Project 4

**中山大学校园导游咨询**

**与最****短路径**

**班级：教务4班**

**姓名：王诚霖 学号：15331294**

**姓名：夏显茁 学号：15331322**

**姓名：冼圣杰 学号：15331323**

**【题目要求】**

1.从中山大学东校区的平面图中选区有代表性的景点（10-15个），抽象成一个无向带权图。以图中顶点表示校内各景点，存放景点名称、代号、简介等信息。

2.为来访客人提供图中任意景点的相关信息的查询。

3.为来访客人提供图中任意景点的问题查询，即查询任意两个景点之间的一条最短的简单路径。

4.区分汽车路线与步行路线。

**【数据结构与算法】**

1.问题抽象：封装Graph，隐藏这个无向有权图的数据信息，并提供相应的操作。

2.存储部分：对于景点（顶点）信息的存储，将每个景点抽象为一个结点（struct Node），内部包含有包括景点名，位置，景点电话等信息，最后用vector容器（vector<Node> \_spot）将所有景点保存下来；考虑到景点规模不大，对于景点之间的路径（边）的存储，采用邻接矩阵的存储方式，并使用文件来保存数据，实现数据的分离。

3.算法部分：查询景点信息：遍历vector容器（vector<Node> \_spot）容，得到对应的景点信息；查询最短路径：使用Dijkstra 算法，返回一个包含路径中所有景点名称和景点间距离的vector容器（vector<pair<string, int> >）。

