算法设计与分析实验报告

实验名称：Dijkstra 算法(单源点最短路径)

一、问题陈述，相关背景、应用及研究现状的综述分析

1.问题陈述：

给定带权有向图，其中每条边的权都是非负实数。另外，还给定 **V** 中的一个顶点，称为源点。现在要计算从源到所有其它各顶点的最短路长度。这里路的长度是指路上各边权之和；

2. 相关背景：

迪杰斯特拉算法**(Dijkstra)**是由荷兰计算机科学家狄克斯特拉于 **1959** 年提出的，因此又叫狄克斯特拉算法。是从一个顶点到其余各顶点的最短路径算法，解决的是有权图中最短路径问题。迪杰斯特拉算法主要特点是以起始点为中心向外层层扩展，直到扩展到终点为止。

二、模型拟制、算法设计和正确性证明

1.算法设计：

算法首先求出长度最短的一条路径，然后参照它求出长度次短的一条最短路径，以此类推，直到从顶点 v 到其他各顶点的最短路径全部求出为止。

利用贪心选择性质和最优子结构性质可以证明算法的正确性

算法步骤如下：

G={V,E}

1. 初始时令 S={VO},T=V-S={其余顶点}，T 中顶点对应的距离值

若存在<VO,Vi>，d(VO,Vi)为<VO,Vi>弧上的权值

若不存在<VO,Vi>，d(VO,Vi)为∞

2. 从 T 中选取一个与 S 中顶点有关联边且权值最小的顶点 W，加入到 S 中

3. 对其余 T 中顶点的距离值进行修改：若加进 W 作中间顶点，从 VO 到 Vi 的距离值缩短，则修改此距离值

重复上述步骤 2、3，直到 S 中包含所有顶点，即 W=Vi 为止

三、时间和空间复杂性分析

时间复杂度：

$$O(n^2)$$

(3) 计算复杂性

对于具有n个顶点和e条边的带权有向图，如果用带权邻接矩阵表示这个图，那么Dijkstra算法的主循环体需要 $O(n)$ 时间。这个循环需要执行n-1次，所以完成循环需要 $O(n^2)$ 时间。算法的其余部分所需要时间不超过 $O(n^2)$ 。

空间复杂度：

没有使用额外空间

S（n）=O(1)

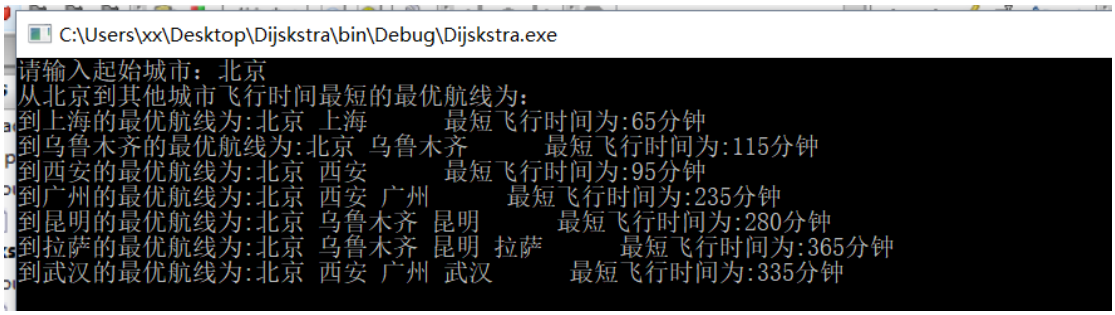
四、程序实现和实验测试过程

源程序见 Dijkstra.cpp

测试过程：
结点以及权值数据

```
Type cities[8]={"北京","上海","乌鲁木齐","西安","广州","昆明","拉萨","武汉"};
Type flight_routes[16][2]={{"北京","上海"},{"上海","北京"},{"北京","乌鲁木齐"},
{"乌鲁木齐","北京"},{"北京","西安"},{"西安","北京"},{"西安","广州"},{"广州","西安"},
{"拉萨","昆明"},{"昆明","拉萨"},{"拉萨","武汉"},{"武汉","拉萨"},{"乌鲁木齐","昆明"},{"昆明","乌鲁木齐"},{"武汉","广州"},{"广州","武汉"}};
Weight flight_time[16]={65,65,115,55,95,100,140,80,85,85,90,90,165,165,100,100};
Weight flight_time_dist[8];
int flight_time_path[8];
MGraph flight_time_graph;
createMGraph(flight_time_graph,cities,8,flight_routes,flight_time,16);
```

测试过程：



五、总结

利用堆可以在图为稀疏图时进行优化

该算法复杂度为 n^2,我们可以发现，如果边数远小于 n^2,对此可以考虑用堆这种数据结构进行优化，取出最短路径的复杂度降为 O(1)；每次调整的复杂度降为 O（elogn)；e 为该点的边数，所以复杂度降为 O((m+n)logn)。