**程序设计与算法综合训练**

**实验报告**

|  |  |
| --- | --- |
| **实验名称**： | 路径分析及应用 |
| **专业**： | 机器人工程 |
| **班级**： | 机器人工程三班 |
| **学号**： | WA2224013 |
| **姓名**： | 郭义月 |

**目录**

[**目录** 2](#_Toc184201931)

[一、 实验内容及要求 4](#_Toc184201932)

[1.1 实验目的 4](#_Toc184201933)

[1.2 实验内容 4](#_Toc184201934)

[1.3 实验要求 5](#_Toc184201935)

[1.4 实验任务 5](#_Toc184201936)

[二、 图的基本知识 5](#_Toc184201937)

[2.1 图的存储结构与基本知识 5](#_Toc184201938)

[2.2 最短路径算法 6](#_Toc184201939)

[2.2.1 Dijkstra算法 6](#_Toc184201940)

[2.2.2 Floyd算法 7](#_Toc184201941)

[三、 任务1：基于所给的旅游景点图、路径及权重（底层开发） 8](#_Toc184201942)

[3.1 任务描述 8](#_Toc184201943)

[3.2 重要部分代码： 8](#_Toc184201944)

[3.3 结果展示 9](#_Toc184201945)

[3.3.1 输入两个可到达的景点 9](#_Toc184201946)

[3.3.2 当输入的景点相同时 10](#_Toc184201947)

[3.3.3 当输入的景点不可直接到达时 10](#_Toc184201948)

[3.4 任务总结 10](#_Toc184201949)

[四、 任务2：ArcGIS最短路径分析—应用操作 11](#_Toc184201950)

[4.1 底图获取 11](#_Toc184201951)

[4.2 地理坐标获取 11](#_Toc184201952)

[4.3 按照老师发的步骤逐步完成 12](#_Toc184201953)

[五、 任务3基于ArcGIS API实现web端地图最短路径分析 12](#_Toc184201954)

[5.1 按照教程Html文件修改 12](#_Toc184201955)

[5.2 结果截图 13](#_Toc184201956)

[六、 实验总结与常见问题汇总 13](#_Toc184201957)

[6.1 实验总结 13](#_Toc184201958)

[6.2 常见问题汇总 14](#_Toc184201959)

[6.2.1 输出端那里不能输入cin 14](#_Toc184201960)

[6.2.2 输出端中文都是乱码 14](#_Toc184201961)

[6.2.3 arcmap没有license不能打开 14](#_Toc184201962)

[6.2.4 面板中找不到图形 15](#_Toc184201963)

# 实验内容及要求

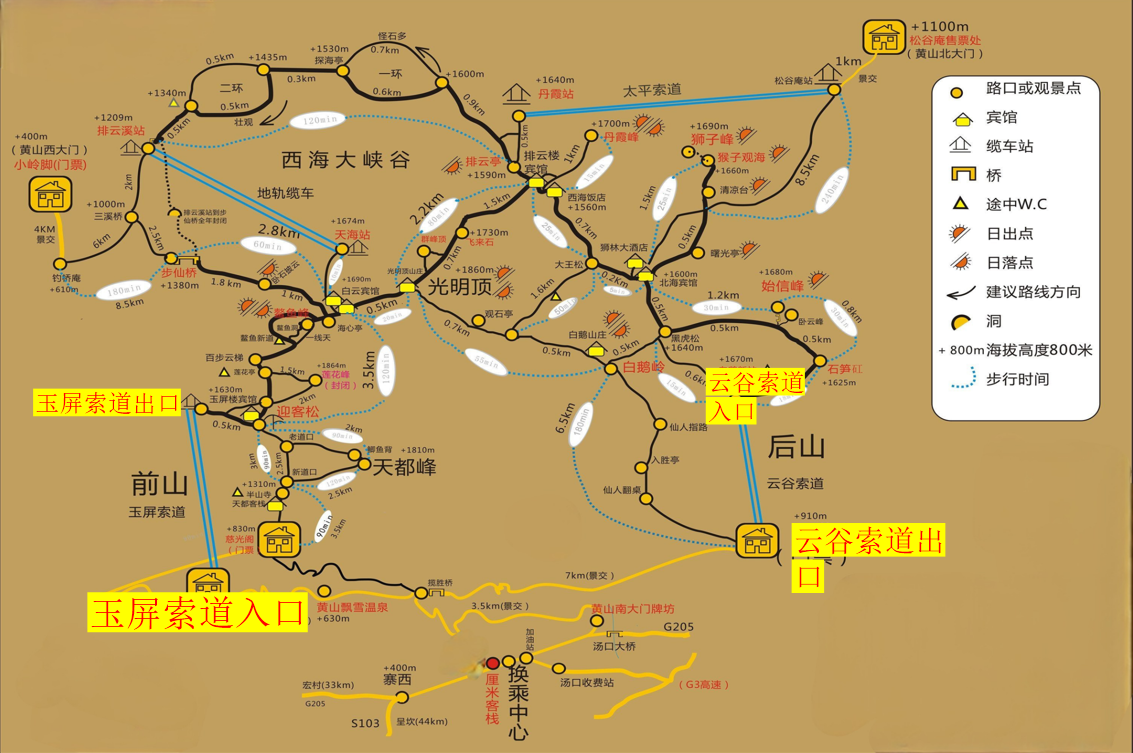
## 实验目的

加深对图状结构的理解，掌握最短路径的计算方法，强化学生的逻辑思维能力和动手能力。创建有向网表示的旅游景点导游图

了解Arcgis server的作用，学会调用Arcgis API来做一个网页并在网页上实现特定的功能，进而强化学生的逻辑思维能力和动手能力，巩固良好的编程习惯，为后续课程的学习打下坚实基础。通过Arcgis server进行地图发布，并在将自己发布的地图显现在自己编写的网页上，然后实现地图上两个点之间的最短路径分析，并将其标注出来。