**【测试数据、结果及分析】**

1.打印出所有的地点下标和对应的名字（如图1）。

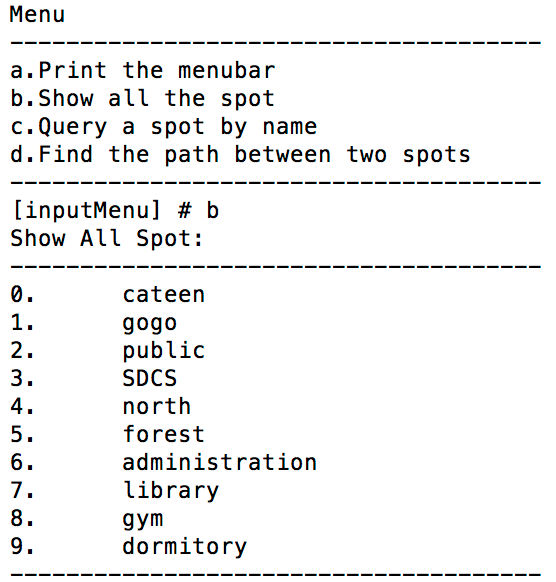
****

图 1

2.查询某个地点的详细信息（如图2），图中例子分别为gogo新天地、北实验楼和体育馆。

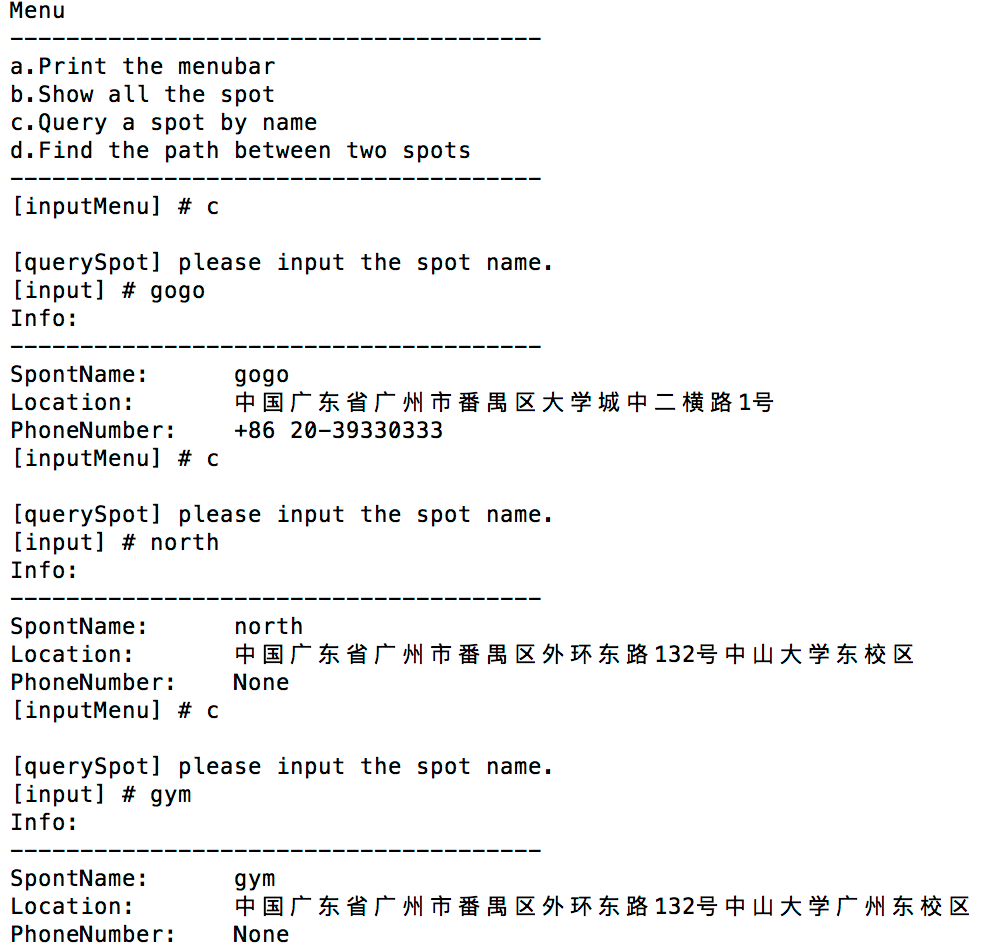
****

图 2

3.打印路径（之一），打印出最远的路径（如图3）。

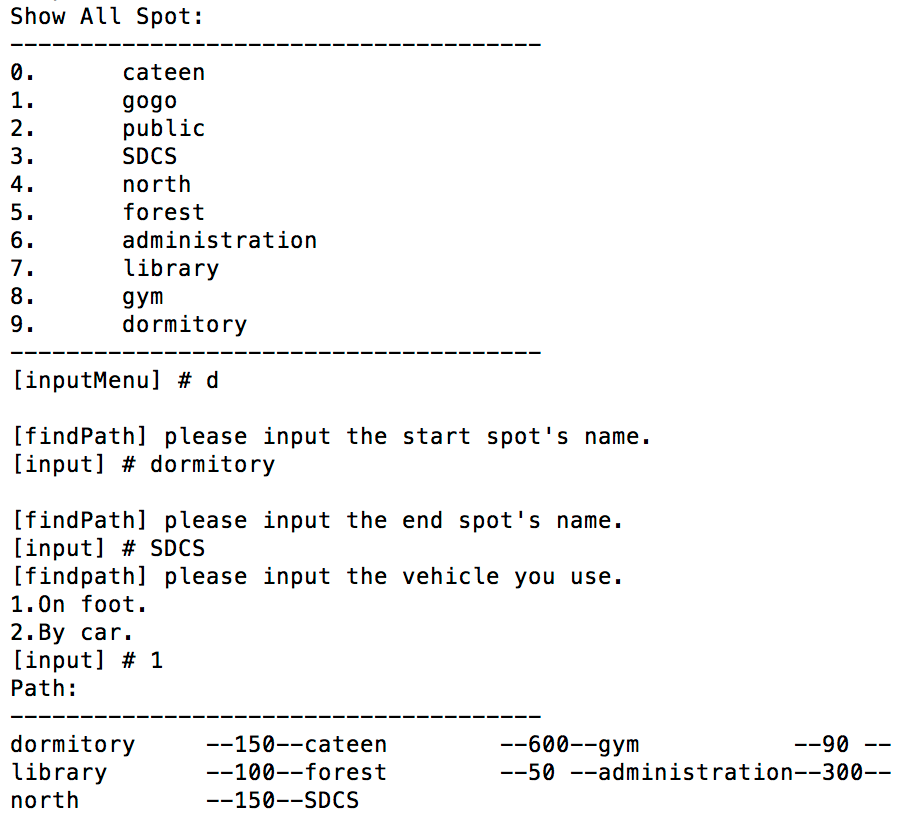
****

图 3

4.打印路径（之二），打印出坐车和走路不同的路径1（从三行情书林到行政楼），走路有捷径，而坐车没有（如图4）。

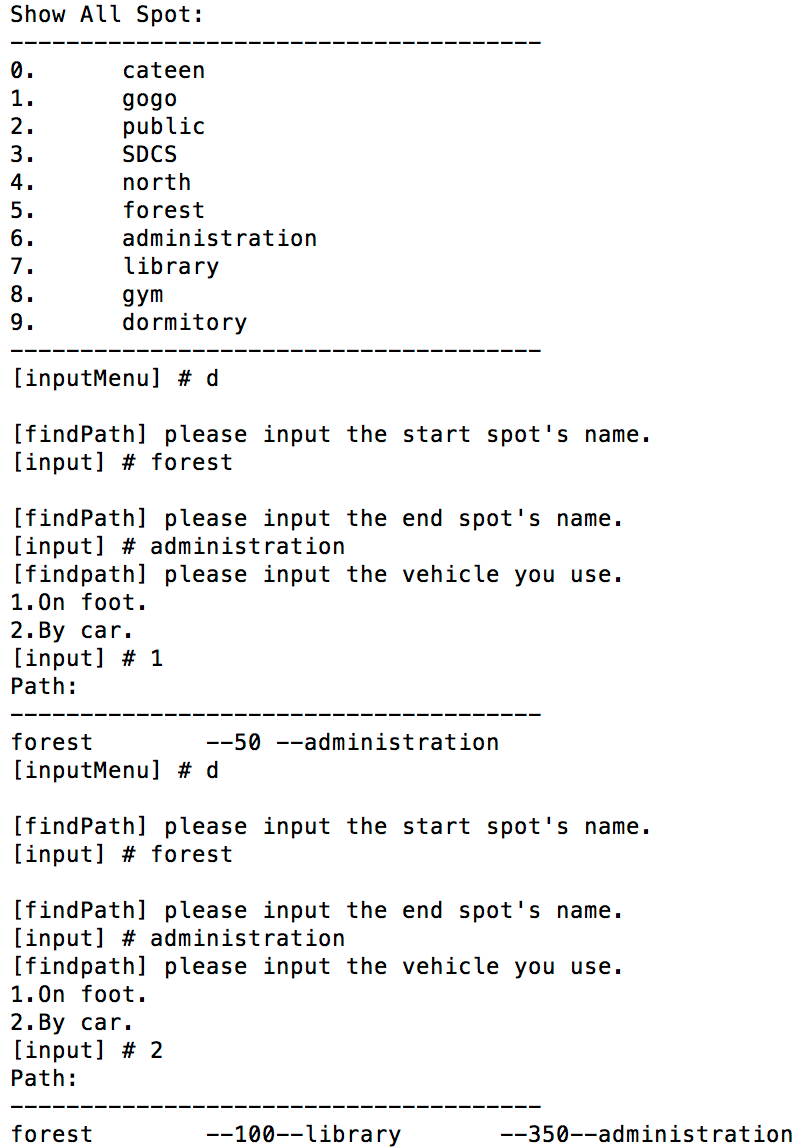
****

图 4

5.打印路径（之三），打印出坐车和走路不同的路径2（从公交楼到行政楼），坐车会绕远路（如图5）。

