

《数据结构》课程设计报告

组员姓名 (学号): 刘威 21160538 李晓宇 21160536 李宇尧 21160537		组号:
提交日期	成绩	指导教师
问题解析 (对问题的分析、解题思路与解题方法): 问题分析: 一个飞机订票系统, 完成对航空客运订票的业务管理。该系统至少包括如下功能模块: 航班信息浏览、查询航线、客票预订和办理退票等。 <ol style="list-style-type: none">1. 录入: 可以录入航班情况 (数据可以存储在一个数据文件中, 数据结构、具体数据自定);2. 航班信息浏览: 预览已经建立的全部航线的相关信息;3. 查询: 可以查询某个航线的情况 (如输入航班号, 可以查询起、降时间, 起飞、抵达城市, 航班票价, 票价折扣, 所属航空公司、确定航班是否满仓等);4. 订票: (订票情况可以存在一个数据文件中, 结构自己设定) 可以订票, 如果该航班已经无票, 可以提供相关可选择航班;5. 退票: 可退票, 退票后修改相关数据文件; 数据的读取存取及存储结构, 如何进行数据文件录入, 如何查询航班信息, 订票的实现和订票信息的存储, 附加有退票功能。6. 高级应用的拓展: 排序操作, 可以根据票价、时间等信息对相关航线 (如相同起飞、抵达城市航线) 进行排序;7. 合理设计线路: 输入两个城市名称, 可以根据条件 (如出发、到达时间) 设计这两座城市之间可能的线路, 进而推荐合理线路;8. 推荐最优化线路: 以机票价钱低作为最优量度标准, 设计两座城市之间最优线路并输出之; 解题思路: 利用线性表、图、队列和文件等数据结构知识进行存储航班、机场信息。先对数据文件里的数据进行读入, 选择合适的数据结构, 然后进行操作。 解题方法: 利用图对航班信息进行存储, 顶点代表机场, 边代表航线, 采用邻接链表的方式进行存储, 对文件的读写采用 json, 然后按照图的算法进行操作, 具体见数据结构选择、算法设计。		

任务分工及进度安排：

任务分工：

刘威：系统框架主要函数部分与测试、实验报告修改。

李晓宇：部分函数与测试、实验报告撰写修改。

李宇尧：部分函数与测试、实验报告撰写与修改。

进度安排：

12/25：基本完成类的构建和文件读写。

12/26：完成基本应用的部分。

12/27：完成高级应用的部分。

12/28：对程序进行调试修改美化，实验报告初步撰写。

12/29：程序的完善和实验报告完成。

数据结构选择、算法设计：

数据结构选择：线性表、图、堆栈和文件

算法设计：

1. 航班信息录入：采用 json 文件，将 excel 文件整理转化为 json 文件，然后进行读入，存储在创建的图中，采用邻接链表的存储形式。
2. 航班信息浏览：对已存储在图里的信息进行浏览，遍历每一个邻接顶点，依次输出每一条边（代表航线），并输出航线的基本信息，如航班号、起降机场、价格等信息。
3. 航班信息查询：如输入航班号，通过遍历所有航班，查询到匹配的航班并输出起、降时间，起飞、抵达城市，航班票价，票价折扣，所属航空公司、确定航班是否满仓（在此基础上，添加了别的方式进行航班查询，有起飞城市查询等）
4. 订票：增加了函数实现此功能，并写入本地文件，首先查找筛选出满足条件的航班，然后选择需要的航班进行预定，输入姓名和身份证进行预定，判断是否预定成功，并写入本地文件。并增加了查看订单情况的操作。（更新航班信息，当前人数减一）。
5. 退票：在订票的基础上实现此操作，首先进入退票系统，然后输入所要退订的航班信息，实现退票，并更新本地文件。（更新航班信息，当前