## 实验内容

1. 旅游景点咨询系统的设计与实现



创建一个有向图表示的黄山旅游景点的导游图。

输入两个景点名，就可以得到其最短路径，如果两者无路径可通，就得出“两景点不可达的信息”。

1. ArcGIS软件完成最短路径分析
2. 安装Arcgis server，使用Arcgis server发布ArcGIS软件最短路径分析地图，并使用ArcGis API for JavaScript 显示地图、实现WEB端最短路径分析、路径显示。

## 实验要求

完成任务1：基于所给的旅游景点图、路径及权重--底层开发

完成任务2：ArcGIS最短路径分析—应用操作—2学时+2学时

完成任务3：基于ArcGIS API实现web端的地图最短路径分析

## 实验任务

**任务1：**基于所给的旅游景点图、路径及权重--底层开发

**任务2：**ArcGIS最短路径分析—应用操作

**任务3：**基于ArcGIS API实现web端的地图最短路径分析—二次开发

# 图的基本知识

## 图的存储结构与基本知识

//图的结构体

typedef struct {

char Vertex[MAXSIZE]; //顶点名称

int Matrix[MAXSIZE][MAXSIZE]; //邻接矩阵

int EdgeNum, VexNum; //边的个数和顶点的个数

}MGraph;

图是由一组顶点和一组边构成的数学结构，用于表示对象之间的关系。图广泛应用于计算机科学中的网络、路径优化、社交网络分析等领域。

顶点，表示图中的基本元素，表示实体；

边，连接两个顶点的线，表示实体之间的关系，边可以是有向的或者无向的。对于有向边，边有方向，表示从一个顶点到另一个顶点的关系；无向边，边没有方向，表示顶点之间的双向关系。

度是与顶点连接的边的数量，分为入度和出度，分别表示指向某个顶点的边的数量和从某个顶点指向其他顶点的边的数量。

路径是一系列顶点和边，表示从一个顶点到另一个顶点的路径。

回路是路径的起点和终点是同一个顶点，且路径上没有重复边或顶点。

连通性描述了图中顶点之间的连接情况，连通图表示任意两个顶点之间都有路径相连。

强连通图是有向图中对于任意两个顶点，都存在路径从一个顶点到另一个顶点。子图是图的部分结构，包含原图的一部分顶点和边。

图的分类包括：无向图，有向图，加权图和完全图。无向图的边没有方向，有向图的边有方向，加权图的边有权重，完全图是图中每一对顶点都有边相连。

图的表示方法有几种，邻接矩阵使用一个二维数组表示图的连接情况，矩阵中的元素表示顶点间是否有边。对于无向图，矩阵是对称的；对于有向图，矩阵中元素的行列表示边的方向。邻接表使用一个数组，其中每个数组元素都是一个链表，表示与该顶点相连的所有顶点。边集使用一个边的列表，每条边由一对顶点组成。

图的遍历方法包括深度优先搜索和广度优先搜索。深度优先搜索从起始顶点出发，尽可能深入到一个顶点的邻接节点，再回溯，常用栈来实现；广度优先搜索从起始顶点出发，逐层访问所有邻接顶点，常用队列来实现。

## 最短路径算法

### Dijkstra算法

Dijkstra算法是一种经典的解决单源最短路径问题的算法，适用于边权为非负数的图。其目的是从一个源节点出发，计算到所有其他节点的最短路径。

**初始化阶段：**首先，初始化一个距离数组 dist[]，其中 dist[source] = 0（表示源节点到自己的距离为0），其他所有节点的初始距离设为无穷大（∞）。此外，还需要一个已访问集合 visited[]，用于记录哪些节点的最短路径已经确定。源节点的距离已经确定，因此标记源节点为已访问。

**选择当前最短路径节点：**接下来，选择一个未被访问的节点，找到其距离源节点最近的节点。这个节点的最短路径已经被确定，并且它的最短路径是从源节点到该节点的当前已知最短路径。将这个节点标记为已访问。

**更新邻接节点的距离：**对于当前节点的每一个邻接节点，检查通过当前节点是否能得到更短的路径。如果通过当前节点到达邻接节点的路径比原来记录的路径更短，就更新邻接节点的最短路径。具体来说，如果 dist[current] + weight(current, neighbor) < dist[neighbor]，则更新 dist[neighbor]。

**标记当前节点为已访问：**完成邻接节点的距离更新后，将当前节点标记为已访问，表示该节点的最短路径已经确定，不再进行更新。

**重复步骤：**重复上述步骤，直到所有节点的最短路径都被确定。每次选择未访问节点中距离源节点最近的节点，更新其邻接节点的距离，并标记该节点为已访问。

**结束：**最终，距离数组 dist[] 存储了源节点到每个节点的最短路径长度。这个数组可以用于输出源节点到所有其他节点的最短路径。

Dijkstra算法的时间复杂度为O(V^2)，其中V是图中的节点数。如果使用优先队列（最小堆）进行优化，则时间复杂度为O((V + E) \* logV)，其中E为图中的边数。

### Floyd算法

Floyd算法（又叫Floyd-Warshall算法）用于解决多源最短路径问题，能够计算任意两节点之间的最短路径。与Dijkstra算法不同，Floyd算法通过遍历所有节点对并考虑每个节点作为中间节点来更新最短路径。

