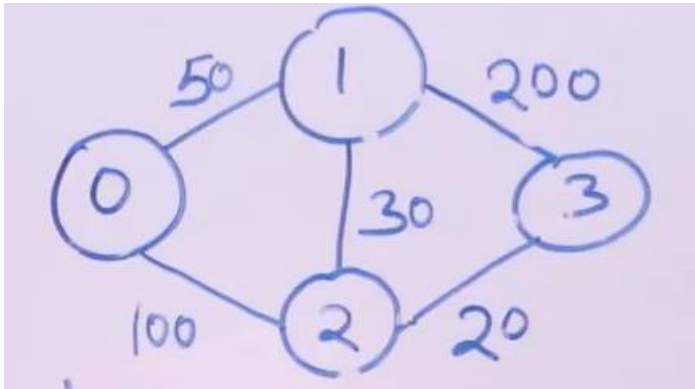


NAME:	GITANJALI GANGURDE
UID:	2021300034
CLASS:	SE-Comps A
BATCH:	B
EXPT NO:	6
6AIM:	Experiment based on greedy approach (Dijkstra's Algorithm)
THEORY:	<p>Dijkstra's algorithm allows us to find the shortest path between any two vertices of a graph.</p> <p>It differs from the minimum spanning tree because the shortest distance between two vertices might not include all the vertices of the graph.</p> <p>Dijkstra's Algorithm works on the basis that any subpath B \rightarrow D of the shortest path A \rightarrow D between vertices A and D is also the shortest path between vertices B and D.</p> <p>Dijkstra used this property in the opposite direction i.e we overestimate the distance of each vertex from the starting vertex. Then we visit each node and its neighbors to find the shortest subpath to those neighbors.</p> <p>The algorithm uses a greedy approach in the sense that we find the next best solution hoping that the end result is the best solution for the whole problem.</p> 

PROGRAM:

```
#include<stdio.h>
#include<stdlib.h>
#include<limits.h>
#include<stdbool.h>

int minDistance(int dist[], bool minSet[], int V)
{ int min = INT_MAX, min_index;

    for (int v = 0; v < V; v++) if (minSet[v]
        == false && dist[v] <= min) min =
        dist[v], min_index = v;

    return min_index;
}

void printSolution(int dist[], int V)
{ printf("Vertex\t\tDistance from Source \n");
    for (int i = 0; i < V; i++) printf("%d
        \t\t\t %d\n", i, dist[i]);
}

void dijkstra( int v, int graph[V][V], int source){
    int dist[V]; bool minSet[V]; for(int
    i=0;i<V;i++){ dist[i]=INT_MAX; minSet[i]=false;
    } dist[source]=0; for(int
    count=0;count<V-1;count++){ int u =
    minDistance(dist,minSet,v);
    minSet[u]=true; for(int i=0;i<V;i++){
    if(!minSet[i] && graph[u][i] &&
    dist[u]!=INT_MAX &&
        dist[u]+graph[u][i]<dist[i]){
        dist[i]=dist[u]+graph[u][i]; }
    }
}
```

	<pre> printSolution(dist,v); } int main(){ int v; printf("Enter the number of vertices: "); scanf("%d",&v); int graph[v][v]; printf("Enter the adjacency matrix: "); for(int i=0;i<v;i++){ for(int j=0;j<v;j++){ scanf("%d",&graph[i][j]); } } dijkstra(v,graph,0); return 0; } </pre>
--	---

RESULT:	<pre> PS C:\Users\dades\OneDrive\Documents\Programming\SEM4_C> cd "c:\Users\dades\OneDrive\Documents\Programming\SEM4_C" } ; if (\$?) { .\dijkstra } Enter the number of vertices: 4 Enter the adjacency matrix: 0 50 100 0 50 0 30 200 100 30 0 20 0 200 20 0 Vertex Distance from Source 0 0 1 50 2 80 3 100 PS C:\Users\dades\OneDrive\Documents\Programming\SEM4_C> </pre>
CONCLUSION:	<p>From this experiment, I learnt Dijkstra algorithm and understood how greedy approach is used. It helped us to find the shortest possible path between the source and other nodes.</p>