注：界面是使用xcode编写，语言为OC（iOS程序开发语言），因此只能运行于xocde，运行于iphone是需要支付99美金加入开发者计划的。所以制作了运行时的gif图片。

# C++代码

//

// main.cpp

// Short Path

//

// Created by 周子聪 on 16/4/16.

// Copyright © 2016年 独白SKY. All rights reserved.

//

#include <iostream>

struct MGraph

{

int arcs[30][30];

int vexnum;

};

MGraph initByArrage(int a[][30]){

MGraph mygraph;

mygraph.vexnum=30;

for (int i=0; i<30; i++) {

for (int j=0; j<30; j++) {

if (a[i][j]==0) {

mygraph.arcs[i][j]=INT32\_MAX;

}else

mygraph.arcs[i][j]=a[i][j];

}

}

return mygraph;

}

void ShortestPath\_DIJ(int v0,int end, bool p[][30],int D[],MGraph mygraph)

{

bool final[30];

for (int v=0; v<mygraph.vexnum; v++) {

final[v]=false;

D[v]=mygraph.arcs[v0][v];

for (int w=0; w<mygraph.vexnum; w++) {

p[v][w]=false;

}

if (D[v]<INT32\_MAX) {

p[v][v0]=p[v][v]=true;

}

}

D[v0]=0;

final[v0]=true;

for (int i=1; i<mygraph.vexnum; i++) {

int min=INT32\_MAX;

int v=0;

for (int w=0; w<mygraph.vexnum; w++) {

if (!final[w]&&D[w]<min) {

v=w;

min=D[w];

}

}

final[v]=true;

for (int w=0; w<mygraph.vexnum; w++) {

if (!final[w]&&min<INT32\_MAX&&mygraph.arcs[v][w]<INT32\_MAX&&(min+mygraph.arcs[v][w]<D[w])) {

D[w]=min+mygraph.arcs[v][w];

for (int j=0; j<mygraph.vexnum; j++) {

p[w][j]=p[v][j];

}

p[w][w]=true;

}

}

}

}

int main(int argc, const char \* argv[]) {

//校园地图的邻接矩阵

int A[30][30]={

0,100,60,0,0,0,0,0,0,0,0,0,0,0,0,100,0,0,0,0,0,0,0,0,200,0,0,0,0,0,

100,0,100,0,0,0,0,0,0,0,0,0,0,0,40,60,0,0,0,0,0,0,0,0,0,0,0,0,0,0,

60,100,0,60,20,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,160,0,0,0,0,0,

0,0,60,0,20,30,0,0,0,0,0,0,0,0,60,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,

0,0,20,20,0,50,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,200,170,0,0,0,0,0,

0,0,0,30,50,0,0,60,200,0,0,0,0,50,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,

0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,30,0,0,0,0,0,0,0,40,40,0,60,0,

0,0,0,0,0,60,0,0,150,30,70,0,0,50,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,

0,0,0,0,0,200,0,150,0,180,0,200,0,0,160,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,

0,0,0,0,0,0,0,30,180,0,0,0,0,0,0,0,50,20,0,0,0,0,0,0,0,0,0,0,0,0,

0,0,0,0,0,0,0,70,0,0,0,0,0,60,0,0,50,0,0,0,0,0,0,200,0,0,60,0,0,0,

0,0,0,0,0,0,0,0,200,0,0,0,60,0,0,0,0,260,100,0,0,0,0,0,0,0,0,0,0,0,

0,0,0,0,0,0,0,0,0,0,0,60,0,0,0,0,0,0,70,0,0,0,0,0,0,0,0,0,50,0,

0,0,0,0,0,50,0,50,0,0,60,0,0,0,0,0,0,0,0,0,0,0,0,200,0,0,0,0,0,0,

0,40,0,60,0,0,0,0,160,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,

100,60,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,

0,0,0,0,0,0,0,0,0,50,50,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,20,0,0,0,

0,0,0,0,0,0,30,0,0,20,0,260,0,0,0,0,0,0,0,0,0,0,0,0,0,0,70,0,0,0,

0,0,0,0,0,0,0,0,0,0,0,100,70,0,0,0,0,0,0,30,70,0,0,0,0,0,0,0,160,0,

0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,30,0,100,190,0,0,0,0,0,0,0,0,