**初始化阶段：**首先，初始化一个二维数组 dist[][]，其中 dist[i][j] 表示从节点i到节点j的最短路径。如果节点i和节点j之间有直接的边，则 dist[i][j] 为该边的权值；如果没有直接的边，则 dist[i][j] 为无穷大（∞）；对角线上的元素 dist[i][i] 为0，表示节点到自身的路径长度为0。

**核心计算：**然后，算法会遍历每一个节点k，并考虑将其作为中间节点来更新路径。具体来说，对于每一对节点 (i, j)，如果通过节点k的路径比直接路径更短，即 dist[i][j] > dist[i][k] + dist[k][j]，则更新 dist[i][j] 为 dist[i][k] + dist[k][j]。这样，算法通过不断检查所有节点对，逐步更新最短路径。

**重复更新：**这一过程会对每一个节点k重复，从第1个节点到第N个节点（N是图中节点的总数）。每次遍历都会考虑一个新的节点k作为中间节点，更新所有可能的路径。

**结束：**最终，二维数组 dist[][] 存储了从任意节点i到任意节点j的最短路径长度。这个数组可以直接用于输出任意两节点之间的最短路径。

Floyd算法的时间复杂度为O(V^3)，其中V为图中的节点数。由于需要三重循环遍历所有的节点对，因此其时间复杂度较高。

总结来说，Dijkstra算法适用于单源最短路径问题，时间复杂度较低，适合稀疏图。而Floyd算法适用于多源最短路径问题，能够计算所有节点对之间的最短路径，适合稠密图，但其时间复杂度较高。

# 任务1：基于所给的旅游景点图、路径及权重（底层开发）

## 任务描述

编程实现：输入两个景点名，计算最短路径、最短路径长度。

|  |  |
| --- | --- |
|  |  |
| 旅游经典图 | 路径与权重 |

## 重要部分代码：

for (int k = 0; k < g->VexNum; k++) {

min = INF;

for (int i = 0; i < g->VexNum; i++) {

if (flag[i] == 0 && dist[i] < INF) {

if (dist[i] < min) {

min = dist[i];

f = i;

}

}

}

// flag[k] = 1;

flag[f] = 1;

for (int i = 0; i < g->VexNum; i++) {

temp = (g->Matrix[f][i] == INF ? INF : g->Matrix[f][i]);

if (flag[i] == 0 && temp + min < dist[i]) {

dist[i] = temp + min;

path[i] = f;

}

}

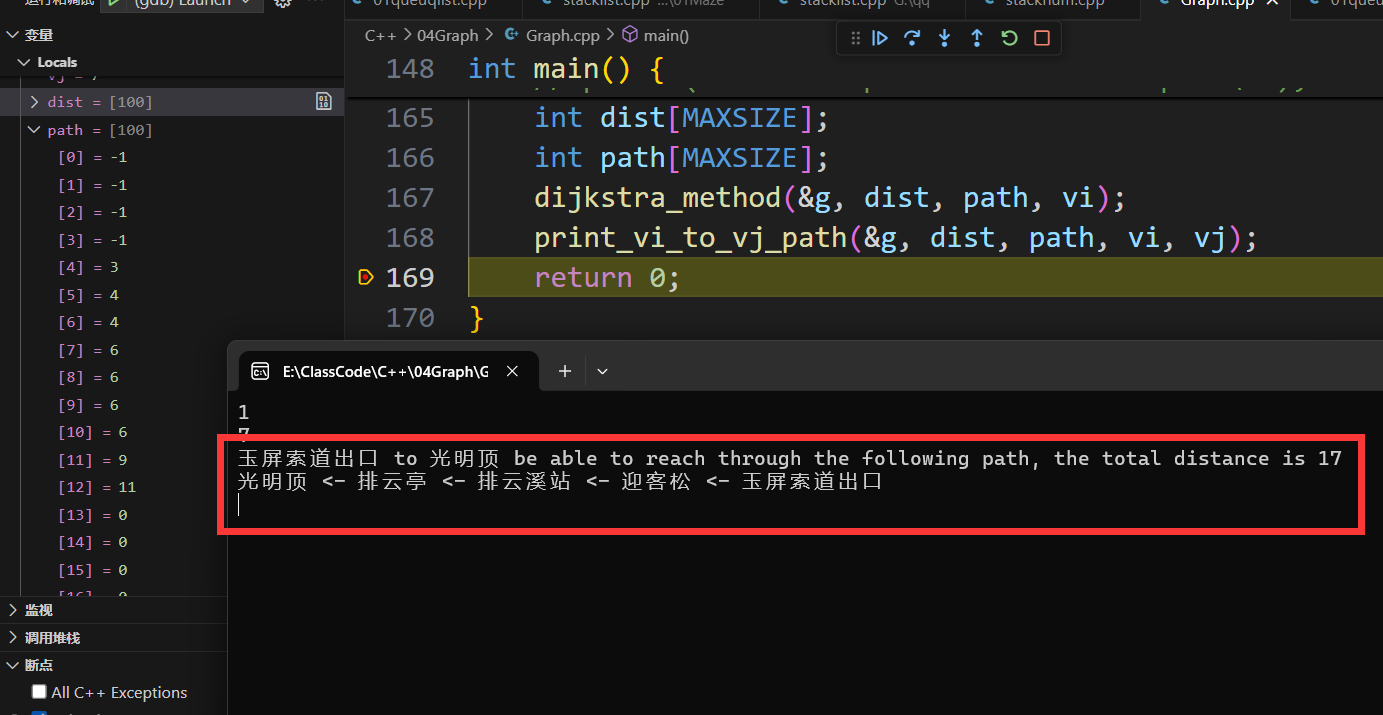
}

根据老师的参考代码修改了逻辑问题，正确的逻辑应该是，找到最小路径对应的节点，添加到flag中，如果加上的节点会产生更短的路径，就更新dist，循环k次，注意在这里k只用于计数，循环把所节点都加入flag，则可以找到vi到达所有节点的最短路径