人数加一)。

6. 排序操作：按照题意我们进行两种排序，一类是按价格排序，一类是按时间进行排序。对于两类排序，我们都先筛选出符合条件的航班，然后用容器进行存放，再对其进行排序，对于按时间排序，我们将时间转化为可比较的数值进行比较，然后再进行排序，此方法可查询两个城市间的所有航线。
7. 合理设计线路：此问题描述是输入两个城市名称，可以根据条件（如出发、到达时间、进行筛选）设计这两座城市之间可能的线路，进而推荐合理线路。解决思路是首先对航线进行 BFS 算法，找到符合条件的路线，然后再次进行筛选，找到符合条件的合理路线。
8. 航线网络输出：本小题我们在上题的基础上，对合适的路线进行价格筛选，用链表的方式进行输出，输入一个城市，将输出该城市到别的其余城市的航线（选取当前最低票价航线）。
9. 推荐最优化路线：由于上小题已经找到合适的航线网络，本题在上小题的基础上进行搜索，找到合适的路线（机票价钱低作为最优度量）并输出。

编程与程序清单（仅给出关键代码和注释说明即可，不必粘贴全部源程序）：

//订票系统类（主体部分）

```
class System
{
public:
    System();
    virtual ~System();
    AirlineGraph* mAirlineGraph;
    vector<BookOrder*>* mBookOrderVector;
    void InsertAirlineInfo(); //插入航班信息
    void ShowAllAirlineToUser(); //预览已经建立的全部航线的相关信息
    void ShowAirlineByAirport(int no); //预览某个机场的航班
    void ShowAirlineByCity(string city); //预览某个城市的航班
    void SearchAirline(string name); //通过航班号查看详细信息
    void LoadBookOrder();
    int GetBookOrderNum();
    void Book(); //订票
    void ShowBookList();
    void Unsubscribe(); //退票
    void UnsubscribeByName(string name); //通过姓名退票
    void UnsubscribeByIdNum(string Id); //通过证件号退票
    void UnsubscribeByNo(int no); //通过序号退票
protected:
private:
    Array GenerateBookJson(); //生成json
};
```

//主要函数

```
void System::ShowMenu(int i)
{
    switch(i)
    {
    case 0:
        cout<<endl
        <<"1) 录入航线"<<endl
        <<"2) 浏览全部航线（!!! 慎用，4000多条，信息量略大!!!）"<<endl
        <<"3) 航班号查询航线"<<endl
        <<"4) 查看订单情况"<<endl
        <<"5) 订票"<<endl
        <<"6) 退票"<<endl
        <<"7) 查询城间航线"<<endl
        <<"8) 合理线路设计"<<endl
        <<"9) 航线网络"<<endl
        <<"10) 推荐最优线路"<<endl
        <<endl
        <<"0) 退出"<<endl;
        break;
    case 1:
        cout<<endl
        <<"请选择排序方式"<<endl
        <<"1) 出发时间"<<endl
        <<"2) 折后票价"<<endl
        <<"0) 退出"<<endl;
        break;
    case 2:
        cout<<endl
        <<"请输入出发城市和到达城市:"<<endl;
        break;
    case 3:
        cout<<endl
        <<"请输入航班号:"<<endl;
        break;
    case 4:
        cout<<endl
        <<"请输入出发城市、到达城市、出发时间和到达时间:"<<endl;
        break;
    case 5:
        cout<<endl
        <<"请输入出发城市:"<<endl;
        break;
    case 6:
        cout<<endl
        <<"请输入出发城市和到达城市:"<<endl;
        break;
    }
}
```