0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,70,100,0,40,180,0,0,0,0,0,240,0,

0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,190,40,0,0,0,0,0,0,0,0,0,

0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,180,0,0,0,0,200,0,0,0,0,

0,0,0,0,200,0,0,0,0,0,200,0,0,200,0,0,0,0,0,0,0,0,0,0,40,0,0,80,0,0,

200,0,160,0,170,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,40,0,0,0,0,0,120,

0,0,0,0,0,0,40,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,200,0,0,0,50,0,140,0,

0,0,0,0,0,0,40,0,0,0,60,0,0,0,0,0,20,70,0,0,0,0,0,0,0,50,0,0,0,0,

0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,80,0,0,0,0,0,50,

0,0,0,0,0,0,60,0,0,0,0,0,50,0,0,0,0,0,160,0,240,0,0,0,0,140,0,0,0,0,

0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,120,0,0,50,0,0,

};

bool p[30][30];

int D[30];

int start=0;//设置起点

int end=11;//设置终点

MGraph dijkstra;

dijkstra=initByArrage(A);

ShortestPath\_DIJ(start, end, p, D, dijkstra);

for (int j=0; j<30; j++) {

std::cout<<p[end][j];

}

std::cout<<std::endl;

int point[30];

int distance[30];

int index=0;

point[0]=0;

for (int v=0; v<30; v++) {

if (p[end][v]==true) {

point[index]=v;

distance[index]=D[v];

index++;

}

}

//冒泡排序

int i, j, tempp,tempd;

for (j = 0; j < index - 1; j++)

for (i = 0; i < index - 1 - j; i++)

{

if(distance[i] > distance[i + 1])

{

tempp = point[i];

point[i] = point[i + 1];

point[i + 1] = tempp;

tempd = distance[i];

distance[i] = distance[i + 1];

distance[i + 1] = tempd;

}

}

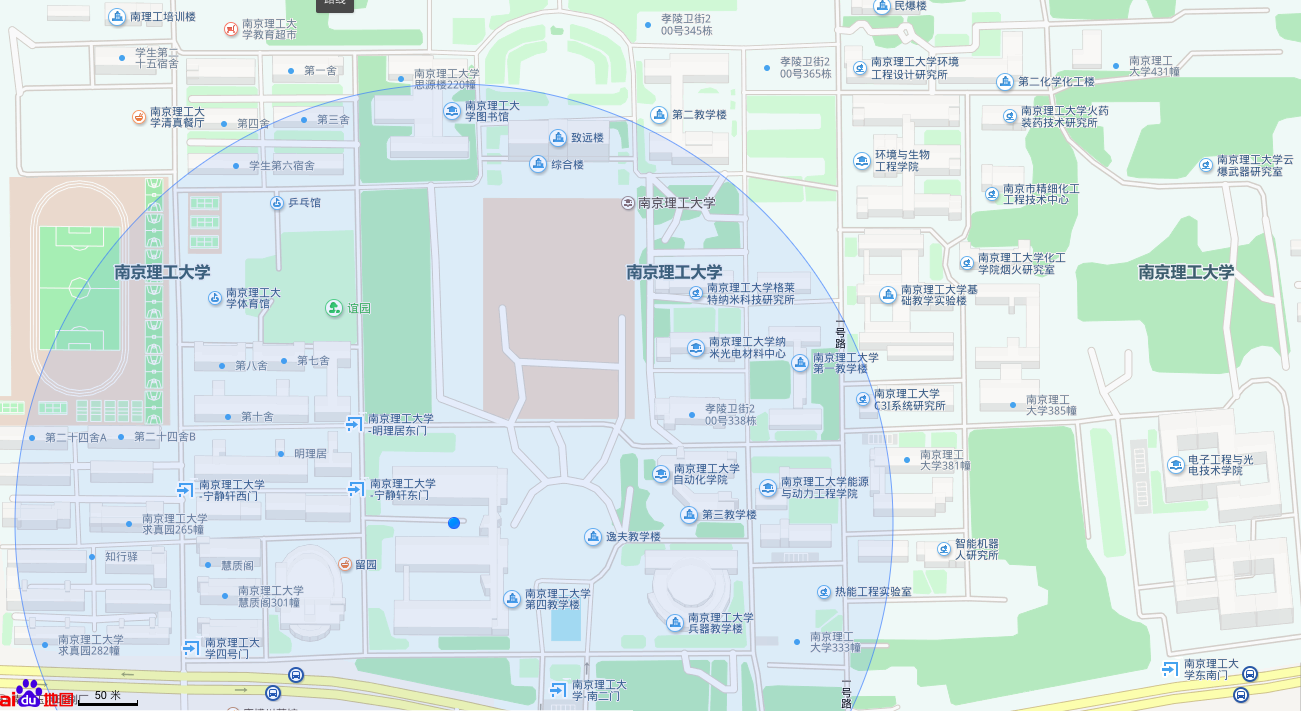
std::cout<<D[end]<<std::endl;

