NAME	Yash Burad
UID	2022301004
BATCH	A
SUBJECT	DAA
EXPERIMENT NO	6
AIM	To understand and implement Dijkstra’s Algorithm to find shortest path
THEORY	Dijkstra's algorithm is an algorithm for finding the shortest paths between nodes in a weighted 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 costs of edge paths represent driving distances between pairs of cities connected by a direct road (for simplicity, ignore red lights, stop signs, toll roads and other obstructions), then 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

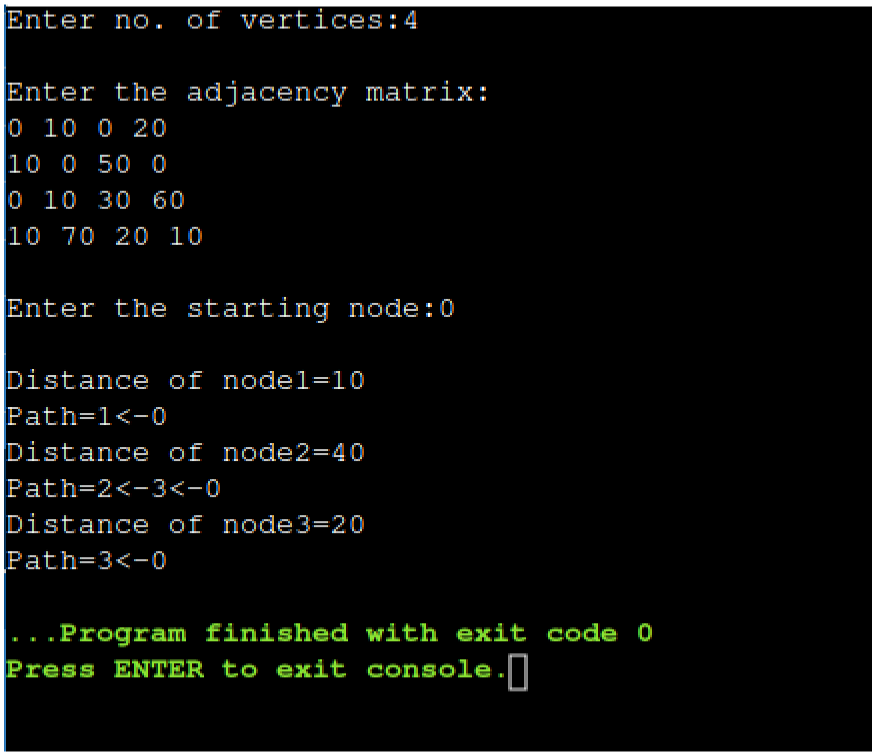
	protocols Below are the basic steps of how Dijkstra’s algorithm works: • So Basically, Dijkstra’s algorithm starts at the node source node we choose and then it analyzes the graph condition and its paths to find the optimal shortest distance between the given node and all other nodes in the graph. • Dijkstra’s algorithm keeps track of the currently known shortest distance from each node to the source node and updates the value after it finds the optimal path once the algorithm finds the shortest path between the source node and destination node then the specific node is marked as visited.
ALGORITHM	DIJKSTRA(G,s) INITIALIZE-SINGLE-SOURCE(G, S) 2 S ← Ø 3 Q ← V[G] 4 while Q ≠ Ø 5 do u ← EXTRACT-MIN(Q) 6 S ← S U {u} 7 for each vertex v ∈ Adj[u] 8 do if dist[v] > dist[u] + w(u,v) 9 then d[v] ←d[u] + w(u,v) INITIALIZE-SINGLE-SOURCE(Graph g, Node s) dist[s] = 0; for each vertex v in Vertices V[G] - s dist[v] ← ∞

PROGRAM	
	```
	#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("\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;
	//pred[] stores the predecessor of each node
	//count gives the number of nodes seen so far
	//create the cost matrix
	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];
//initialize pred[],distance[] and visited[]
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;
//nextnode gives the node at minimum distance
for(i=0;i<n;i++)
if(distance[i]<mindistance&&!visited[i])
{
mindistance=distance[i];
nextnode=i;
}
//check if a better path exists through nextnode
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++;
}
```

```
//print the path and distance of each node
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);
}
}
```

Ouput:



```
Enter no. of vertices:4

Enter the adjacency matrix:
0 10 0 20
10 0 50 0
0 10 30 60
10 70 20 10

Enter the starting node:0

Distance of node1=10
Path=1<-0
Distance of node2=40
Path=2<-3<-0
Distance of node3=20
Path=3<-0

...Program finished with exit code 0
Press ENTER to exit console.
```

Observation: Dijkstra’s algorithm is used to find the shorest path from starting

node to last node, visiting every node. It works on the direced graph. It adds the distance of starting node with as well as cost of edge to get the distance of destination vertex, but the distance of destination vertex must be greater than the addition of starting vertex and edge cost. And the edge from where there is minimum distance that edge will be considered as short path. **Time complexity of Diskstra's algorithm:** O(n^2) where 1st n is number of vertices in the graph and another n is number of vertices relaxed where relaxation is updatation of vertices from infinity to the actual distance.	
Conclusion:	In this practical, I performed the dijkstra's algorithm program in c that finds the shortest distance between two vertices. It gives the optimal solution to visit the nodes in the graph. Its best application is Google maps