**Design & Analysis of Algorithms Lab**

**CSEN2251**

**Dijkstra’s algorithm**

**Name:Subhashish Paul  
Roll:1951008**

**CSE A**

**Question:To Implement Dijkstra Algorithm using C,  
1. Take all the input through keyboard (number of vertices, graph, source vertex)  
2. Convert the entered graph into adjacency list for Cost Calculation (you may take the graph input directly as adjacency list to save time)  
3. Print the minimum cost from source to destination and also print the path between source and destination vertex**

**Dijkstra Algorithm Working code**

#include <stdio.h>

#include <stdlib.h>

#include <limits.h>

struct vertex

{

int vertex\_val;

int cost\_paid;

struct vertex \*next;

};

void Dijkstra\_Algorithm(struct vertex \*\*head,int source,int destination,int V);

int main()

{

int V,src,dstn,i,j,connected,check=0;

printf("Enter number of vertices and source vertex\n");

scanf("%d%d",&V,&src);

struct vertex \*\*adjacency\_list\_head=(struct vertex \*\*)malloc(V\*sizeof(struct vertex \*));

struct vertex \*temp,\*end;

for(i=0;i<V;i++)

{

printf("enter number of connected vertices to %d\n",i);

scanf("%d",&connected);

check=0;

printf("enter connected vertex and cost with vertex %d\n",i);

for(j=0;j<connected;j++)

{

temp=(struct vertex \*)malloc(sizeof(struct vertex));

temp->next=NULL;

scanf("%d %d",&temp->vertex\_val,&temp->cost\_paid);

if(check==0)

{

adjacency\_list\_head[i]=temp;

end=temp;

check=1;

}

else

{

end->next=temp;

end=temp;

}

}

}

printf("Enter destination vertex\n");

scanf("%d",&dstn);

Dijkstra\_Algorithm(adjacency\_list\_head,src,dstn,V);

return 0;

}

void Dijkstra\_Algorithm(struct vertex \*\*head,int source,int destination,int V)

{

struct vertex \*temp;

int i,j,min=INT\_MAX,save=0;

int \*\*arr=(int\*\*)malloc(3\*sizeof(int\*));

for(i=0;i<3;i++)

arr[i]=(int\*)malloc(V\*sizeof(int));

for(i=0;i<V;i++)

{

arr[0][i]=INT\_MAX; //i=0- distance

arr[1][i]=-1; //i=1- parent\_node (-1) no parent node yet

arr[2][i]=0; //i=2- visited 0=not blocked 1=blocked

}

arr[0][source]=0;

for(i=0;i<V;i++)

{

min=INT\_MAX;

for(j=0;j<V;j++)

{

if(arr[0][j]<min && arr[2][j]==0)

{

save=j;

min=arr[0][j];

}

}

j=save;

arr[2][j]=1;

temp=head[j];

while(temp!=NULL)

{

if((arr[0][save]+temp->cost\_paid) < arr[0][temp->vertex\_val] && arr[2][temp->vertex\_val]==0)

{

arr[0][temp->vertex\_val]=arr[0][save]+temp->cost\_paid;

arr[1][temp->vertex\_val]=save;

}

temp=temp->next;

}

}

printf("minimum cost from source to destination - %d\n",arr[0][destination]);

i=destination;

printf("parent vertex\n%d ",i);

while(i!=source)

{

printf("<- %d ",arr[1][i]);