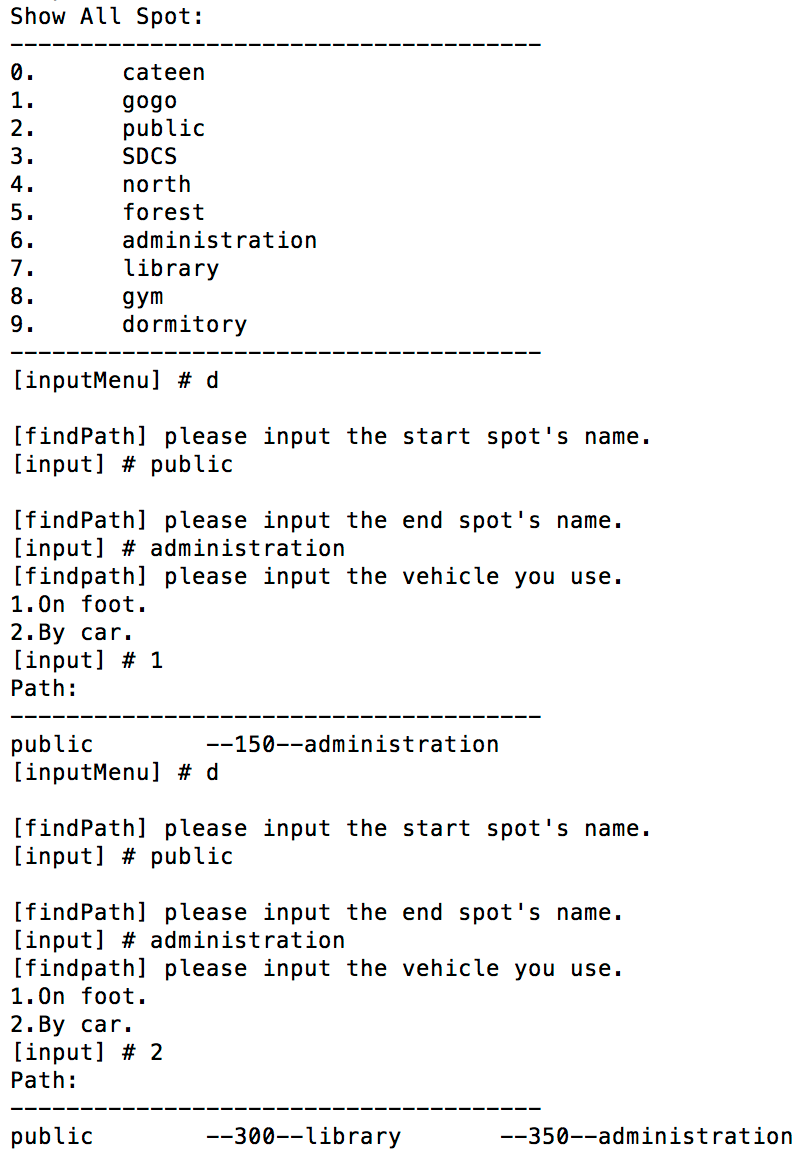


图 5

**【分工、贡献%、自我评分】**

|  |  |  |  |
| --- | --- | --- | --- |
| **组员** | **分工** | **贡献** | **自我评分** |
| **王诚霖** | **数据的收集和抽象** | **33%** | **95** |
| **夏显茁** | **路径的求解、项目分工** | **33%** | **95** |
| **冼圣杰** | **UI实现、数据测试** | **33%** | **95** |

**【项目总结】**

此次实验需要分别实现步行、车行的最短路径搜索，因此需要有两个图分别存储。此次我们使用两份文件分别保存两份图。好处有二，一是可以灵活切换步行、车行两种模式（通过change2walk、change2drive两个方法便可实现两种图的切换），一是实现数据持久化，增大程序的可拓展性（如果后续想扩大图的范围，只需修改相应图的文件即可，无需修改代码）。另外，程序运行时才读取数据，因此可以在程序运行的同时进行路径修改，十分方便。

此次实验求解最短路径，使用了Dijkstra算法，因为使用了邻接矩阵用于存储边的关系和边的权值，算法实现起来较为方便。为了获得目标路径中的每个节点，需要为每一个目的地创建一个vector，用于存储目标路径中每个节点和节点间的相应距离。此次求解最短路径算法的实现，使我们熟悉了Dijkstra算法，也让我们学会在原有算法的基础上进行修改以完成任务。

本次实验再次运用了github来管理代码，使得代码更行过程更加清晰，有理由项目开发的正常进行。在实验过程中，我们体会到了分工与合作的重要性，在实现本project时，我们首先把任务分割成三份，分别是UI、线路图的构造、路径查找算法的实现。在各自完成自己应有的任务之后，我们进行了合力debug，通常一个人看不出来的bug，另外一个人能迅速发现并提出解决方法。在不断的修改之后，我们终于完成了这次实验。

**【程序清单**】