## 结果展示

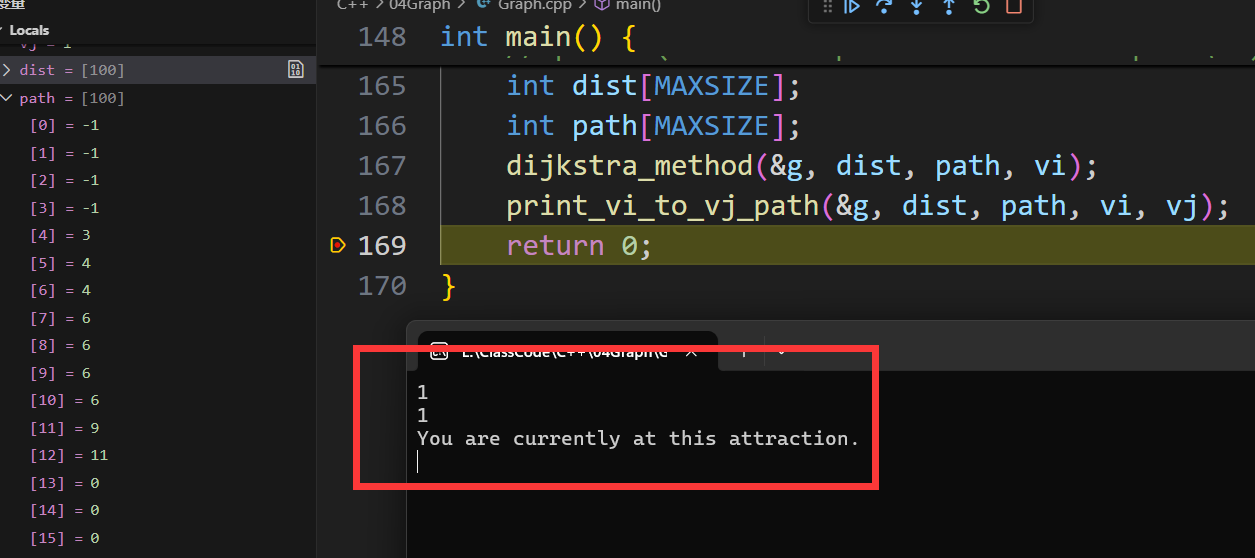
### 输入两个可到达的景点

可以正确显示路径以及距离



### 当输入的景点相同时

可以正确显示已经在该位置



### 当输入的景点不可直接到达时

可正确显示两地不可直接到达



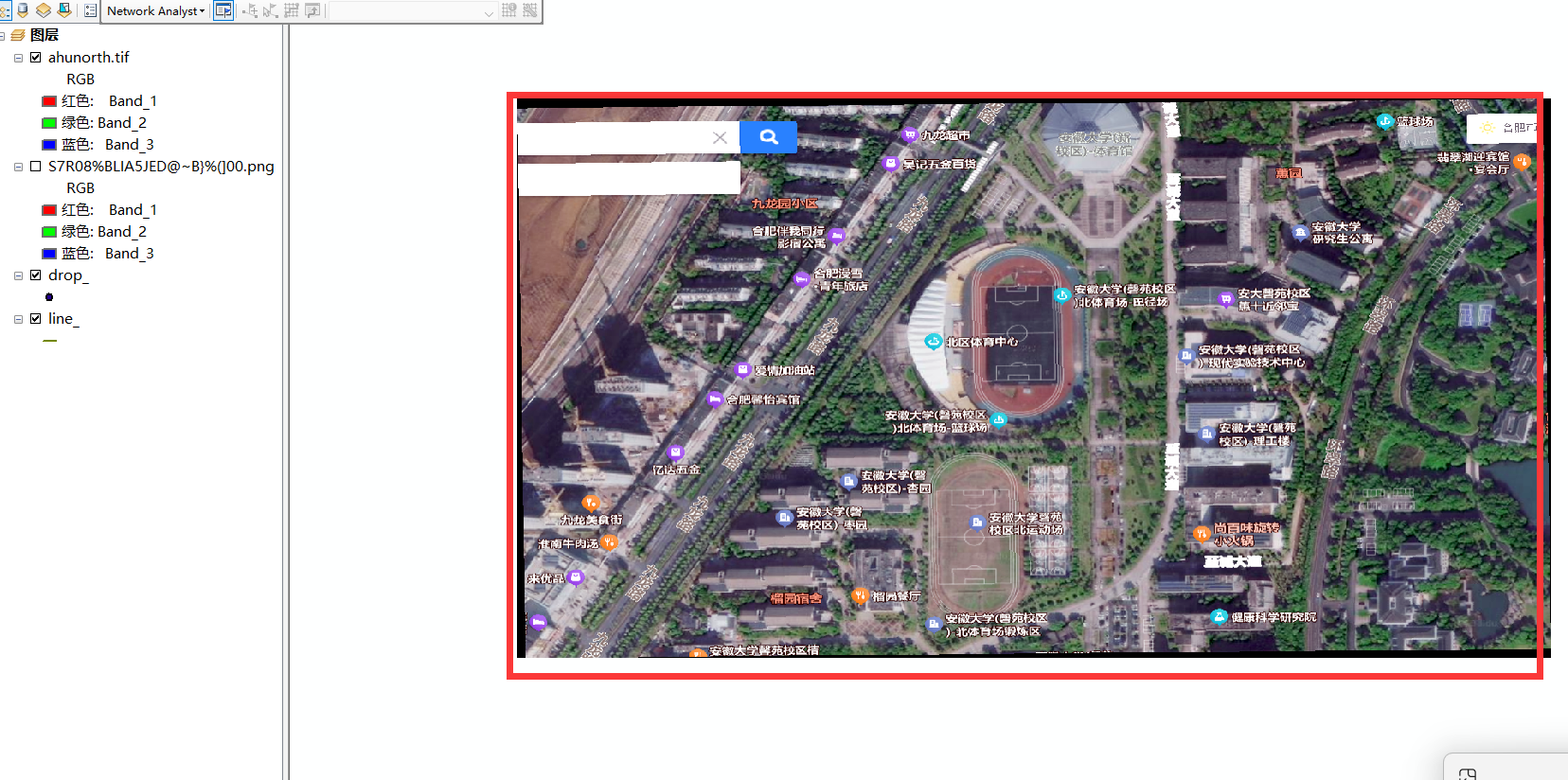
## 任务总结

这项任务相对来说并不困难，因为理解了Dijkstra算法的基本逻辑后，修改现有的参考代码就可以顺利完成。Dijkstra算法的核心思想是从源节点开始，通过逐步更新每个节点的最短路径，最终求得从源节点到其他所有节点的最短路径。算法的过程包括初始化距离数组、选择当前最短路径节点、更新其邻接节点的距离以及标记节点为已访问等基本步骤。只要理解了这些步骤，参考代码的修改工作就变得简单。通常，参考代码已经实现了大部分基础功能，可能只是一些细节或者特定场景的处理需要我们调整或优化。例如，可能需要处理图的表示方式、优化数据结构（如使用优先队列来加速最短路径的选择），或是加入一些特定的边界条件检测。总之，通过掌握Dijkstra算法的核心逻辑并在参考代码的基础上进行适当修改，完成任务并不困难。