//航线图类（包含核心算法）

```
//航线图
class AirlineGraph
{
public:
    AirlineGraph();
    virtual ~AirlineGraph();
    int mAirportNumber; //机场数量
    Airport** mAirportHeadArray; //顶点表
    vector<Airline*>* mAirlineVector; //保存航线
    Airport* FindAirportByName(string name); //查找机场，获取指针
    int GetAirlineNumber(); //返回航线数目
    void InsertAirline(Airline* airline); //外部插入接口
    void ShowAllAirlineToUser(); //预览已经建立的全部航线的相关信息
    void ShowAirlineByAirport(int no); //预览某个机场的航班
    void ShowAirlineByCity(string city); //预览某个城市的航班
    void ShowAirlineGraph(); //打印输出边链表
    void WriteAirlineJson(); //写json到文件
    void Book(Airline* airline);
    void Unsubscribe(BookOrder* bookOrder);
    vector<int*>* GetAirportIdByLocation(string loc);
    string GetAirportLocation(string airportName);
    Airport* FindAirportByNo(int number);
    Airport* FindAirlineByName1(string name); //通过航班号查找
    Airport* FindAirportByCity(string city); //根据城市查询航班
    Airline* FindAirlineByDepartureAirport(string name); //根据出发城市查找
    void Search(string city1,string city2); //根据出发和目的地查找
    void SortByPrice(string name1,string name2); //按价格排序
    vector<Airline*>* FindAirlineByName(string name); //通过航班号查找
    void SortByTime(string name1,string name2); //按时间排序
    void SearchByTime(string time1,string time2,string name1,string name2); //按时间查找
    void SearchByPrice(int price1,int price2,string name1,string name2); //按票价查找
    void SearchCheapest(string city1,string city2);
    int timetransform(string time); //时间转换函数
    void Properline(string city1,string city2); //推荐合理路线
    void ShowBestAirline(); //推荐最优化线路
    bool timecomp(string time1,string time2);
protected:
private:
    Array GenerateAirlineJson(); //生成json
    void LoadAirport(); //从本地载入机场数据
    void LoadAirline(); //从本地载入航线数据
    void InsertAirlineGraph(Airport* airport,Airline* airline); //在图中插入边（插入航线）
};
```

//主要函数

```
//读入机场信息
void AirlineGraph::LoadAirport()
{
    Array AirportArray;
    ifstream infile;
    string s;
    infile.open("data/Airport.json");
    ostringstream tmp;
    tmp<<infile.rdbuf();
    s=tmp.str();
    AirportArray.parse(s); //解析json
    mAirportNumber=AirportArray.size();
    mAirportHeadArray=new Airport*[mAirportNumber];
    for(int i=0;i<mAirportNumber;i++) //生成顶点表
    {
        mAirportHeadArray[i]=new Airport();
        mAirportHeadArray[i]->No=AirportArray.get<Object>(i).get<Number>("序号");
        mAirportHeadArray[i]->mAirportName=AirportArray.get<Object>(i).get<String>("机场");
        mAirportHeadArray[i]->mShortName=AirportArray.get<Object>(i).get<String>("机场代号");
        mAirportHeadArray[i]->mLocation=AirportArray.get<Object>(i).get<String>("城市");
    }
    //cout<<AirportArray.json();
}
```

```

//读入航线信息
void AirlineGraph::LoadAirline()
{
    Array AirlineArray;
    ifstream infile;
    string s;
    infile.open("data/Airline.json");
    ostringstream tmp;
    tmp<<infile.rdbuf();
    s=tmp.str();
    AirlineArray.parse(s); //解析json
    mAirlineVector=new vector<Airline*>();
    for(int i=0;i<AirlineArray.size();i++) //保存航线到vector
    {
        Airline* airline=new Airline();
        airline->mAirlineName=AirlineArray.get<Object>(i).get<String>("航班号");
        airline->mCompany=AirlineArray.get<Object>(i).get<String>("公司");
        airline->mDepartureAirport=AirlineArray.get<Object>(i).get<String>("起飞机场");
        airline->mArrivalAirport=AirlineArray.get<Object>(i).get<String>("到达机场");
        airline->mDepartureTime=AirlineArray.get<Object>(i).get<String>("起飞时间");
        airline->mArrivalTime=AirlineArray.get<Object>(i).get<String>("到达时间");
        airline->mAirplaneModel=AirlineArray.get<Object>(i).get<String>("机型");
        airline->mDepartureCity=AirlineArray.get<Object>(i).get<String>("起始城市");
        airline->mArrivalCity=AirlineArray.get<Object>(i).get<String>("到达城市");
        airline->mPrice=AirlineArray.get<Object>(i).get<Number>("价格");
        airline->mIntDiscount=AirlineArray.get<Object>(i).get<Number>("最大折扣");
        airline->mCapacity=AirlineArray.get<Object>(i).get<Number>("满载");
        airline->mCurrentNumber=AirlineArray.get<Object>(i).get<Number>("当前人数");
        mAirlineVector->push_back(airline);

        Airport* dAirport=GetAirportByName(airline->mDepartureAirport);
        Airport* aAirport=GetAirportByName(airline->mArrivalAirport);
        if(dAirport!=NULL&&aAirport!=NULL) //判断机场是否存在
        {
            airline->mAirportNo=aAirport->No;
            InsertAirlineGraph(dAirport,airline); //插入到图
        }
    }
    infile.close();
    //cout<<AirlineArray.json();
}