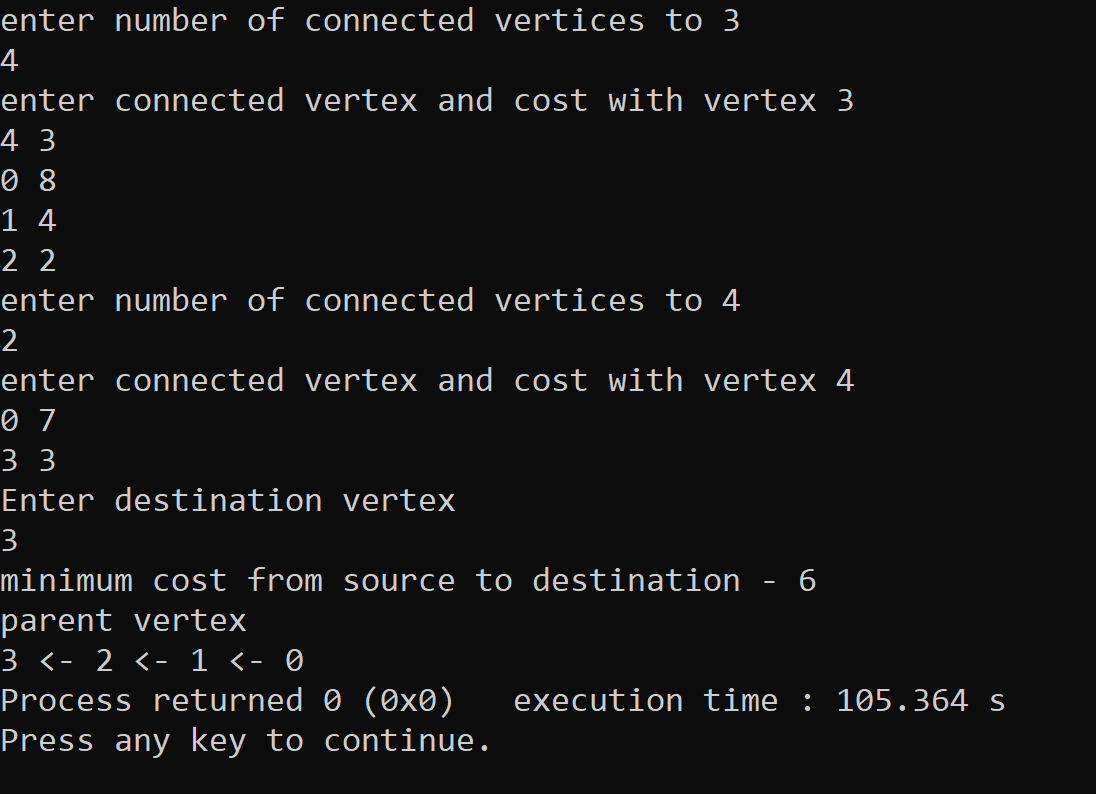
i=arr[1][i];

