# Lab3 航班查询系统实验报告

## 一、功能简介

1. 机场遍历查询
   提供每个机场的两种遍历查询，分别为**DFS查询**与**BFS查询**。可以将用户输入的机场在现有的航线内可以通过无上限的中转到达的机场全部显示出来。
2. 机场连通性查询
   提供在计算机计算能力允许范围内的中转次数上限内，用**矩阵图**表示任意两个机场的连通性。
3. 机场航线数目查询
   提供在计算机计算能力允许范围内的中转次数上限内，任意两个机场之间的航线条数。
4. 航线最短时间查询
   提供在现有航线内，用户输入任意两个机场之间的可达的最短时间。
5. 特定要求航线查询
   包含了三个按需查询模块，分别为**按起飞时段要求查询，按降落时段要求查询，按机型要求查询**。用户可以在在现有的航线中任意查询满足中转上限的航线
6. 最低航费航线查询
   包含了三个按需查询模块，分别为**按起飞时段要求查询，按降落时段要求查询，按机型要求查询**。系统将呈现任意两个机场之间所需的最低航费及其所对应的航线。
7. 特殊中转航线查询
   提供在用户给定中转时间上限与中转次数上限的情况下，为用户呈现所有满足条件的航线。
8. 特殊航费航线查询
   提供在用户给定中转时间上限与中转次数上限的情况下，为用户呈现最低航费及其对应的的航线。
9. 开发者选项
   提供短时间内遍历所有航线的快捷入口。

## 二、具体实验方式与测试样例

> 对于连续输入输出，下面所有样例均采取红进黑出的方式。

### 机场遍历查询

1. BFS查询
   - 实现方式
     首先建立邻接表，标明每个机场可以达到的其他机场。然后通过建立队列，从初始点开始，不断进行入队与出队操作。对可以到达的机场进行标记，最后，带有标记的机场就是可以到达的机场。
   - 测试样例
     - *输入与输出：*
       请输入您想查询的机场ID:
       1
       当前1机场可以达到的机场为:
       1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79
       提示:查询成功!执行重置操作

2. DFS查询
    - 实现方式
      首先建立邻接表，然后从初始机场开始，找到一个符合要求的机场后进行递归，直到目前路径上所有的机场均已被标记，然后进行下一个的机场的递归，直到将该初始机场所有的可行结果都标记完全。
    - 测试样例
        - *输入与输出：*
          请输入您想查询的机场ID:
          1
          当前1机场可以达到的机场为:
          1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79
          提示:查询成功!执行重置操作

---

## 机场连通性查询

- 查询
    - 实现方式
      建立二维数组，表示每个机场的连通性。在for循环中，通过使用两个一维数组，根据中转次数更新可以到达的机场，达到查询无上限次的中转次数的机场的目的。最后将每个结果归入二维数组即可。
    - 测试样例

      > 因结果为80*80的矩阵，在报告中无太大实际意义，故略去。

---

## 机场航线数目查询

- 查询
    - 实现方式 建立队列，根据BFS的思路，以中转次数为判断是否出队的依据，进行入队与出队操作，直到队列为空。
    - 测试样例一
        - *输入与输出：*
          请输入您想查询的第一个机场ID:
          49
          请输入您想查询的第二个机场ID:
          56
          请输入您想查询的中转次数:
          0
          第49号机场与第56号机场在中转次数为0的条件下有4条航线
          提示:查询成功!执行重置操作

---

## 航线最短时间查询

- 查询

- 实现方式
  基本思路是dijkstra算法。将航线记录中所有从起始点出发的记录确定下来，然后通过for循环，每次选取一条边进入下一步的建图，以到起点的距离，即距离出发时间的长短为依据进行选择，使用优先级队列进行排序，降低复杂度。
- 测试样例一
  - *输入与输出：*
    请输入您想查询的第一个机场ID:
    1
    请输入您想查询的第二个机场ID:
    2
    从第1号机场到第2号机场最短需要4035分钟。
    提示:查询成功!执行重置操作
- 测试样例二
  - *输入与输出：*
    请输入您想查询的第一个机场ID:
    56
    请输入您想查询的第二个机场ID:
    39
    从第56号机场到第39号机场最短需要1640分钟。

