Python大作业文档

学号：5130379057 姓名：曹立

1. **项目介绍**
   1. 项目及功能介绍
   2. 项目文件列表及文件说明
   3. 使用说明
2. **编程实现说明**
   1. 逻辑视图
   2. 类图
   3. 核心算法

**1.1项目及功能介绍：**

本项目的名称为“Shanghai subway shortest path”。是一个用python实现的小程序。顾名思义，本程序的核心功能就是搜索上海轨道交通某两站之间的最短路径。上海轨道交通路线可谓是错综复杂。我常常往返于家和学校，坐地铁要坐二十多站，而且要几次换乘。我发现其实有好几条路线可以从家到学校，那我何不选择一条最快的路线呢。考虑地铁在两站之间的时间都差不多，那经过站数最短的路线应该就是最短最快的路线了吧。当然这不是绝对的，还有许许多多其他因素会决定一条路线的快慢，不过一条经过最少站数的路线还是很有参考价值的。因此我想写一个小程序可以让我们找出两个站点之间的最短路径。

**1.2项目文件列表及文件说明：**

项目文档.pdf：即本文档。

“可执行文件”文件夹中：

Application.pyc：Application.py编译后的文件；

Contants.pyc：Constants.py编译后的文件；

Graph.pyc：Graph.py编译后的文件；

ShanghaiSubwayGraph.gif：会显示在程序中的一张上海轨道交通图片，使用程序时输入的站名必须和上面标的一致才能找到相应站点；

ShanghaiSubwayGraph.txt：包含上海轨道交通路线的信息，运行程序时通过载入这个文件来完成建图；

ShanghaiSubwayShortestPath.py：本程序的入口；

“源代码”文件夹中：

Application.py：包含Application类，本程序运行将用到一个该类的实例，然后直接调用它的方法；

Constants.py：定义了本程序中用到的一些常量，主要都是些字符串；

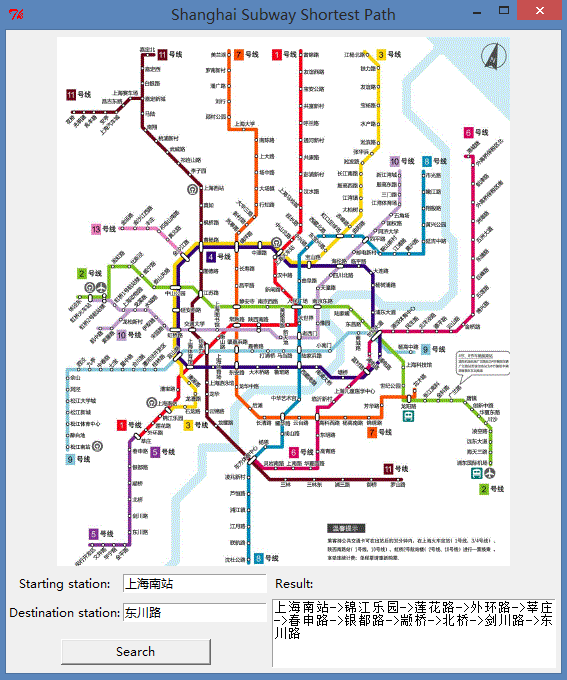
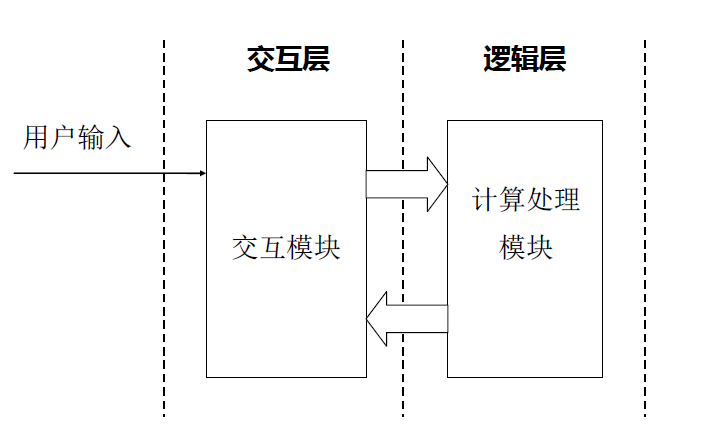
Graph.py：包含Vertex类和Graph类，Vertex是一个顶点，Graph是图类。本程序用这两个类来寻找最短路径；