# 任务2：ArcGIS最短路径分析—应用操作

## 底图获取

从百度底图上找到安大新区位置，切换至遥感影像模式，截图保存，我这里截取的是安大北门那里



## 地理坐标获取

在百度地图查看地理坐标，注意不要选择的距离太近，否则很容易产生图片严重变形，我选择一下坐标点：

北区体育中心：117.190531,31.778327

榴园餐厅：117.189817,31.775875

研究生公寓：117.194218,31.779497

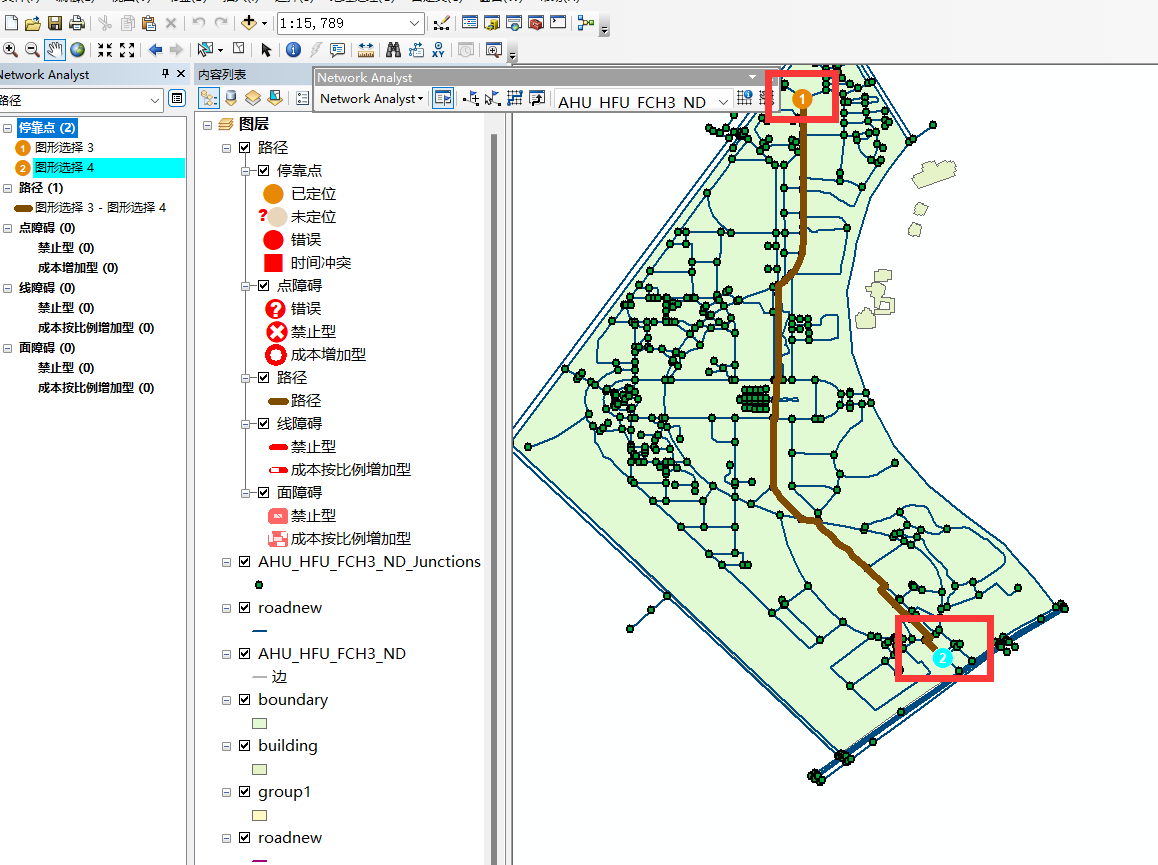
理工楼：117.193262,31.777421

旋转小火锅：117.193226,31.776424

（下面破的）北运动场：117.190989,31.776562

## 按照老师发的步骤逐步完成

导入老师给的安大坐标之后，用自带的工具，可以 实现最短路径的规划操作

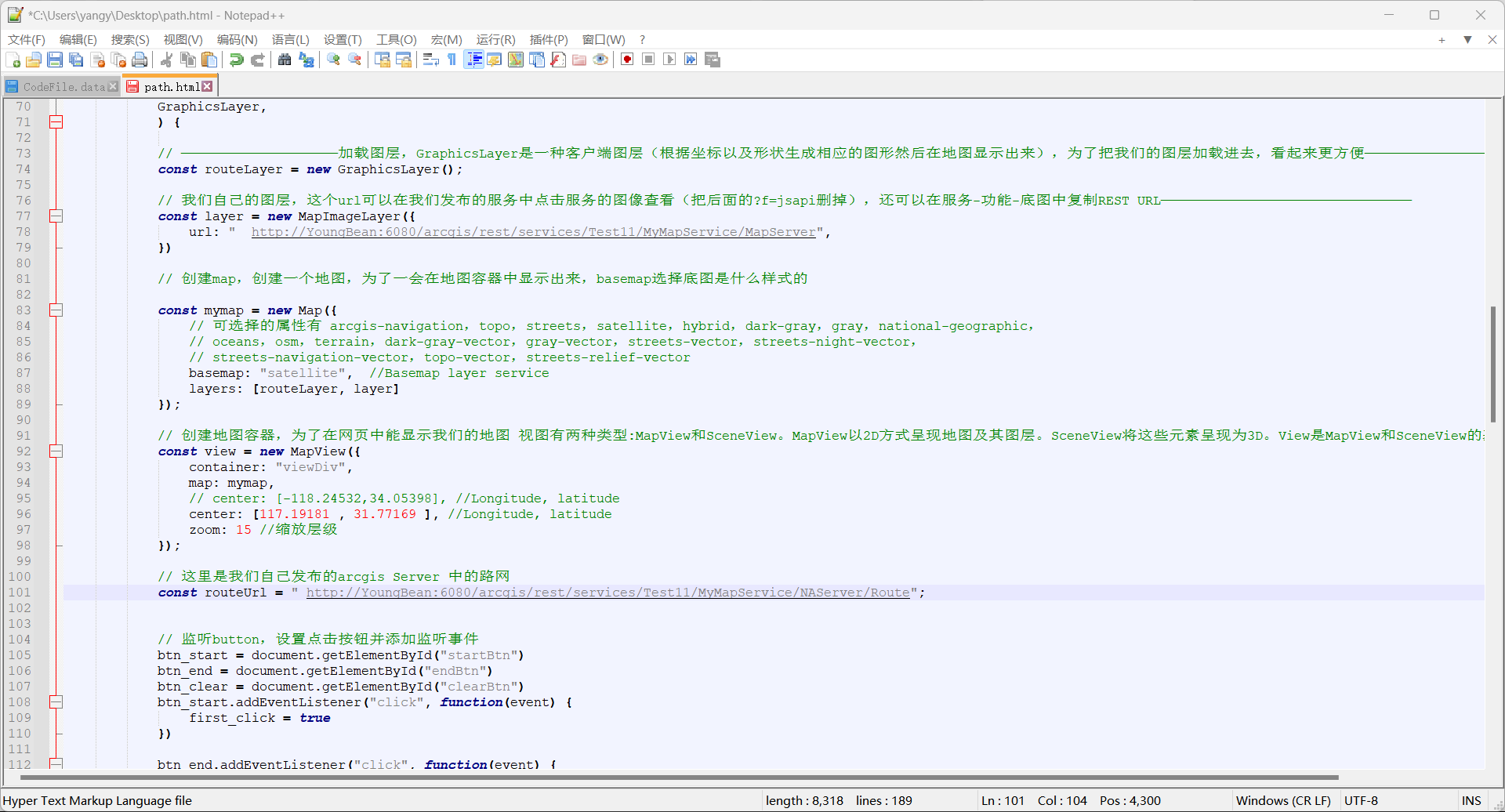


# 任务3基于ArcGIS API实现web端地图最短路径分析

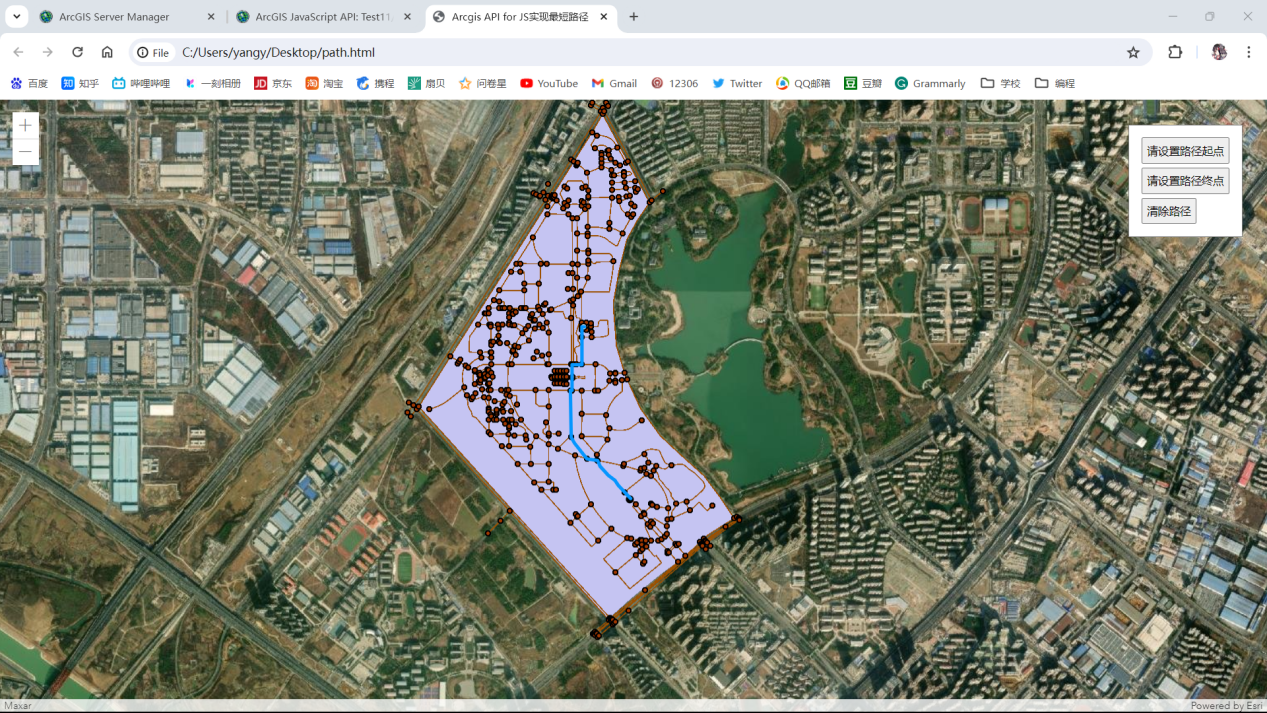
## 按照教程Html文件修改

按照教程创建html文件，按照要求修改路径，更改url与routeurl

|  |  |