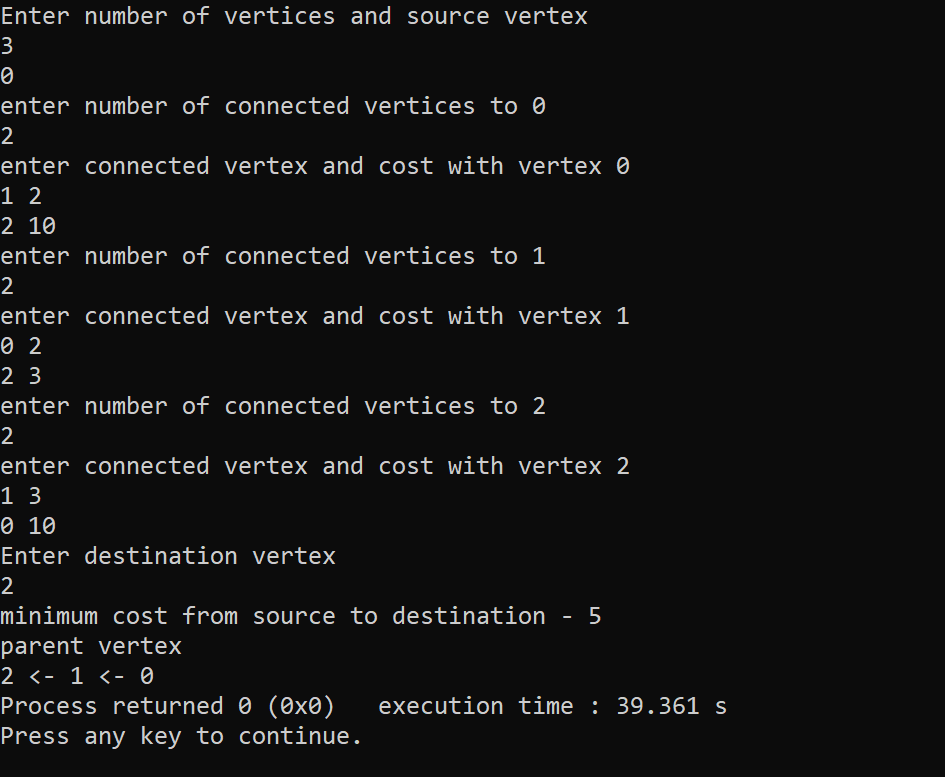
}

}

**Explanation:** This code directly takes input in adjacency list format then runs Dijkstra’s algorithm on it and prints the minimum cost to destination and path from source to destination.

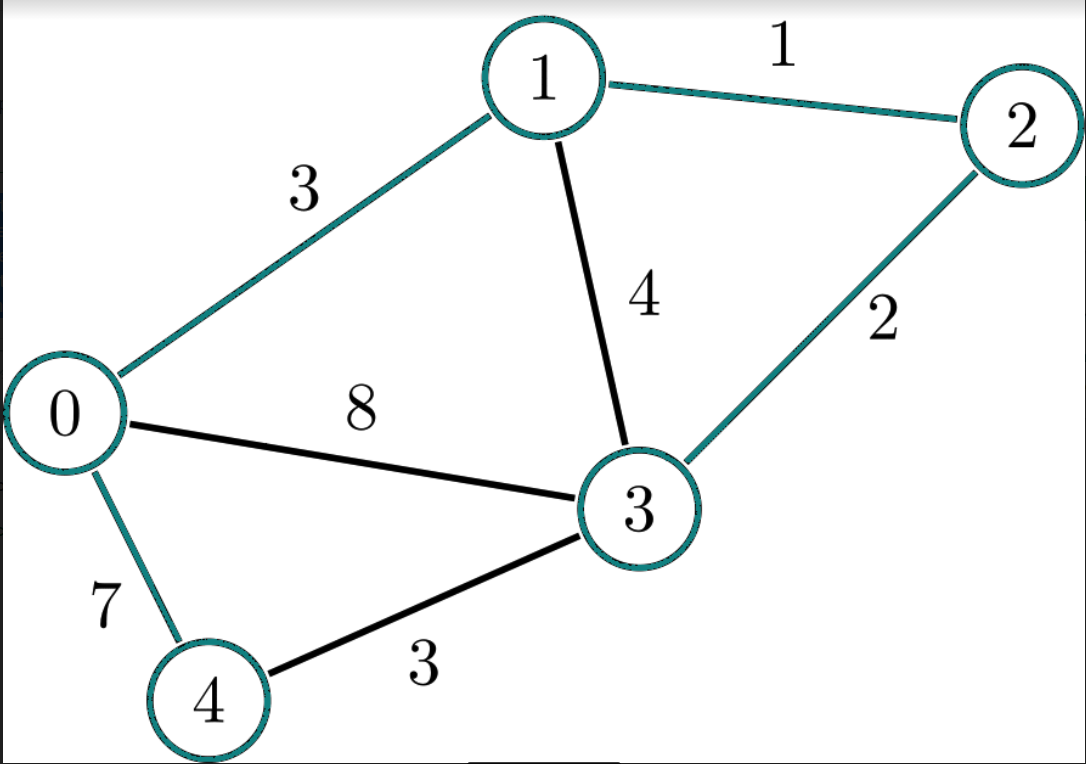
As dijkstra’s algorithm produces shortest path from source to all vertices this code can also produces minimum cost and path to all vertices from source.

Output:



Test case

1.



:Passed