```

//广度优先搜索 BFS

```

void AirlineGraph::BFS(int f,int a,int* InD,int* visit,vector<Route*> mainVec)
{
    int k=1; //参数k
    queue<Route> q;
    Route r;
    r.mPrevNo=f;
    q.push(r);
    while(!q.empty())
    {
        Route r0=q.front();
        q.pop();
        Airline* airline=mAirportHeadArray[r0.mPrevNo]->mAdjAirline;
        while(airline!=NULL)
        {
            if(!r0.CheckPass(airline->mArrivalAirport))
            {
                if(((r0.mAirlineVec.size()-1)>0&&r0.mAirlineVec[r0.mAirlineVec.size()-1]->GetAirlineArrivalTimeStamp()-<airline->GetAirlineDepartureTimeStamp())
                ||(r0.mAirlineVec.size()==0) //若不是始发航线则需要判断前后航班时间是否衔接上
                &&airline->GetAirlineDepartureTimeStamp()-<airline->GetAirlineArrivalTimeStamp()) //不衔接
                {
                    int no=GetAirportByName(airline->mArrivalAirport)->No;
                    if(visit[no]*k>InD[no]) //入度的k倍，经过一个点是入度的10倍，决定BFS精密程度，但是运行时间会增大
                    {
                        Route rNew=r0;
                        rNew.mAirlineVec.push_back(airline);
                        rNew.mPrevNo=no;
                        visit[no]+=1;
                        if(no!=a)
                        {
                            q.push(rNew);
                        }
                        else
                        {
                            mainVec->push_back(rNew);
                        }
                    }
                }
                airline=airline->mNextAirline;
            }
            airline=airline->mNextAirline;
        }
    }
}

```

深度优先搜索 DFS

```
void    AirlineGraph::DFS(int    v,int    a,int*    InD,int*
visit,vector<    vector<Airline*>    >*    mainVec,vector<Airline*>
routeVec)
{
    if(v!=a)    //未到达目的地
    {
        visit[v]+=1;

        Airline*    airline=mAirportHeadArray[v]->mAdjAirline;
        while(airline!=NULL)
        {

            int    no=airline->mAirportNo;

            bool    tag=0;

            for(int    i=0;i<routeVec.size();i++)
            {

if(routeVec[i]->mArrivalAirport==airline->mArrivalAirport)

                {

                    tag=1;

                    break;

                }

            }

        }
    }
```

```

        if(routeVec.size()==0)
        {
            if(visit[no]<InD[no]&&!tag)    //比较访问次
数，检测是否小于入度
            {
                vector<Airline*> newRoute;
                for(vector<Airline*>::iterator
it=routeVec.begin(); it!=routeVec.end(); it++)
                {
                    newRoute.push_back(*it);
                }
                newRoute.push_back(airline);
                DFS(no, a, InD, visit, mainVec, newRoute);
            }
        }
        else
        if(routeVec[routeVec.size()-1]->GetAirlineArrivalTimeStamp(
)<airline->GetAirlineDepartureTimeStamp()/*&&airline->GetAi
rlineDepartureTimeStamp()<airline->GetAirlineArrivalTimeSta
mp()*/)
        {
            if(visit[no]<InD[no]&&!tag)    //比较访问次

