**算法训练 Car的旅行路线**

时间限制：1.0s   内存限制：256.0MB

问题描述

　　又到暑假了，住在城市A的Car想和朋友一起去城市B旅游。她知道每个城市都有四个飞机场，分别位于一个矩形的四个顶点上，同一个城市中两个机场之间有一 条笔直的高速铁路，第I个城市中高速铁路了的单位里程价格为Ti，任意两个不同城市的机场之间均有航线，所有航线单位里程的价格均为t。  
　　那么Car应如何安排到城市B的路线才能尽可能的节省花费呢?她发现这并不是一个简单的问题，于是她来向你请教。  
　　找出一条从城市A到B的旅游路线，出发和到达城市中的机场可以任意选取，要求总的花费最少。

输入格式

　　的第一行有四个正整数s，t，A，B。  
　　S表示城市的个数，t表示飞机单位里程的价格，A，B分别为城市A，B的序号，(1<=A，B<=S)。  
　　接下来有S行，其中第I行均有7个正整数xi1，yi1，xi2，yi2，xi3，yi3，Ti，这当中的(xi1，yi1)，(xi2，yi2)，(xi3，yi3)分别是第I个城市中任意三个机场的坐标，T I为第I个城市高速铁路单位里程的价格。

输出格式

　　共有n行，每行一个数据对应测试数据，保留一位小数。

样例输入

1  
1 10 1 3  
1 1 1 3 3 1 30  
2 5 7 4 5 2 1  
8 6 8 8 11 6 3

样例输出

47.55

数据规模和约定

　　0<S<=100，

锦囊1

最短路。

锦囊2

先把所有的机场作为顶点构图，所有的公路和航线作为边。 然后用Dijkstra之类的算法求最短路。

本题的C++参考代码如下：

#include <float.h>

#include <math.h>

#include <stdio.h>

#include<cstring>

#define Sqr(x) ((x)\*(x))

double s[400][400], x[400], y[400];

int main()

{

int a, b, n, tt,t,i,j,k,N;

double res = DBL\_MAX;

scanf("%d%d%d%d", &n, &tt, &a, &b);

a-=1;

b-=1;

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

{

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

scanf("%lf%lf", &x[i\*4+j], &y[i\*4+j]);

scanf("%d", &t);

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

{

double x1 = x[i\*4+(j+1)%3], y1 = y[i\*4+(j+1)%3],

x2 = x[i\*4+(j+2)%3], y2 = y[i\*4+(j+2)%3];

if (fabs((x[i\*4+j]-x1)\*(x[i\*4+j]-x2)+(y[i\*4+j]-y1)\*(y[i\*4+j]-y2)) < 1E-7)

x[i\*4+3] = x1+x2-x[i\*4+j], y[i\*4+3] = y1+y2-y[i\*4+j];

}

for ( j = 0; j < 4; ++j)

for (k = 0; k < 4; ++k)

s[i\*4+j][i\*4+k] = t\*sqrt(Sqr(x[i\*4+j]-x[i\*4+k])+Sqr(y[i\*4+j]-y[i\*4+k]));

for ( j = 0; j < i\*4; ++j)

for ( k = 0; k < 4; ++k)

s[i\*4+k][j] = tt\*sqrt(Sqr(x[i\*4+k]-x[j])+Sqr(y[i\*4+k]-y[j])),

s[j][i\*4+k] = s[i\*4+k][j];

}

for ( k = 0; k < n\*4; ++k)

for (i = 0; i < n\*4; ++i)

for ( j = 0; j < n\*4; ++j)

if (s[i][k]+s[k][j] < s[i][j])

s[i][j] = s[i][k]+s[k][j];

for (i = 0; i < 4; ++i)

for ( j = 0; j < 4; ++j)

if (s[a\*4+i][b\*4+j] < res)

res = s[a\*4+i][b\*4+j];

printf("%.1lf\n", res);

return 0;

}

本题的C参考代码如下：

#include<stdio.h>

#include<math.h>

#include<stdlib.h>

#include<string.h>

#define BIG 999999.0

void Dijkstra(void);

float dist(float x1,float y1,float x2,float y2);

void construct(void);

typedef struct

{

float x,y;

}Point;

Point ct[100][4];

int rail[100];

int s,t,a,b;

float dis[100];

float g[400][400];

int main()

{

int i,j,k;

float answer=BIG;

scanf("%d%d%d%d",&s,&t,&a,&b);

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

scanf("%f %f %f %f %f %f %d ",&ct[i][0].x,&ct[i][0].y,&ct[i][1].x,&ct[i][1].y,

&ct[i][2].x,&ct[i][2].y,&rail[i]);

construct();//构图

Dijkstra();

for (i=0;i<4;i++){

if(dis[(b-1)\*4+i]<answer)

answer=dis[(b-1)\*4+i];

}

printf("%.1f ",answer);

system("pause");

return 0;

}

void construct(void)

{

int i,j,k,l;

for (i=0;i<s;i++){ //计算第四个点的坐标

if((ct[i][1].x-ct[i][0].x)\*(ct[i][2].x-ct[i][0].x)+

(ct[i][1].y-ct[i][0].y)\*(ct[i][2].y-ct[i][0].y)==0){

ct[i][3].x=ct[i][2].x+ct[i][1].x-ct[i][0].x;

ct[i][3].y=ct[i][2].y+ct[i][1].y-ct[i][0].y;

}else if((ct[i][0].x-ct[i][1].x)\*(ct[i][2].x-ct[i][1].x)

+(ct[i][0].y-ct[i][1].y)\*(ct[i][2].y-ct[i][1].y)==0){

ct[i][3].x=ct[i][2].x+ct[i][0].x-ct[i][1].x;

ct[i][3].y=ct[i][2].y+ct[i][0].y-ct[i][1].y;

}else if((ct[i][0].x-ct[i][2].x)\*(ct[i][1].x-ct[i][2].x)

+(ct[i][0].y-ct[i][2].y)\*(ct[i][1].y-ct[i][2].y)==0){

ct[i][3].x=ct[i][1].x+ct[i][0].x-ct[i][2].x;

ct[i][3].y=ct[i][1].y+ct[i][0].y-ct[i][2].y;

}

}

for(i=0;i<s;i++){//以城市为单位构图

for(j=0;j<4;j++)//城市内部

for(k=0;k<4;k++)

g[4\*i+j][4\*i+k]=rail[i]\*dist(ct[i][j].x,ct[i][j].y,ct[i][k].x,ct[i][k].y);

for(j=0;j<s;j++){//城市之间

if(j==i)continue;

for(k=0;k<4;k++)

for(l=0;l<4;l++)

g[4\*i+k][4\*j+l]=t\*dist(ct[i][k].x,ct[i][k].y,ct[j][l].x,ct[j][l].y);

}

}

}

float dist(float x1,float y1,float x2,float y2)

{

return sqrt((x2-x1)\*(x2-x1)+(y2-y1)\*(y2-y1));

}

void Dijkstra()

{

int i,j;

int min=BIG,minn;

int vis[100];

memset(vis,0,sizeof(vis));

for(i=0;i<s\*4;i++)

dis[i]=BIG;

dis[a-1]=dis[a]=dis[a+1]=dis[a+2]=0;

for(i=0;i<s\*4;i++){

min=BIG;

for(j=0;j<s\*4;j++){

if(!vis[j] && dis[j]<min){

min=dis[j];

minn=j;

}

}

vis[minn]=1;

for(j=0;j<4\*s;j++){

if(min+g[minn][j]<dis[j])

dis[j]=dis[minn]+g[minn][j];

}

}

}

本题的Java参考代码如下：

无