实验报告

题目：设计一个某航空公司航线图，用户可以提出不同的的航班时段、时长、航费额度或者机型要求，程序返回航班线路（航班 ID 顺序表）。

班级：理科实验班6班 姓名：李梓童

学号：2017202121 完成时间：2018.12.

【目录】

一、需求分析

二、概要设计

三、详细设计

四、调试分析

五、用户使用说明

六、测试结果

七、经验与体会

1. 需求分析

设计一个某航空公司航线图，用户可以提出不同的的航班时段、时长、航费额度或者机型要求，程序返回航班线路（航班 ID 顺序表）。

1. 具有交互界面或图形界面。

2. 用邻接表来完成从任意机场出发的遍历，包括深度优先遍历和广度优先遍历。

3. 使用邻接矩阵表来完成任意两个机场的可连通性，包括是否可以直飞、1 次中转、2 次中转等。

4. 求任意两个机场之间的最短飞行时间。

5. 仅限直飞或 1 次中转，求任意两个机场的航线（航班 ID 顺序表）。

6. 给定起飞时段或者降落时段或者机型要求，求任意两个机场的多个备选航线（航班 ID 顺序表）。

7. 给定起飞时段或者降落时段或者机型要求，求任意两个机场之间的航费（机票价格）最低的路径。

1. 概要设计

1.数据结构定义：

程序将涉及到如下图结构的数据类型，用C语言描述如下：

①路线信息结构体：

typedef struct RouteInfo

{

int DprtTime; //出发时间

int FlgtLength; //飞行时间

int ArivTime;

int Cost; //航班费用

int StrtPoint; //出发点

int EndPoint; //目的地

int PlaneType; //飞机型号

int FlgtID; //航班ID

}RouteInfo;

②机场结点结构体

typedef struct RouteNode{

int RouteEnd; //航班终点

RouteNode \*nextroute; //下一个航班

RouteInfo info; //航班信息

}RouteNode;

③邻接表结构体

typedef struct{

StPortList Ports;

int \_RouteNum; //航班数

int \_PortNum; //机场数

}PortGraph;

④邻接矩阵结构体

typedef struct { // 图的定义

int Ports2[PortNum]; // 顶点向量

RouteMatrix RouteMap; // 邻接矩阵

int \_PortNum2, \_RouteNum2; // 机场数和路线数

} RMap;

2.模块设计：

1）主程序模块：主函数设计如下

使用提示；

输入操作代码；

根据操作代码执行相应操作；

结束一轮工作，提示是否继续操作；

2）求任意两个机场之间的最短飞行时间。

3）仅限直飞或 1 次中转或 2次中转，求任意两个机场的航线（航班 ID 顺序表）。

4）给定起飞时段或者降落时段或者机型要求，求任意两个机场的多个备选航线（航班 ID 顺序表）以及其中航费最低的路线。

5） 给定飞行时长或者中转时间限制，求任意两个机场的多个备选航线（航班 ID 顺序表）。

3.各模块间的调用关系：

主函数

求最短飞行时间

直飞/ 1 次中转/2次中转，求两个机场的航线

给定飞行时长/中转时间限制，求两个机场的多个备选航线

给定起飞时段/降落时段/机型要求，求两个机场的多个备选航线以及其中航费最低的路线

三、详细设计

**1、头文件**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <math.h>

**2、存储结构设计**

//路线信息结构体

typedef struct RouteInfo

{

int DprtTime; //出发时间

int FlgtLength; //飞行时间

int ArivTime;

int Cost; //航班费用

int StrtPoint; //出发点

int EndPoint; //目的地

int PlaneType; //飞机型号

int FlgtID; //航班ID

}RouteInfo;

//机场结点结构体

typedef struct RouteNode{

int RouteEnd; //航班终点

RouteNode \*nextroute; //下一个航班

RouteInfo info; //航班信息

}RouteNode;

//出发机场结点

typedef struct StPortNode{

StPortNum data; //出发机场的编号

RouteNode \*frstroute; //第一个航班

}StPortNode,StPortList[PortNum];

//邻接表结构体

typedef struct{

StPortList Ports;

int \_RouteNum; //航班数

int \_PortNum; //机场数

}PortGraph;

//用于邻接矩阵边的存储

typedef struct ArcCell { //弧的定义

int rlt; // 顶点关系类型,1或0示相邻否

RouteInfo info; // 该弧相关信息的指针

} ArcCell, RouteMatrix[PortNum][PortNum];

//邻接矩阵结构体

typedef struct { // 图的定义

int Ports2[PortNum]; // 顶点向量

RouteMatrix RouteMap; // 邻接矩阵

int \_PortNum2, \_RouteNum2; // 机场数和路线数

} RMap;

