Name	Manav Rajesh Bhanushali
UID no.	2021700008
Experiment No.	6

AIM:	Greedy Approach – Single source shortest path
Program 1	
PROBLEM STATEMENT :	To find the shortest path using greedy approach – Dijkstra Algorithm.
ALGORITHM/ THEORY:	***It is an algorithm that is used for finding the shortest distance, or path, from starting node to target node in a weighted graph is known as Dijkstra's Algorithm.*** This algorithm makes a tree of the shortest path from the starting node, the source, to all other nodes (points) in the graph. Dijkstra's algorithm makes use of weights of the edges for finding the path that minimizes the total distance (weight) among the source node and all other nodes. This algorithm is also known as the single-source shortest path algorithm. Dijkstra's algorithm is the iterative algorithmic process to provide us with the shortest path from one specific starting node to all other nodes of a graph. **ALGORITHM :** • First step is to mark all nodes as unvisited, • Mark the picked starting node with a current distance of 0 and the rest nodes with infinity, • Now, fix the starting node as the current node, • For the current node, analyse all of its unvisited neighbours and measure their distances by adding the current distance of the current node to the weight of the edge that connects the neighbour node and current node, • Compare the recently measured distance with the current distance assigned to the neighbouring node and make it as the new current distance of the neighbouring node,

- After that, consider all of the unvisited neighbours of the current node, mark the current node as visited,
- If the destination node has been marked visited then stop, an algorithm has ended, and
- Else, choose the unvisited node that is marked with the least distance, fix it as the new current node, and repeat the process again from step 4.

APPLICATIONS-

- *For map applications*
- *For telephone networks*

ADVANTAGES -

- It can be used to calculate the shortest path between a single node to all other nodes and a single source node to a single destination node by stopping the algorithm once the shortest distance is achieved for the destination node.

DISADVANTAGES-

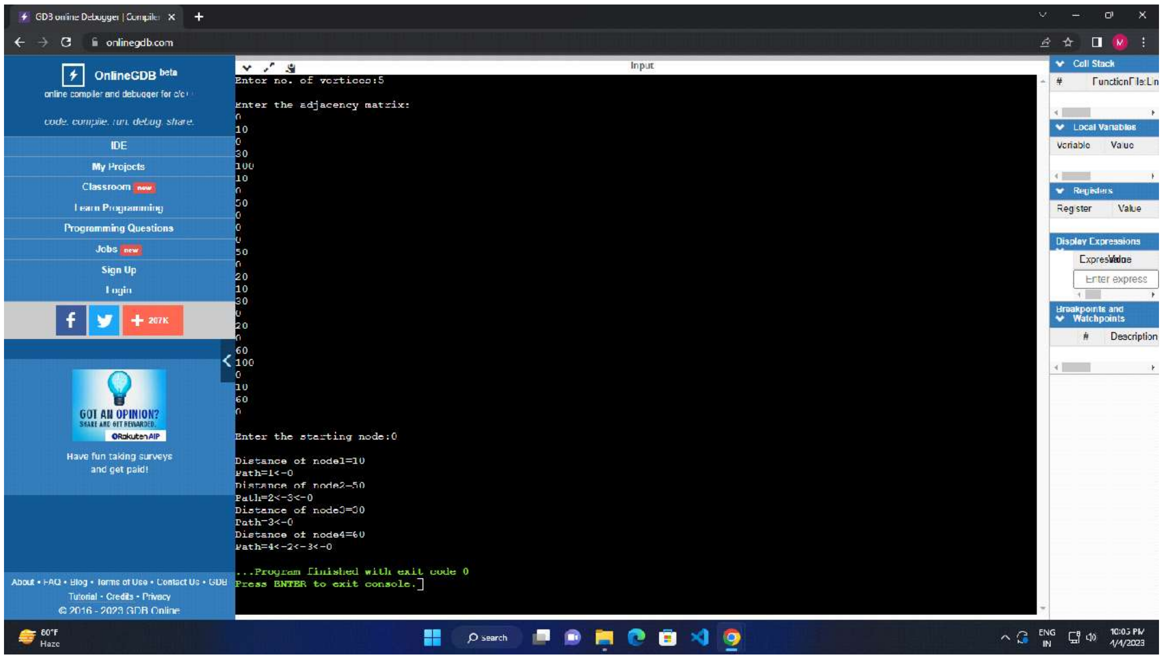
- It does an obscured exploration that consumes a lot of time while processing,
- It is unable to handle negative edges,
- As it heads to the acyclic graph, so can’t achieve the accurate shortest path, and
- Also, there is a need to maintain tracking of vertices, have been visited.

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;
   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[startnode]=0;
	visited[startnode]=1;
	count=1;
	while(count<n-1)
	{
	  mindistance=INFINITY;
	  for(i=0;i<n;i++)
	    if(distance[i]<mindistance&&!visited[i])
	    {
	      mindistance=distance[i];
	      nextnode=i;
	    }
	  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++;
	}


	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);
	  }
	}
	```

RESULT:



```
Enter no. of vertices:5

Enter the adjacency matrix:
0
10
0
30
100
10
0
50
0
0
0
50
0
20
10
30
0
20
0
60
100
0
10
60
0

Enter the starting node:0

Distance of node1=10
Path=1<-0
Distance of node2=50
Path=2<-3<-0
Distance of node3=30
Path=3<-0
Distance of node4=60
Path=4<-2<-3<-0

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

CONCLUSION:	I understood how to find the shortest path using greedy approach and how to use Dijkstra Algorithm.