),i); %Xoring again to decript data

end

decripted\_data=char(decript);

display(decripted\_data);

**RESULT:**

**Graphical user interface, text, application, email

Description automatically generated**

**Practical 7**

**AIM:** Implement Asymmetric Key Ciphering and Deciphering using Modern Ciphers

**CODE:**

close all;

text = 'This is communication networks practical';

p = 61;

q = 53;

n = p\*q;

phi = lcm((p-1),(q-1));

e=7;

d=1;

while 1

if mod(e\*d, phi) ==1

break

end

d=d+1;

end

text = double(text);

encrypt=powermod(text,e,n);

encrypted\_data = char(encrypt);

display(encrypted\_data);

t=double(encrypted\_data);

decrypt=powermod(t,d,n);

decrypted\_data=char(decrypt);

display(decrypted\_data);

**RESULT:**

Graphical user interface, text, application, email

Description automatically generated