---

## 特定要求航线查询

1. 起飞时段要求
   - 实现方式
     采用dijkstra算法，不同的是，在本功能中，在初始点的所有航线记录中，加入了对起飞时段的判断，并且判断到起点的距离依据变为了中转次数，判断到起始点中转次数最少的航线，并且在超过中转次数之后会弹出优先级队列，达到剪枝的效果。
   - 测试样例
     - *输入与输出：*
       请输入您的第一个机场ID:
       56
       请输入您的第二个机场ID:
       34
       请输入您的中转次数上限:
       6
       请输入您的起飞时段上限:(例:5/5/2017 12:20)
       5/1/2017 0:00
       请输入您的起飞时段下限:(例:5/5/2017 12:20)
       5/9/2017 0:00
       为您查询到7条符合条件的航线
       450 -> 1147
       451 -> 1149
       462 -> 1150
       474 -> 1148
       491 -> 1960 -> 1097 -> 1058
       492 -> 350 -> 355 -> 1150

493 -> 1994 -> 1584
提示:查询成功!执行重置操作

---

2. 降落时段要求
   - 实现方式
     采用dijkstra算法，不同的是，在本功能中，在即将入队的的所有航线记录中，加入了对目的地降落时段的判断，并且判断到起点的距离依据变为了中转次数，判断到起始点中转次数最少的航线，并且在超过中转次数之后会弹出优先级队列，达到剪枝的效果。
   - 测试样例
     - *输入与输出：*
       请输入您的第一个机场ID:
       42
       请输入您的第二个机场ID:
       37
       请输入您的中转次数上限:
       8
       请输入您的降落时段上限:(例:5/5/2017 12:20)
       5/5/2017 23:30
       请输入您的降落时段下限:(例:5/5/2017 12:20)
       5/9/2017 0:00
       为您查询到11条符合条件的航线
       999 -> 108
       1000 -> 108
       1001 -> 109
       1002 -> 109
       1133 -> 947
       1138 -> 946
       1140 -> 948
       1894 -> 555 -> 2156
       1895 -> 555 -> 2156
       2210 -> 1909 -> 2016
       2220 -> 1909 -> 2016
       提示:查询成功!执行重置操作

---

3. 机型要求
   - 实现方式
     采用dijkstra算法，不同的是，在本功能中，在即将入队或者出队的的所有航线记录中，加入了对航班机型的判断，并且判断到起点的距离依据变为了中转次数，判断到起始点中转次数最少的航线，并且在超过中转次数之后会弹出优先级队列，达到剪枝的效果
   - 测试样例
     - *输入与输出：*
       请输入您的第一个机场ID:
       54
       请输入您的第二个机场ID:
       28
       请输入您的中转次数上限:
       6

请输入您的机型:
2
为您查询到8条符合条件的航线
338 -> 1962 -> 2108
339 -> 1963 -> 2108
371 -> 407 -> 2108
878 -> 166 -> 2108
1538 -> 407 -> 2108
1542 -> 458 -> 2108
1816 -> 166 -> 2108
2052 -> 1978 -> 2108

---

## 最低航费航线查询

1. 起飞时段要求
    - 实现方式
      采用dijkstra算法，不同的是，在本功能中，在即将入队的所有航线记录中，加入了对起飞时段的判断，并且判断到起点的距离依据变为了航费，判断到起始点航费最少的航线。
    - 测试样例
        - *输入与输出：*
          请输入您的第一个机场ID:
          46
          请输入您的第二个机场ID:
          72
          请输入您的起飞时段上限:(例:5/5/2017 12:20)
          5/1/2017 0:00
          请输入您的起飞时段下限:(例:5/5/2017 12:20)
          5/9/2017 0:00
          已为您查询到最低航费航线!共花费1634航费.
          884 -> 18
          提示:查询成功!执行重置操作