**3、特殊功能实现：**

（1）给定飞行时长限制，求任意两个机场的多个备选航线（航班 ID 顺序表）。

设置一个临时变量t，存储飞行时长的累加值。相应的递归函数中，每一次递归中，都会将t和给出的时长限制tmax比较，如果不超过则输出。

（2）给定中转时间限制，求任意两个机场的多个备选航线（航班 ID 顺序表）。

每次调用递归函数时，都会比较与下一次要调用递归函数的所传参数之间的差值，如果不超过中转时间限制，则可继续递归。

四、调试分析

1、程序设计过程中遇到的几个主要问题：

1）在设计递归函数时，由于递归终止条件不准确（如应当是p==null时终止还是p->next==null时终止）出现了一些错误。

2）邻接表和逆邻接表在书写上混淆，找出毛病用了一些时间。

3）用邻接矩阵判断直飞、一次中转、两次中转的时候，多重循环的条件没有把握好（比如Rm[i][j]==0时应当过滤不进入内部循环），调了几次才修正。

2、时空复杂度分析：

1）求解最短飞行时间：采用递归算法，时间复杂度为O(nlogn)；开辟了常数个空间去存储临时值，空间复杂度为O(1)。

2）用邻接矩阵判断直飞、一次转机、两次转机航线：采用循环遍历，共有三个大循环，最大的循环有三层，时间复杂度为O(n^3)；开辟了常数个空间去存储临时值，空间复杂度为O(1)。

3）给定起飞时段或者降落时段或者机型要求，求任意两个机场的多个备选航线（航班 ID 顺序表）以及其中航费最低的路线。采用递归算法求出符合要求的航线，时间复杂度为O(nlogn)；开辟了常数个空间去存储临时值，空间复杂度为O(1)。再从已有航线中通过循环找出花费最小的路线，时间复杂度为O(n)，开辟了常数个空间去存储临时值，空间复杂度为O(1)。

4） 给定飞行时长或者中转时间限制，求任意两个机场的多个备选航线（航班 ID 顺序表）。采用递归算法求出符合要求的航线，时间复杂度为O(nlogn)；开辟了常数个空间去存储临时值，空间复杂度为O(1)。