```


数，检测是否小于入度

```
{  
  
    vector<Airline*> newRoute;  
    for(vector<Airline*>::iterator  
it=routeVec.begin(); it!=routeVec.end(); it++)  
    {  
        newRoute.push_back(*it);  
    }  
    newRoute.push_back(airline);  
    DFS(no, a, InD, visit, mainVec, newRoute);  
}  
}  
    airline=airline->mNextAirline;  
}  
}else    //到达目的地，终止 DFS  
{  
    visit[v]+=1;  
    mainVec->push_back(routeVec);    // 将 路 径 保 存 至  
mainVec  
}  
}  
  
//Dijkstra 算法
```

```

Route** AirlineGraph::Dijkstra(int v)
{
    int TotalCost[mAirportNumber];
    int path[mAirportNumber];
    bool visit[mAirportNumber];
    for(int i=0; i<mAirportNumber; i++)
    {
        TotalCost[i]=INT_MAX;
        path[i]=-1;
        visit[i]=0;
    }
    TotalCost[v]=0;
    visit[v]=1;
    Airline* airline=mAirportHeadArray[v]->mAdjAirline;
    int u=v;
    for(int i=0; i<mAirportNumber-1; i++)
    {
        while(airline!=NULL)    //更新长度、路径信息
        {
            int k=airline->mAirportNo;

if(visit[k]!=1&&TotalCost[k]+airline->GetPriceAfterDiscount
()<TotalCost[k])
{    TotalCost[k]=TotalCost[k]+airline->GetPriceAfterDiscou
nt();
            path[k]=u;
        }
        airline=airline->mNextAirline;
    }
    int tmpMin=INT_MAX;
    for(int j=0; j<mAirportNumber; j++) //决定下一被访问
    结点
    {
        if(TotalCost[j]<tmpMin&&visit[j]==0)
        {
            tmpMin=TotalCost[j];
            u=j;
        }
    }
}

```

```

    }
    visit[u]=1;
    airline=mAirportHeadArray[u]->mAdjAirline;
}

Route** routeArray=new Route*[mAirportNumber];

/*for(int i=0;i<mAirportNumber;i++)
    cout<<"i\t"<<i<<"\t"<<path[i]<<endl;*/
for(int i=0;i<mAirportNumber;i++)
{
    if(path[i]!=-1) //是 v 本身或没有可及路径
    {
        stack<int> s;
        int j=i;
        while(j!=v)    //回溯路径，压栈
        {
            s.push(j);
            j=path[j];
        }
        int prev=v;
        Route* r=new Route();
        while(!s.empty())    //弹栈，生成路径
        {
            int f=s.top();
            Airline* airline=GetMinCostAirline(prev,f);
            r->mAirlineVec.push_back(airline);
            s.pop();
            prev=f;
        }
        routeArray[i]=r;
    }else
    {
        routeArray[i]=NULL;
    }
}

return routeArray;
}

```

测试数据：约 124 个机场，4000 多条航班数据。以下是部分数据截图。

[illegible]

订单信息

- 1) 录入航线
- 2) 删除航线
- 3) 浏览全部航线 (!!! 慎用, 4000多条, 信息量略大!!!)
- 4) 航班号查询航线
- 5) 查看订单情况
- 6) 订票
- 7) 退票
- 8) 查询城间航线
- 9) 合理线路设计
- 10) 航线网络
- 11) 推荐最优线路
- 0) 退出

1、录入航线

请输入航空公司：
南方航空
请输入航班号：
CA9999
请输入起飞时间：
6:00
请输入到达时间：
8:00
请输入起飞机场：
首都国际机场
请输入到达机场：
浦东国际机场
请输入机型：
A720
请输入容量：
200
请输入当前乘客人数：
150
请输入票价：
1500
请输入折扣（1000%）：
150