---

2. 降落时段要求
    - 实现方式
      采用dijkstra算法，不同的是，在本功能中，在即将出队的所有航线记录中，加入了对起飞==降落时段的判断，并且判断到起点的距离依据变为了航费，判断到起始点航费最少的航线
    - 测试样例
        - *输入与输出：*
          请输入您的第一个机场ID:
          56
          请输入您的第二个机场ID:
          42
          请输入您的降落时段上限:(例:5/5/2017 12:20)
          5/3/2017 0:00
          请输入您的降落时段下限:(例:5/5/2017 12:20)
          5/9/2017 0:00

已为您查询到最低航费航线!共花费894航费.
450 -> 1040
提示:查询成功!执行重置操作

---

3. 机型要求
   - 实现方式
     采用dijkstra算法，不同的是，在本功能中，在即将入队与出队的所有航线记录中，加入了对机型的判断，并且判断到起点的距离依据变为了航费，判断到起始点航费最少的航线
   - 测试样例
     - *输入与输出：*
       请输入您的第一个机场ID:
       61
       请输入您的第二个机场ID:
       18
       请输入您的机型:
       2
       已为您查询到最低航费航线!共花费3643航费.
       1782 -> 32 -> 1403

---

## 特殊中转航线查询

1. 单次中转时间上限
   - 实现方式
     采用dijkstra算法，在本功能中，判断到起点的距离依据变为了中转次数，判断到起始点中转次数最少的航线，并且在超过中转次数之后会弹出优先级队列，达到剪枝的效果。但即使是这样，时间也是远远不够的，所以需要单次中转时间作为依据来减枝，达到符合要求的运行时间。
   - 测试样例
     - *输入与输出：*
       请输入您的第一个机场ID:
       39
       请输入您的第二个机场ID:
       10
       请输入您的中转次数上限:
       4
       请输入您的中转时间上限:
       610
       为您查询到36条符合条件的航线
       2300 -> 148 -> 117 -> 1369
       2300 -> 148 -> 215 -> 1369
       2300 -> 148 -> 1300 -> 1369
       2300 -> 167 -> 185 -> 1369
       2300 -> 167 -> 471 -> 1369
       2300 -> 204 -> 215 -> 1369
       2300 -> 204 -> 1300 -> 1369
       2300 -> 283 -> 377 -> 1369
       2300 -> 556 -> 557 -> 1369

2300 -> 872 -> 876 -> 1369
2300 -> 886 -> 1817 -> 1369
2300 -> 1013 -> 1033 -> 1369
2300 -> 1013 -> 1063 -> 1369
2300 -> 1194 -> 1157 -> 1369
2300 -> 1243 -> 1215 -> 1369
2300 -> 1246 -> 1208 -> 1369
2300 -> 1276 -> 1290 -> 1369
2300 -> 1293 -> 666 -> 1369
2300 -> 1293 -> 1256 -> 1369
2300 -> 1416 -> 1423 -> 1369
2300 -> 1463 -> 1464 -> 1369
2300 -> 1568 -> 1570 -> 1369
2300 -> 1810 -> 1817 -> 1369
2300 -> 2117 -> 378 -> 1369
2300 -> 2269 -> 99 -> 1369
2300 -> 44 -> 614 -> 522 -> 1370
2300 -> 44 -> 1763 -> 99 -> 1369
2300 -> 560 -> 682 -> 1480 -> 1370
2300 -> 567 -> 1990 -> 125 -> 1370
2300 -> 567 -> 2179 -> 1480 -> 1370
2300 -> 670 -> 1234 -> 1358 -> 1370
2300 -> 670 -> 1880 -> 1063 -> 1369
2300 -> 872 -> 1861 -> 629 -> 1370
2300 -> 872 -> 2311 -> 1480 -> 1370
2300 -> 1194 -> 1667 -> 125 -> 1370
2300 -> 1243 -> 1254 -> 99 -> 1369
提示:查询成功!执行重置操作

