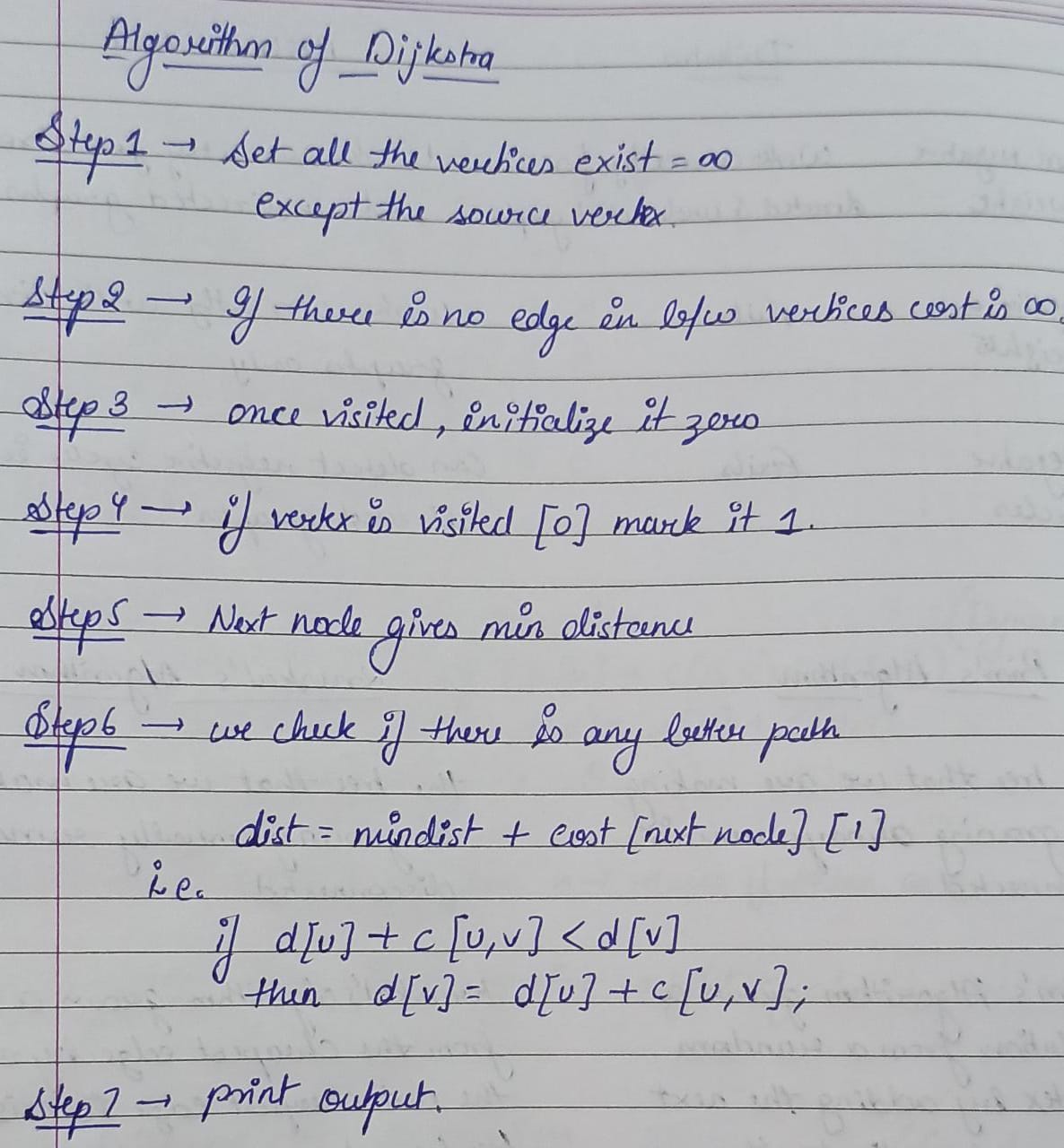
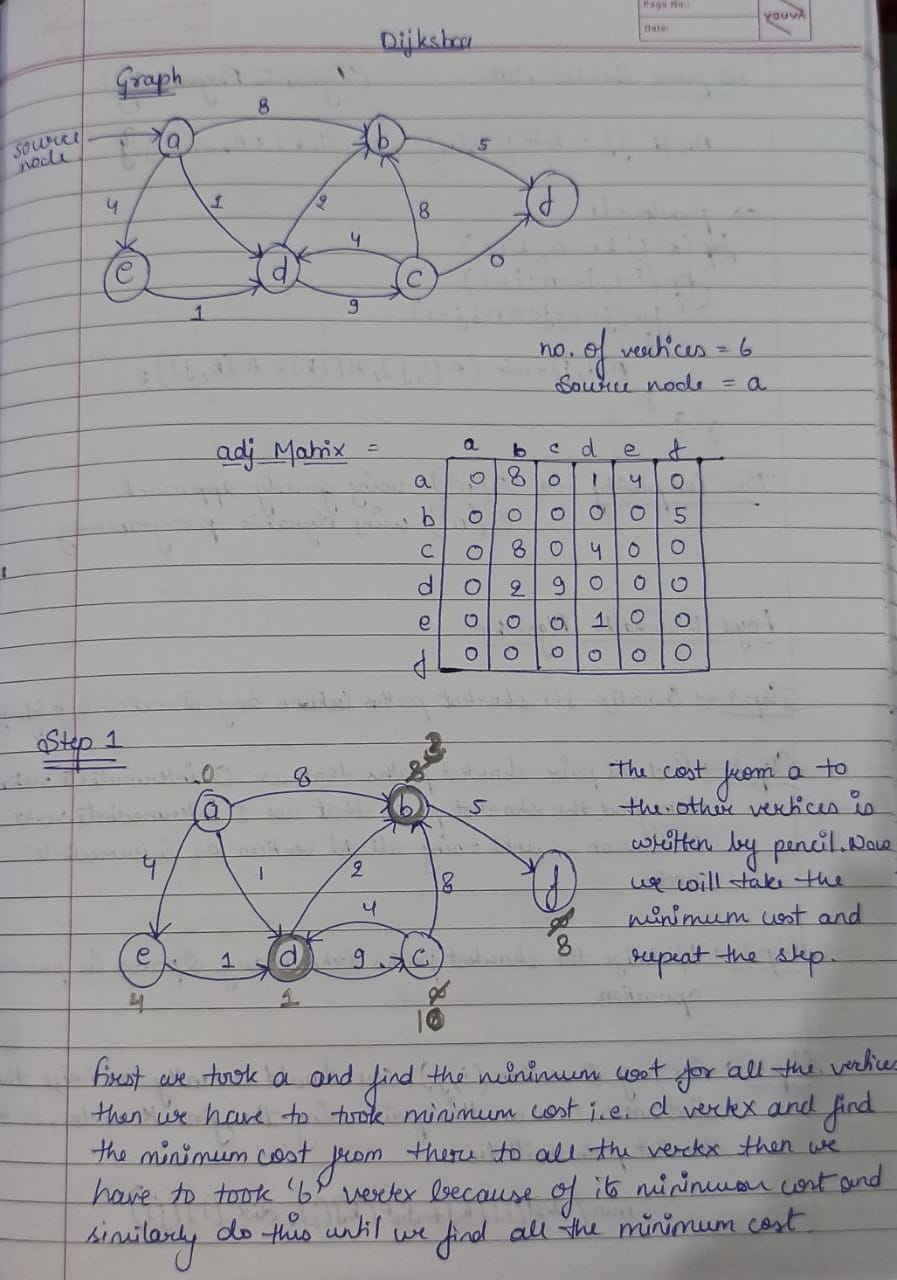