//总长度D[end] 路线point[] 所经过的点个数index

return 0;

}

# 校园部分地图线路及其长度设计图纸



PS：原来是打算从整个校园选取30个地点来制作的，但是感觉不妥，就选取了该地图靠左的30个点。编号0—29

/\* 0-29编号分别代表

0教育超市

1二十五舍

2一舍

3四舍

4三舍

5六舍

6明理居

7体育馆

8二运

9八舍

10谊园

11二十四舍A

12二十四舍B

13兵乓馆

14清真食堂

15培训楼

16七舍

17十舍

18求真园265栋

19知行驿

20智慧阁

21智慧阁301栋

22留园

23图书馆

24思源楼220栋

25宁静轩东门

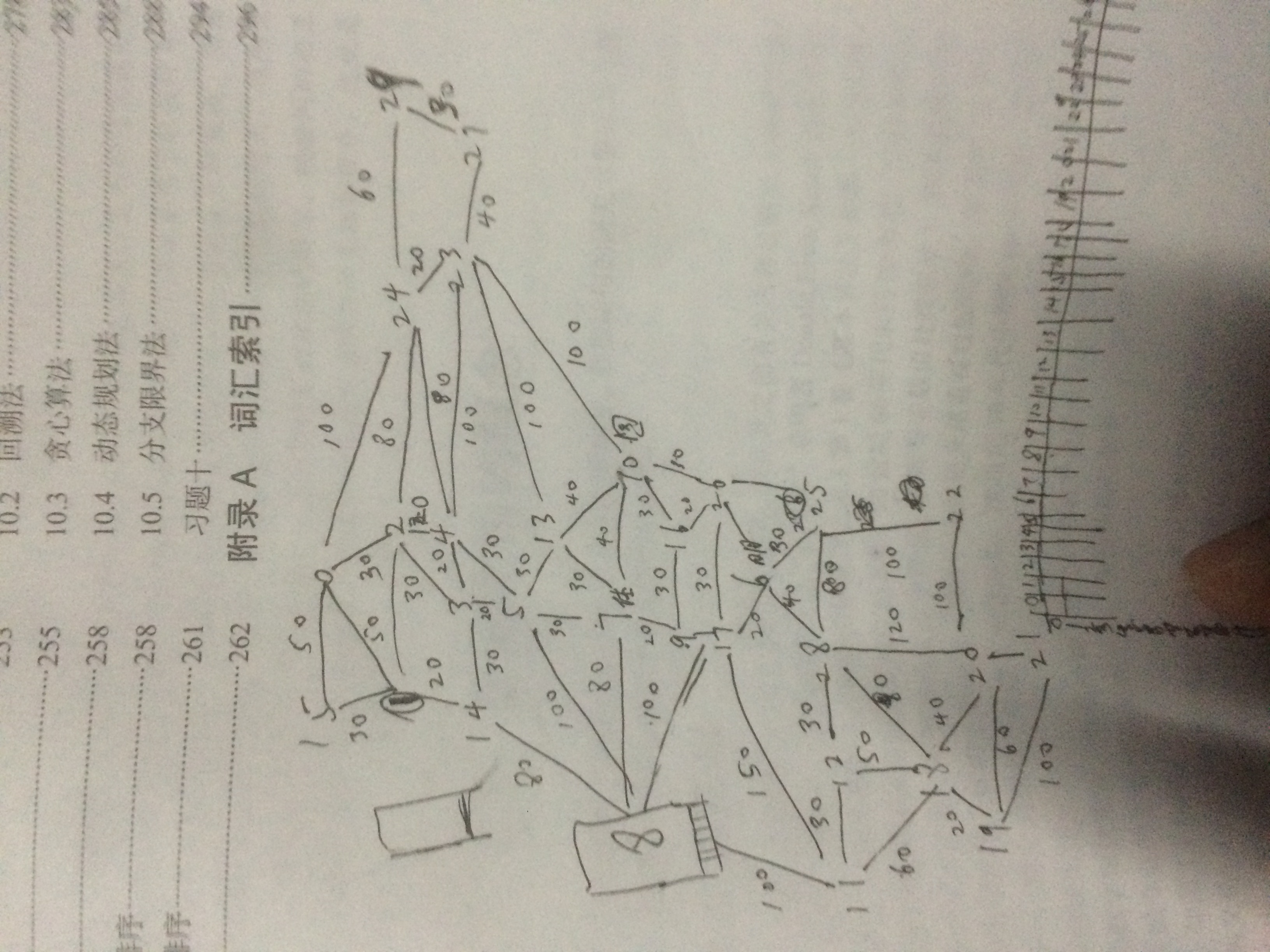
26明理居东门

27综合楼

28宁静轩西门

29致远楼

\*/



注：转换为邻接矩阵时感觉权值有所不妥，大部分值增加了一倍。