五、用户手册

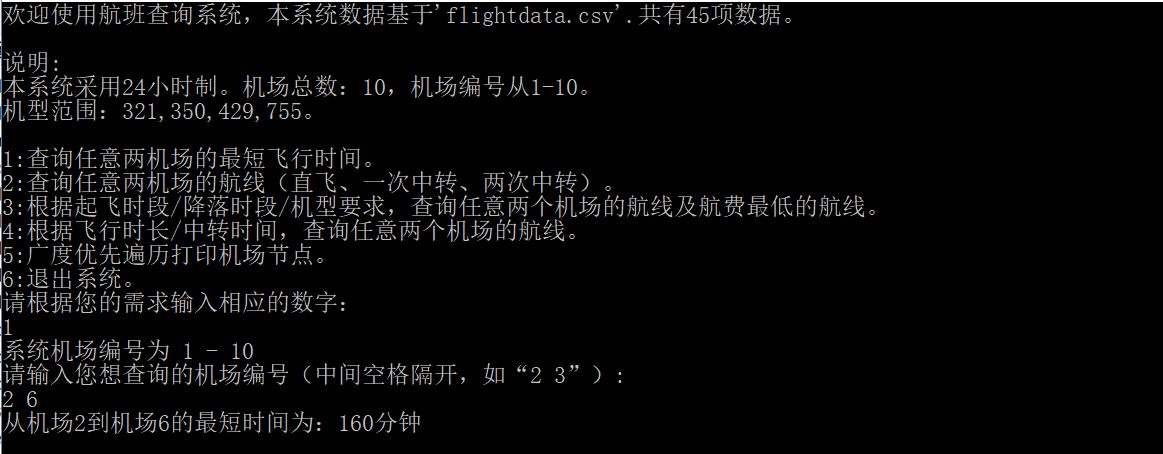
1)先输入对应序号，进行功能选择；

2)按照程序之中的提示进行操作

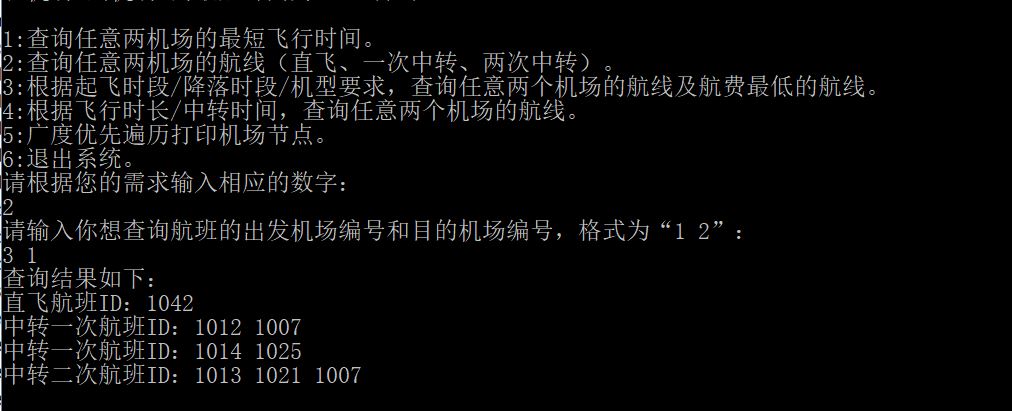
3)输入相应数字继续使用计算器，输入6退出程序。

六、测试结果

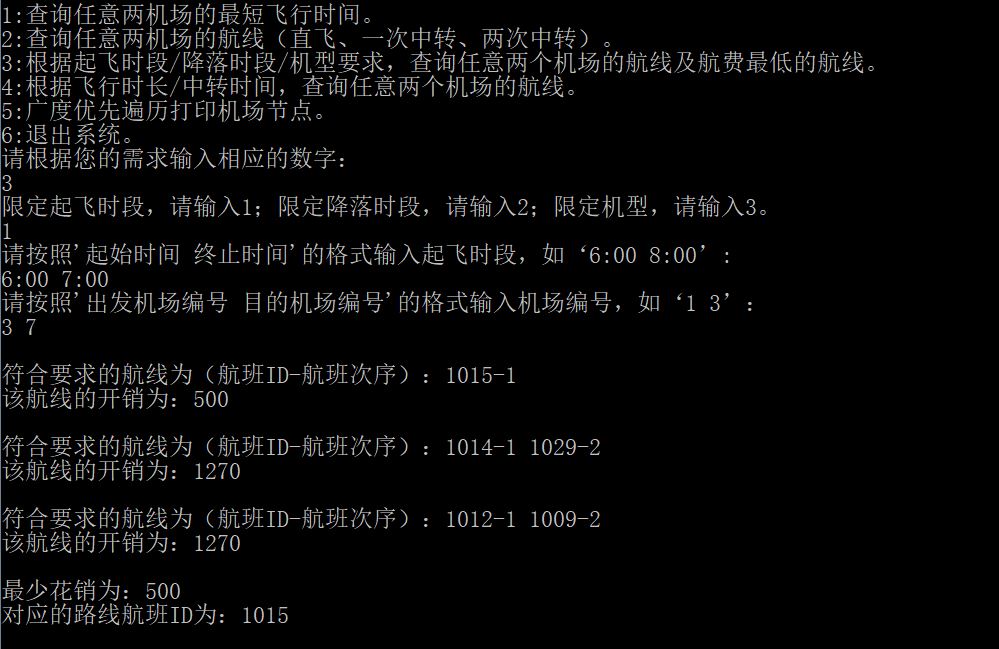
1）最短飞行时间



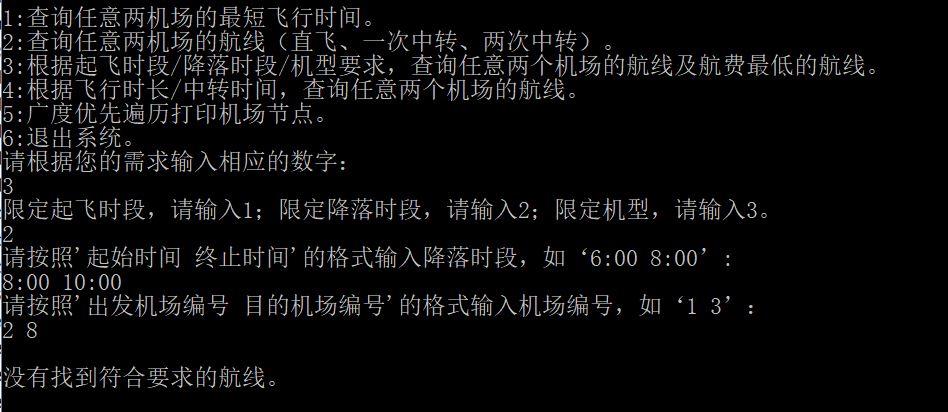
2）查询两机场航线



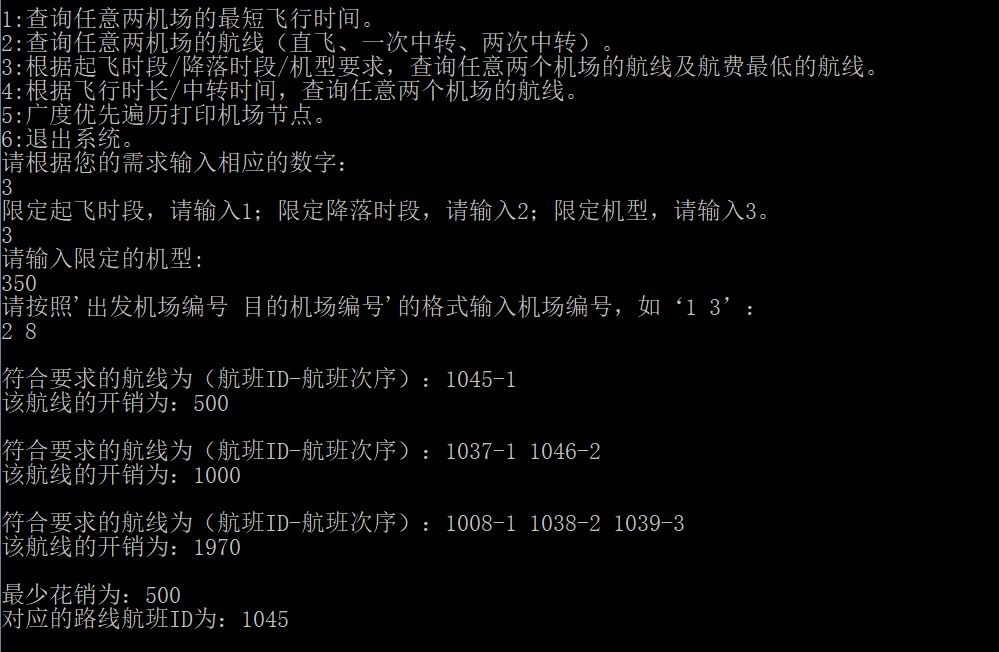
3）根据起飞时段查询



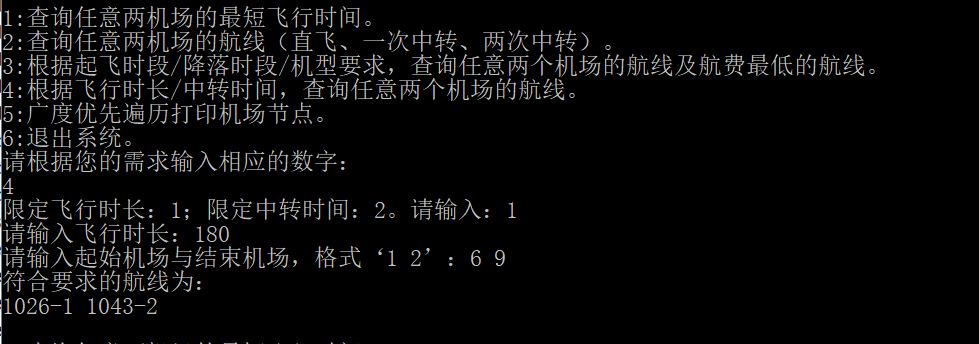