ShanghaiSubwayShortestPath.py：本程序的入口；

**1.3使用说明：**

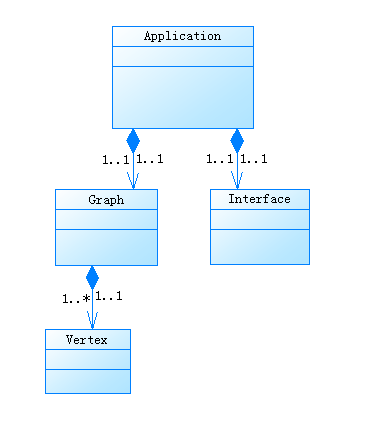
进入“可执行文件”文件夹，直接运行“ShanghaiSubwayShortestPath.py”即可。注意：上一节中“可执行文件”文件夹中的文件一个都不能少，负责将导致程序无法正常运行。运行之后，在Starting station后面的输入框输入起点站，在Destination station后面的输入框输入终点站，点击“Search”，然后就能在“Result”下面框里看到结果了。注意，起点站和终点站的名字必须和图片上的名字完全一致，否则会找不到该站点，程序结果就会显示站点不存在。

运行截图：



**2.1逻辑视图：**

程序的逻辑非常简单，主要就分为两个模块。交互模块接受用户的输入，并将结果呈现给用户。计算处理模块计算寻找出最短路径。

**2.2类图：**

根据程序的结构设计了以上的这些类，整个程序产生一个Application类的一个实例。一个Application类中含有一个Graph对象和一个Interface对象。一个Graph对象中含有若干个Vertex对象。

**2.3核心算法：**

本程序实现中最关键的一点就是搜索最短路径的广度优先算法，代码不长，实现如下：

|  |
| --- |
| #. Breath First Search .#  **def** BFS**(**self**,** origin**):**  self**.**clearVertices**()**  queue **=** Queue**.**Queue**(**maxsize**=-**1**)**  origin**.**visited **=** **True**  queue**.**put**(**origin**)**  **while** **not** queue**.**empty**():**  v **=** queue**.**get**()**  **for** adj **in** v**.**adjVertices**:**  **if** **not** adj**.**visited**:**  adj**.**visited **=** **True**  adj**.**distance **=** v**.**distance **+** 1  adj**.**path **=** v  queue**.**put**(**adj**)** |

这个过程从起始点出发，用广度优先搜索遍历整张图的所有节点。并记下到达每一个节点的上一个节点，即path属性。接下来就可以很容易地从终点找出最短路径一路到达起点。最后反一反就是我们要找的最短路径了，代码如下：

|  |
| --- |
| #. Return shortest path .#  **def** shortestPath**(**self**,** dest**):**  stack **=** Queue**.**LifoQueue**(**maxsize**=-**1**)**  path **=** ''  **while** **True:**  stack**.**put**(**dest**.**name**)**  **if** dest**.**path **==** **None:**  **break**  **else:**  dest **=** dest**.**path  **while** **True:**  path **=** path **+** stack**.**get**()**  **if** **not** stack**.**empty**():**  path **=** path **+** '->'  **else:**  **break**  **return** path |

过程也很简单。把一路上的节点一个个压入一个栈，因为最后是要反过来的。再一个个弹出，就得到了最短路径。

作者：曹立

邮箱：oos1111@sjtu.edu.cn