# 将C++代码移植到iOS加入界面后运行时示范图（GIF）

示范1：搜寻清真食堂（编号14）到留园（22）的最短路径

start=14；

end=22；

利用dijkstra算法得到

D[end]=470;

P[end][]={ 0,0,0,1,0,1,1,1,0,1,0,0,0,,0,1,0,0,1,0,0,0,1,0,0,1,0,0,0,0};

因此需要经过的点集为

point[]={3,5,6,7,9,14,17,22,25}

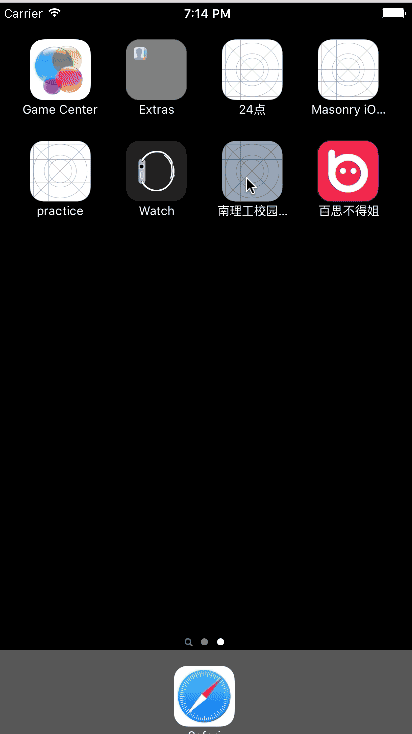
共9个点

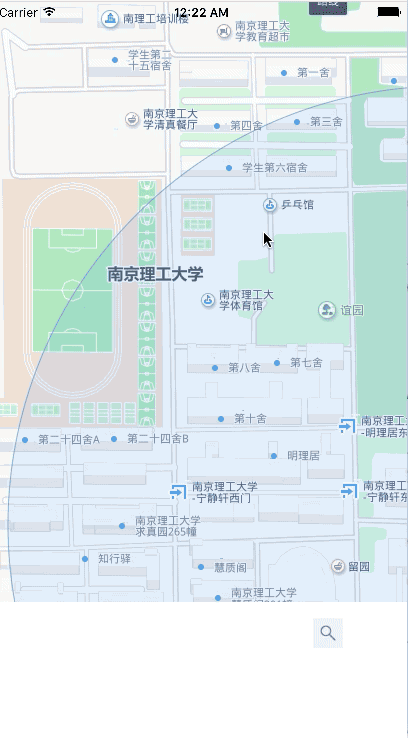
根据对应的D[]值对这9个数进行冒泡排序

得到有序点集

point[]={14,3,5,7,9,17,6,25,22};

最后通过绘图语句将路线用紫色线条依次连接。





注：制作这两个动态图的时候忘记将总路线长度D[end]输出了，现已加上。