| --- | --- |
|  |  |

修改后截图： 

## 结果截图



# 实验总结与常见问题汇总

## 实验总结

通过完成Dijkstra算法底层分析景点之间的最短路径，以及利用ArcGIS创建地图、进行分析并发布地图，对整个项目的实践过程有了深入的体会。第一阶段主要集中在Dijkstra算法的实现和底层逻辑的探索，深入理解了图论中的最短路径算法，掌握了理论知识，并通过实际编程体验了算法在处理复杂网络问题中的高效性。

在地图创建和分析环节，借助ArcGIS这一强大的地理信息系统平台，学会了如何将抽象的算法结果直观地可视化。这个过程理解了从数据分析到地图发布的完整工作流，包括数据准备、地图样式设计以及如何让分析结果以最友好的方式呈现给用户。特别是在使用ArcGIS API实现web端的地图最短路径分析时，接触到了一些网页开发的基础知识，如前后端的交互、API调用流程以及如何设计用户界面。这让我对前端开发有了一点了解，也体会到技术从底层实现到用户端可视化展现的完整过程。

整个实验意识到，从底层算法到前端可视化，每一个环节都不可或缺。底层算法提供了功能支持，地图分析和发布则将复杂结果形象化，而Web端开发则是将这一切推向用户的重要桥梁。这个从理论到实践再到产品化的过程，不仅巩固了算法知识，也对信息技术的整体应用有了全新理解。