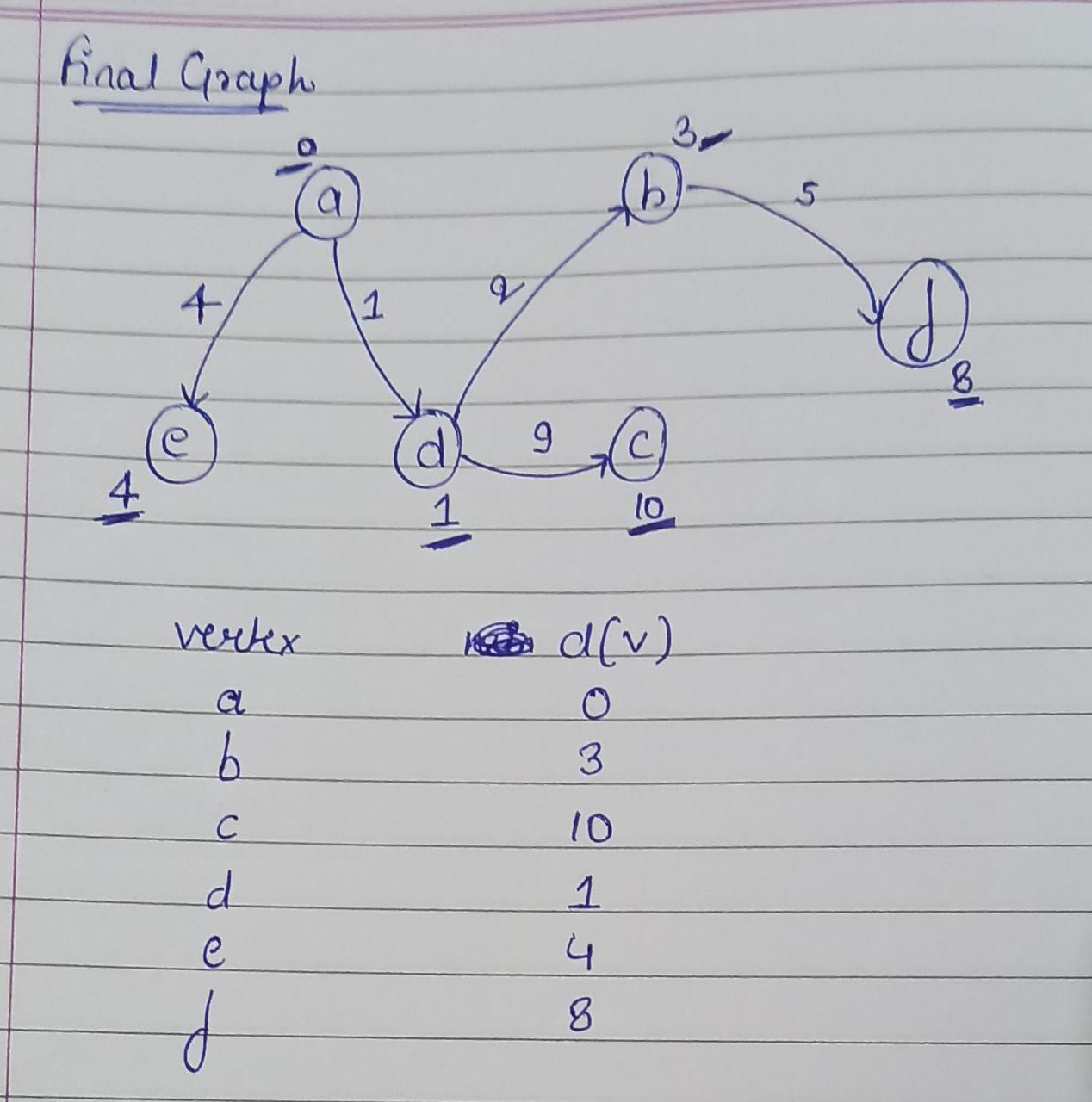
**DJIKSTRA’S ALGORITHM**

