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**：22 级	**上机实践成绩**：
指导教师：金澈清	**姓名**：石季凡	
上机实践名称：路径规划	**学号**：10225501403	**上机实践日期**：2023.6.6
上机实践编号：No. 13	**组号**：1-403	

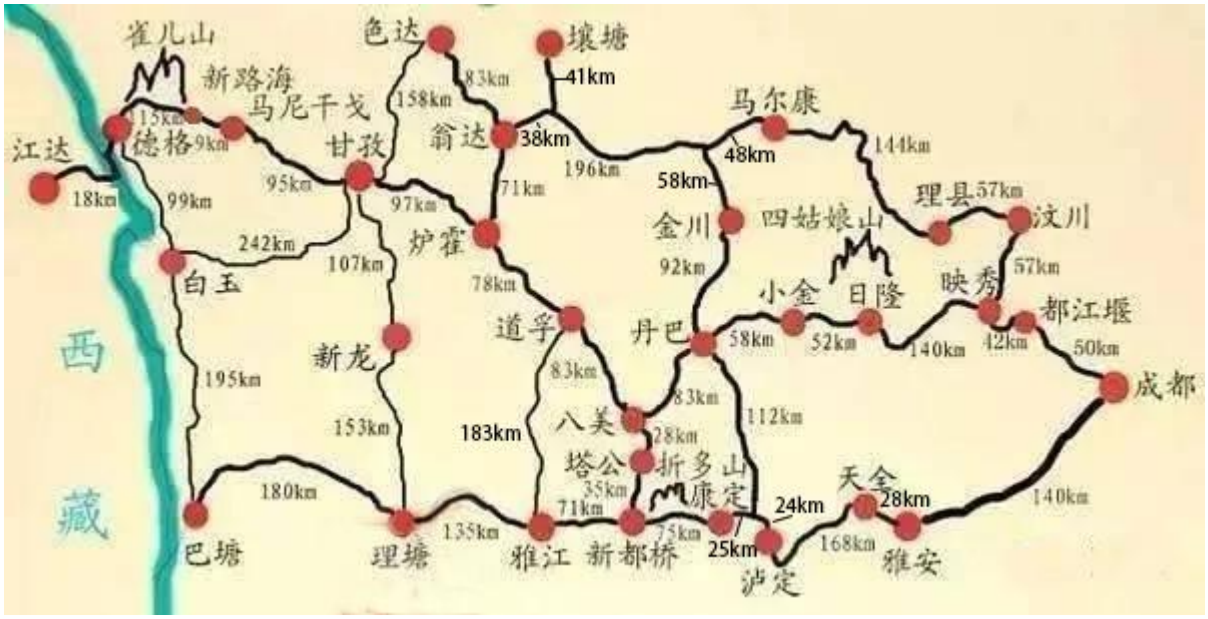
一、目的

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

C/C++集成编译环境。

四、实验过程

1. 编写相关实验代码
2. 写出算法的思路。

五、总结

【Dijkstra 算法的实现】:

构成该算法的函数有名字数字标记转化函数、道路输入函数、最短距离更新函数、获得最短路径函数、最短路径计算函数。

【search 函数】：给定一个地点的名字，获得其数字标记，若没有，则在链表中存入该地点，并赋予数字标记，返回数字标记。

【getName 函数】：给定一个数字标记，返回链表中对应的地点名字字符串。

【roadInput 函数】：给定两个数字标记，根据给定的数字标记，在道路指针数组中存入道路信息。道路结构体包括数字标记、距离和下一个结构体的指针，结构体指针数组的含义为，在某一个位置的指针下对应的链表，记录了到该位置的数字标记所代表城市的距离，如 roads[i]->location=j,roads[i]->diatance=k，这代表 i 号城市与 j 号城市有一条路，长度为 k，而通过 next 传递后得到一个新指针，即 tmp=roads[i]->next,tmp->location=l, tmp->distance=m，这代表 i 号城市和 l 号城市有一条道路，长度为 m，以此类推。

【Input 函数】：该函数主要作用为整合，根据输入的城市名字与距离调用 search 函数与 roadInput 函数，最终得到一个结构体指针数组 roads。