## 常见问题汇总

### 输出端那里不能输入cin

使用vscode运行c++时很容易出现问题，但是可以修改配置文件来解决

解决方案：在setting.json中加入以下一段话，就会自动匹配到终端，就可以正常输出

"code-runner.runInTerminal": true,

加上这段话之后，就可以正常输出

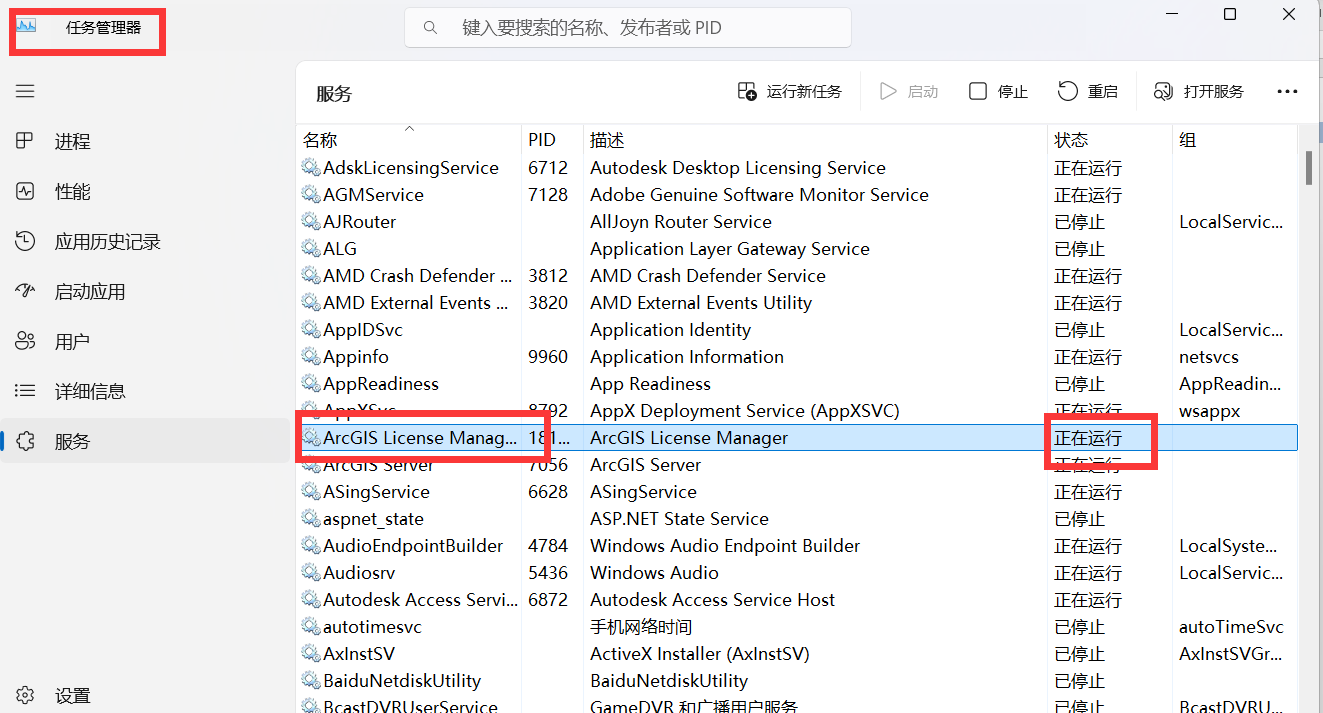
### 输出端中文都是乱码

但是调试和终端都不会出现

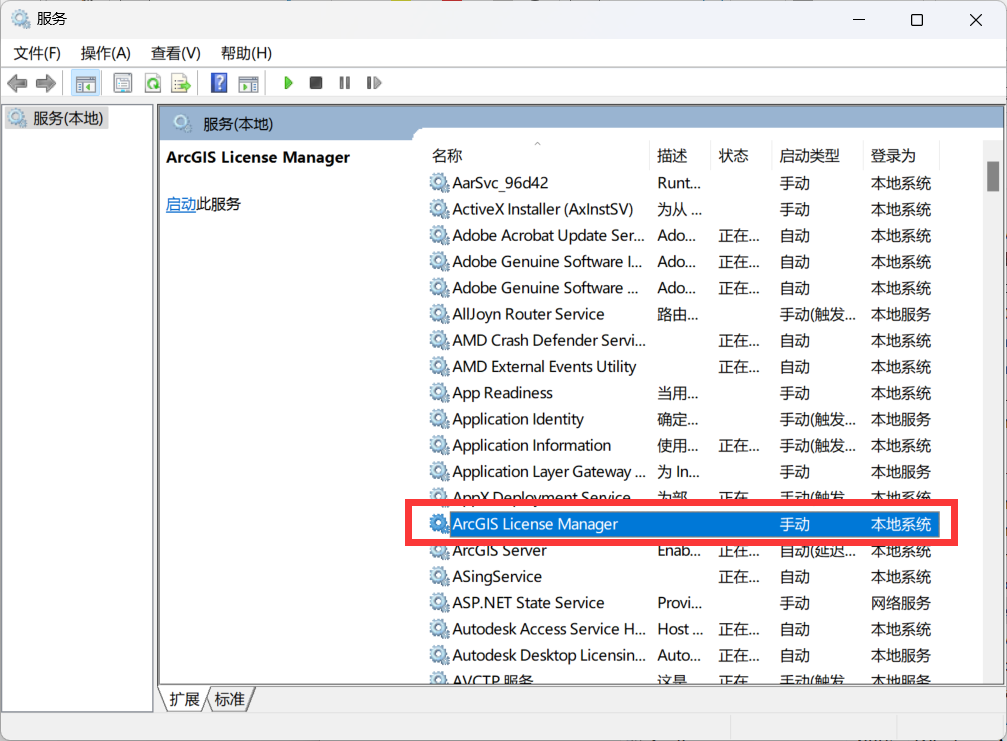
解决方案：暂无，只能在return 0 打断点，使其在终端输出