**ALGORITHM**

****  
**SOLVED  
**

****

**CODE**

#include<iostream>

using namespace std;

#define INFINITY 9999

#define max 10

void dijkstra(int G[max][max],int n,int startnode);

int main() {

int G[max][max]={{0,8,0,1,4,0},{0,0,0,0,0,5},{0,8,0,4,0,0},{0,2,9,0,0,0},{0,0,0,1,0,0},{0,0,0,0,0,0}};

int n=6; //number of vertices

int u=0; //source vertice

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) //no edge in btw vertices

cost[i][j]=INFINITY; //cost will be 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; //once visited,then it is initialized zero

}

distance[startnode]=0;

visited[startnode]=1; //if vertex is visited[0],is marked 1

count=1;

while(count<n-1)

{

mindistance=INFINITY;

for(i=0;i<n;i++) //next node gives the node at min distance

if(distance[i]<mindistance&&!visited[i]) {

mindistance=distance[i];

nextnode=i;

}

visited[nextnode]=1; //we check if there is any other better path through nextnode

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++) //printing path and distance of each node

if(i!=startnode) {

cout<<"\nDistance of node"<<i<<"="<<distance[i];

cout<<"\nPath="<<i;

j=i;

do {

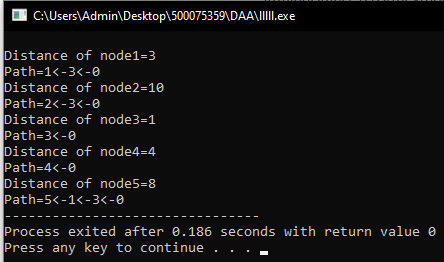
j=pred[j];

cout<<"<-"<<j;

}while(j!=startnode);

}

}

**OUTPUT  
**