4）根据降落时段查询



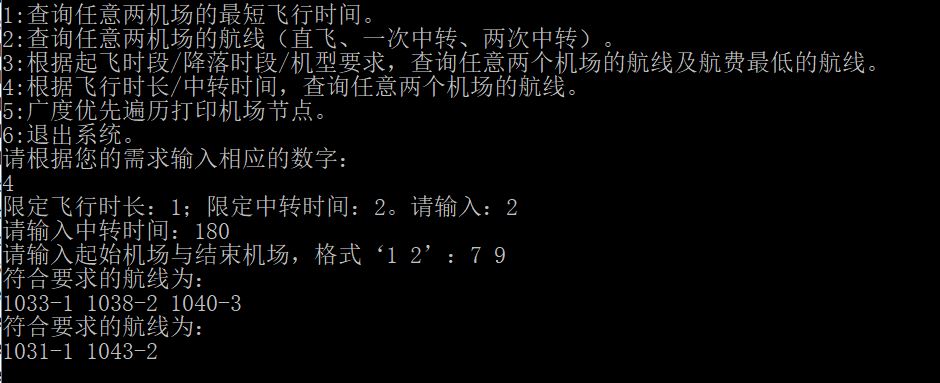
5）根据机型要求查询



6）限定飞行时长查询



7）限定中转时间查询



七、经验与体会

1、编写代码的能力得到锻炼。这次设计应用程序相比以前考虑更多细节的完善，比如输入无效值的检验、递归边界条件的设置等。

2、对课本知识有了更深的理解。航线图的设计很多思路都可以借鉴书本上的内容，在运用邻接表、邻接矩阵等相关知识的时候，对书本知识也有了更好的掌握。

3、图的实现得到了练习。图是非常重要的一类数据结构，编写程序的过程中也考虑到很多之前没有考虑过的问题，比如如果图的一个节点到另一个节点不只一条路径会怎样，如此等等。有向图相较无向图更为复杂，它的实现也在这次程序编写中得到了练习。