华东师范大学数据科学与工程学院上机实践报告

课程名称：算法设计与分析	年级：21级	上机实践成绩：
指导教师：金澈清	**姓名**：包亦晟	
上机实践名称：路径规划	**学号**：10215501451	**上机实践日期**：2023.6.3
上机实践编号：No.13	**组号**：1-451	

一、目的

1．熟悉算法设计的基本思想

2．掌握最短路径相关算法的思路

二、内容与设计思想

川西风光几枚，以下图片是川西路线图。张三是旅游爱好者，他从成都出发自驾到西藏江达。

1）从成都到江达的最短自驾路线是什么？可以用Dijkstra算法来求解。

2）张三把理塘列为必游之地。怎么规划路线，使得总行程最短？

3）张三觉得理塘风景很美，道孚也不错，两个地方如果能够去一个地方的话就心满意足了。应该怎么安排行程使得总行程最短？

4）张三在规划线路的时候，发现不同路况行驶速度不一样。地图中最粗的路径（成都-雅安）表示平均时速可以达到80公里每小时，而中等的路径表示平均时速每小时60公里每小时，最细的路径表示平均速度仅仅只有40公里每小时。那么(2)(3)中用时最短的路径分别是哪一条？

5）（***思考题***）考虑到Dijkstra算法仅仅从一段开始寻找路径，效率不高。李教授想到一个高招，就是同时从出发地和目的地进行搜索，扩展搜索节点，然后两个方向扩展的路径会在中途相遇，则拼接起来的路径就是最短路径。如何实现李教授这个想法？

三、使用环境

推荐使用C/C++集成编译环境。

四、实验过程

本次实验中由于存在某些交汇点并没有地名，所以我将泸定和康定之间的交汇点标记为A，将翁达和马尔康之间靠近翁达的交汇点标记为B，将翁达和马尔康之间靠近马尔康的交汇点标记为C。

task1

读取数据

最先碰到的问题是如何将数据从文件中读取出来。因为本次实验与上次实验不同，上次实验中只需要读取地点的经纬度，然后我们编写代码根据经纬度计算距离。而本次实验是直接给出的距离。

我使用了unordered_set，vector以及unordered_map。我的基本思路如下：是读取两次文件，第一次读取是记录下所有地点名，unordered_set用于去重。之后将unordered_set转化为vector便于用索引访问。第二次读取文件，根据unordered_map中的索引值将距离存入存放图的二维数组中。

一定要注意第三步当中的代码：

```
f.clear();
f.seekg(0, ios::beg);
```

当第一遍读取完之后，f的eof标记就是true了，这时调用seekg是没用的。要在seekg之前需要调用流对象的clear方法，把流的标记清除掉，清除以后就可以正常调用seekg方法了。

Dijkstra

Task1通过dijkstra算法即可解决。Dijkstra算法是一个非常经典的最短路径算法，用于计算一个节点到其他节点的最短路径。它是基于贪心思想的，每次从未确定距离的节点中选择距离起始点最近的点加入确定距离的节点的集合当中，并更新它与其相邻所有点到起始点的最近距离和前驱节点编号，就能够得到起始点到目标点的最短距离。

代码中用到的变量如下：

```
int g[N][N];int dist[N];bool st[N];int pre[N];
```

其中，g数组用于存储图；dist[i]数组保存起始点到第i节点的距离；st[i]数组记录了是否找到了起始点到第i节点的最短距离，true则找到了，false则没找到；pre数组用于记录路径。

用s来表示当前已确定最短距离的点的集合，基本思路如下：

- 将起始点的dist置为0，其他的置为无穷大
- for i = 0 to n - 1（只需要循环n-1次，因为起始点到起始点自身的距离为0是确定的）
    - 找到当前未在s中标记且距离起始点最近的点
    - 用该点更新其他点的距离
    - 将该点的st数组对应位置置为true

```
总长度为：920
江达 德格 新路海 马尼干戈 甘孜 炉霍 道孚 八美 丹巴 小金 日隆 映秀 都江堰 成都
```

task2

运用两次dijkstra算法即可，先后求出成都到理塘的最短路径和理塘到江达的最短距离，最后拼接在一起即可。要注意，别忘了要把所需要用到的变量重新初始化，即将dist数组全部初始化为无穷大，st数组初始化为全false。

```
江达 德格 白玉 巴塘 理塘 理塘 雅江 新都桥 康定 A 泸定 天全 雅安 成都
总长度为：1158
```

task3

方法和task2相同，最终得到的结果和task2中的结果比较下选出更短的即可。最终的结果和task1一致。

task4

在建图的时候，要将路程/时速来作为每条边的权值，再使用dijkstra算法即可。

该任务花了我一番功夫，因为路程/时速的结果可能是个**浮点数**！所以，不光读取文件的时候要把distance和speed改为double变量，g数组和dist数组也要改为double数组。同时，也不能将g和dist初始化为0x3f3f3f3f了，因为在浮点数中这个值并不代表着无穷大，要重新选择一个无穷大值。

```
总长度为：15.3333
江达 德格 新路海 马尼干戈 甘孜 炉霍 道孚 八美 丹巴 小金 日隆 映秀 都江堰 成都
```

task5

task5中的思路是可以基于dijkstra算法的。我们分别从起始点和目标点出发遍历图。一旦其中一个遍历到另一个已经遍历到的点的时候，就记录下该节点并停止。最后再根据pre数组各自回溯即可。

我们要将task1中用到的变量全都增加1个，然后在每次循环的时候都要同时更新这两个变量组。最最关键的代码是下面这一行：

```
int inter;
if (st1[t2] || st2[t1]) {
    inter = st1[t2] ? t2 : t1;
    break;
}
```

st1[t2]和st2[t1]中只要一个为true，那就代表其中一个遍历到了另一个已经遍历到点，记录下来后break掉即可。最终的结果和task1是相同的。

五、总结

对上机实践结果进行分析，问题回答，上机的心得体会及改进意见。

我在本次实验中还实现了dijkstra的堆优化版本。因为算法的主要耗时的步骤就是从dist数组中选出没有确定路径的节点中距离起始点最近的点 。我们很容易想到使用小根堆，而不需要遍历一遍。我们可以使用标准库中的优先队列来实现，具体可以看代码。要注意pair<int, int>的优先队列，不指定比大小规则的话，默认按照第一个int进行排序。所以我在代码实现中将pair中第一个int用于存放距离，第二个int用于存放节点编号。最终得到的结果和朴素版本的结果是一样的。

在上次实验中也提到过了，prim算法其实和dijkstra算法是有着些许的相似性的。前者是更新到集合的距离，而后者是更新到起始点的距离。我们可以结合这一点对算法流程进行记忆。