【updateDistances 函数】：当我们新放入一个节点后，我们需要对最短路径数组进行更新，即如果一个位置没有被放入路径中时，将其对应的地点相连的地点间的最短距离 Dstances 修改即，Distances[tmp->location] = Min(Distances[tmp->location], tmp->distance + Distances[location])，同时，若距离被更新，则修改被更新位置的 before 数组的数值，更新为输入的更新位置，该数组记录达到某位置路径的上一个地点。

【getMinlocation 函数】：获得当前可以加入路径的距离最近的地点的数字编号，通过遍历实现即可。

【computeMindistance 函数】：该函数负责整合调用上述函数，并计算出最短路径，初始化一个 Distances 数组，赋值为 100000，初始化 flag 数组，初始化为 1，用于标记是否为自由点，即没有被纳入路径，先将起点放入 Distances 数组并更新，然后开始不断调用 getMinlocation 获得最近位置，再调用 updateDistances 进行更新，当新放入的位置为终点时停止，最终返回 Distances 数组中终点位置对应的数值。

【解题】:

【1】：输入道路数 43，输入 43 条道路，格式为“地点 地点 距离”，最终得到答案以及路径如下

```
The min distance is 920 km
```

成都 → 都江堰 → 映秀 → 日隆 → 小金 → 丹巴 → 八美 → 道孚 → 炉霍 → 甘孜 → 马尼干戈 → 新路海 → 德格 → 江达

【2】：分别计算起点为成都，终点为理塘，起点为理塘，终点为江达，合并即可。

得到答案如下：

The min distance is 1158 km

成都 → 雅安 → 天全 → 泸定 → 岔口 1 → 康定 → 新都桥 → 雅江 → 理塘 → 巴塘 → 白玉 → 德格 → 江达

【3】：同上，两次拆分为两段进行计算，取最短的即可。

得到答案如下：

The min distance is 920 km

成都 → 都江堰 → 映秀 → 日隆 → 小金 → 丹巴 → 八美 → 道孚 → 炉霍 → 甘孜 → 马尼干戈 → 新路海 → 德格 → 江达

【4】：修改输入即可，即不再输入距离，而是输入距离和速度

第三小问最快路径如下：

成都 → 雅安 → 天全 → 泸定 → 岔口 1 → 康定 → 新都桥 → 雅江 → 理塘 → 巴塘 → 白玉 → 德格 → 江达

Time = 20.9667h

第四小问最快路径如下：

成都 → 雅安 → 天全 → 泸定 → 岔口 1 → 康定 → 新都桥 → 塔公 → 八美 → 道孚 → 炉霍 → 甘孜 → 白玉 → 德格 → 江达

Time = 15.33332h

【5】：算法修改如下：

由于双向的算法，原数函数直利用数组已经不能很直观地进行运算，我们可以整合一个结构体，结构体包扩距离出发点距离、距离终点距离、来自出发点路径的上一地点、来自终点路径的上一地点、是否被加入起点路径、是否被加入终点路径，然后我们还需要两个整形来记录终点出发路径的最短长度 endMin 和起点出发路径的最短长度 starMin，而 updateDistances 函数在更新路径后，需要再更新 endMin 和 startMin，更新也很简单，遍历路径末端即可。而 getMinlocation 函数则需要寻找距离出发点距离和距离终点距离这两个数值最短的一个节点的位置进行返回，至于 computeMindistance 函数，每次要新放入的节点 newInput 既在起点路径中，也在终点路径中，如果 Distances[newInput]->Edis + Distances[newInput]->Sdis <= (startMin + endMin)，则终止。

最终我们得到了相同的结果，即

The min distance is 920 km

成都 → 都江堰 → 映秀 → 日隆 → 小金 → 丹巴 → 八美 → 道孚 → 炉霍 → 甘孜 → 马尼干戈 → 新路海 → 德格 → 江达

【总结】：

双向 Dijkstra 算法由于其终止条件的缘故，在很多情况下可以节省操作的次数，但是由于其需要记录的信息也大量增加，因此这是一个通过空间取换取时间的思路。而在本次实验中，由于粗心犯下了很多的小错误，但是小错误却是最难发现的，因此在写代码的时候一定要一步一个脚印，减少粗心犯错。