Python 大作业文档

学号：5130379057　　姓名：曹立

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；

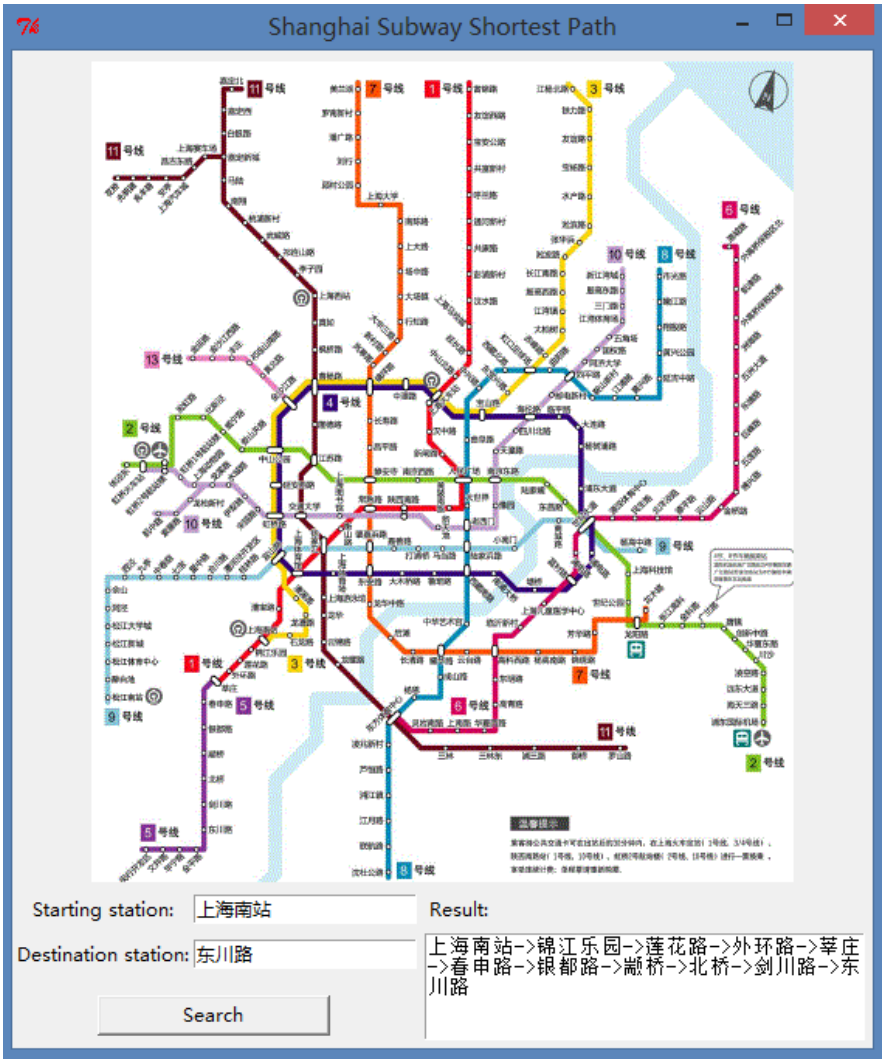
Graph.py：包含 Vertex 类和 Graph 类，Vertex 是一个顶点，Graph 是图类。本程序用这两个类来寻找最短路径；

ShanghaiSubwayShortestPath.py：本程序的入口；

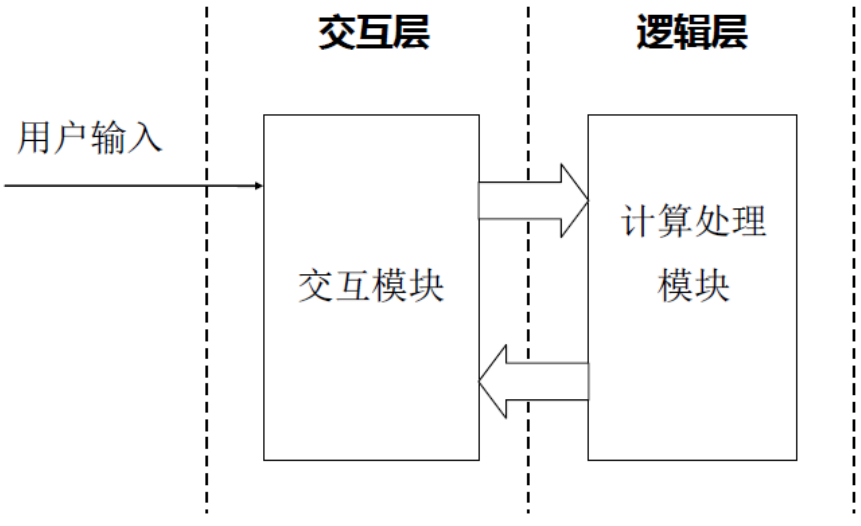
1.3 使用说明：

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：

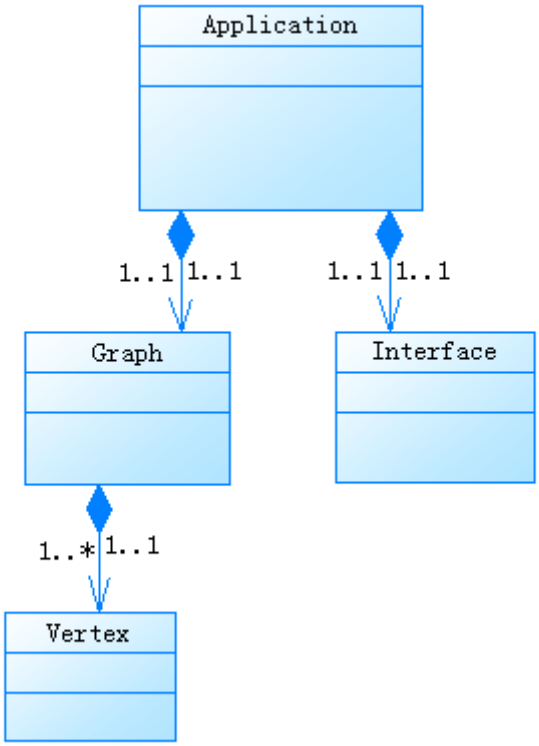

Shanghai Subway Shortest Path
Starting station:
上海南站
Destination station:
东川路
Search
Result:
上海南站->锦江乐园->莲花路->外环路->莘庄->春申路->银都路->颛桥->北桥->剑川路->东川路

2.1 逻辑视图：


交互层
逻辑层
用户输入
交互模块
计算处理模块

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