### arcmap没有license不能打开

打开arcmap时，可能会显示没有license不能打开，这是因为之前安装时，停止运行arcGIS License Manager，只要在任务管理器，找到arcGIS License Manager并保证其实在正常运行的状态就可以打开了

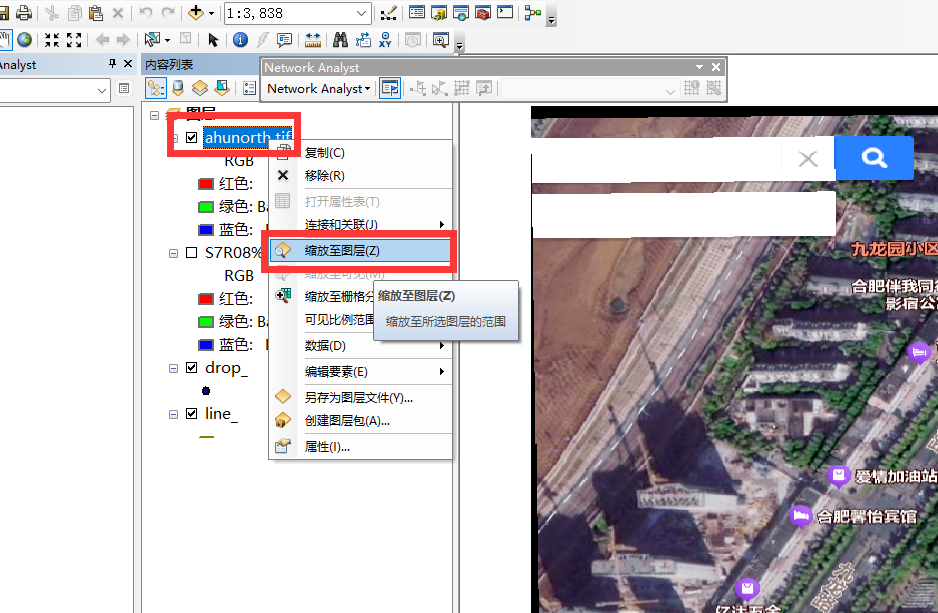


也可以设置这里是手动还是自动启动



### 面板中找不到图形

在标注点的过程中，经常出现图形由于不恰当的放大或者缩小，导致在面板中找不到图形，这时需要右键图层，点击缩放至图层即可



在选点的时候，尽可能让点均匀分布在图片上，否则图片就会产生比较大的变形