2. 总计中转时间上限
   - 实现方式
     采用dijkstra算法，在本功能中，判断到起点的距离依据变为了中转次数，判断到起始点中转次数最少的航线，并且在超过中转次数之后会弹出优先级队列，达到剪枝的效果。但即使是这样，时间也是远远不够的，所以需要总计中转时间作为依据来减枝，达到符合要求的运行时间。
   - 测试样例
     - *输入与输出：*
       请输入您的第一个机场ID:
       39
       请输入您的第二个机场ID:
       10
       请输入您的中转次数上限:
       4
       请输入您的中转时间上限:
       610
       为您查询到1条符合条件的航线
       2300 -> 283 -> 377 -> 1369
       提示:查询成功!执行重置操作

## 特殊航费航线查询

1. 单次中转时间上限
   - 实现方式
     采用dijkstra算法，在本功能中，判断到起点的距离依据变为了航费，判断到起始点花费最少的航费的航线，与上功能比起来时间复杂度更甚，所以需要单次中转时间作为依据来减枝，达到符合要求的运行时间。
   - 测试样例
     - *输入与输出：*
       请输入您的第一个机场ID:
       9
       请输入您的第二个机场ID:
       68
       请输入您的中转时间上限:
       600
       已为您查询到最低航费航线!共花费3650航费.
       1505 -> 1078 -> 490 -> 1230
       提示:查询成功!执行重置操作
2. 总计中转时间上限
   - 实现方式
     采用dijkstra算法，在本功能中，判断到起点的距离依据变为了航费，判断到起始点花费最少的航费的航线，与上功能比起来时间复杂度更甚，所以需要总计中转时间作为依据来减枝，达到符合要求的运行时间。
   - 测试样例
     - *输入与输出：*
       请输入您的第一个机场ID:
       9
       请输入您的第二个机场ID:
       68
       请输入您的中转时间上限:
       600
       已为您查询到最低航费航线!共花费6774航费.
       1185 -> 178 -> 124 -> 204 -> 215 -> 1229
       提示:查询成功!执行重置操作

---

## 开发者选项

1. 所有机场邻接表
   - 实现方式
     实现简略的边与顶点结构体，遍历航班信息，建立邻接表。
   - 测试样例

   > 结果在报告中无太大实际意义，故略去。

2. 所有机场遍历

- 实现方式
  通过for循环调用每个机场的DFS或者BFS遍历即可。
- 测试样例

> 结果在报告中无太大实际意义，故略去。

3. 所有机场最短时长
   - 实现方式 通过for循环调用每个机场的航线最短时间查询功能即可。
   - 测试样例

> 结果在报告中无太大实际意义，故略去。

---

# 三、优势与不足

## 优势

- 功能极其**全面**，到目前为止，此款系统到达了作者以往所有作品的巅峰。丰富而全面的功能使得作者有自信可即时投放到市场，可以满足绝大部分用户的需求。
- 航班查询系统依旧继承了本系列用户友好的传统特点，每一步都有详尽的使用指导与错误提示，简洁优美的界面可以给用户提供较好的体验。

## 不足

- **部分算法缺乏优化**，时间复杂度较高，部分特殊要求需要较长时间的运算。
- 缺乏**在线读取**数据功能，灵活性较为缺乏。
- 存在可能的bug

# 四、总结

1. 收获
   本次Lab磨砺了自己对图的理解。从对知识一知半解，到磕磕绊绊写完框架，到最后运用解决问题的得心应手，极大的巩固了图的知识,培养了自己从知识到实践的能力转化，发自肺腑地感受到受益匪浅。
2. 参考资料


- 《数据结构(c++语言版)》邓俊辉编著 清华大学出版社
- 基于优先队列的Dijkstra算法
- dijkstra 算法优先队列实现


3. 项目地址
   本次Lab所有文件已上传至https://github.com/rucerchui/Sophomore/tree/main/Lab2

# 欢迎您的使用!

---