2、删除航线

航班CA9999录入成功！

- 1) 录入航线
- 2) 删除航线
- 3) 浏览全部航线（!!! 慎用，4000多条，信息量略大!!!）
- 4) 航班号查询航线
- 5) 查看订单情况
- 6) 订票
- 7) 退票
- 8) 查询城间航线
- 9) 合理线路设计
- 10) 航线网络
- 11) 推荐最优线路

0) 退出

2

请输入航班号：
CA9999

航班CA9999已删除！

3、浏览全部航线（由于数据量过大，以及输出时间过长，以下给出部分截图）

CA1339	中国国际航空公司	北京	首都国际机场	广州	新白云国际机场	12:00	15:00	738
2872	0.2 2297.6	220	215	5				
CZ5052	中国南方航空公司	北京	首都国际机场	昆明	巫家坝国际机场	12:00	15:10	763
2661	0.23 2048.97	290	280	10				
CZ5008	中国南方航空公司	北京	首都国际机场	南京	禄口国际机场	12:00	13:50	321
2938	0.1 2644.2	240	231	9				
CA4114	中国国际航空公司	北京	首都国际机场	成都	双流国际机场	12:00	14:45	JET
1728	0.2 1382.4	220	214	6				
CZ5110	中国南方航空公司	北京	首都国际机场	上海	虹桥国际机场	12:00	14:05	333
2458	0.2 1966.4	190	185	5				
CA1704	中国国际航空公司	北京	首都国际机场	杭州	萧山国际机场	12:00	14:05	319
2282	0.5 1141	250	242	8				
CA1583	中国国际航空公司	北京	首都国际机场	南京	禄口国际机场	12:00	13:50	319
1567	0.4 940.2	110	103	7				
CA1347	中国国际航空公司	北京	首都国际机场	汕头	外砂机场	12:05	14:50	738
2874	0.299 2014.67	120	113	7				
CA1323	中国国际航空公司	北京	首都国际机场	珠海	三灶机场	12:10	15:25	738
2293	0.359 1469.81	240	230	10				
CA1457	中国国际航空公司	北京	首都国际机场	广元	广元机场	12:10	14:20	73G
2699	0.299 1892	220	211	9				
CA1325	中国国际航空公司	北京	首都国际机场	郑州	新郑机场	12:10	13:30	738
2276	0.5 1138	140	136	4				
CZ3102	中国南方航空公司	北京	首都国际机场	广州	新白云国际机场	12:15	15:05	77A
3329	0.1 2996.1	180	172	8				
CA1545	中国国际航空公司	北京	首都国际机场	烟台	莱山机场	12:15	13:25	738
3386	0.2 2708.8	110	100	10				
CA4162	中国国际航空公司	北京	首都国际机场	贵阳	龙洞堡机场	12:15	15:15	733
2921	0.4 1752.6	230	225	5				
CA4172	中国国际航空公司	北京	首都国际机场	昆明	巫家坝国际机场	12:20	15:45	738
2691	0.299 1886.39	180	177					

4、航班号查询航线（选取 2 中第一条数据 CA1339 查询）

请输入航班号：
CA1339

共有1个结果：

航班号： CA1339
航空公司： 中国国际航空公司
出发地： 北京
起飞机场： 首都国际机场
目的地： 广州
着陆机场： 新白云国际机场
起飞时间： 12:00
抵达时间： 15:00
机型： 738
票价： 2872
折扣： 0.2
折后票价： 2297.6
载客量： 220
已售： 215
余票： 5

类比二中的数据完全正确！

5、订单情况查询（打开 Book.json 文件，输出订单信息，以下给出部分截图）

【1】

姓名： Zhao
证件号： 3546845313543
航班号： CZ3860
航空公司： 中国南方航空公司
出发地： 重庆
起飞机场： 江北国际机场
目的地： 汕头
着陆机场： 外砂机场
起飞时间： 12:20
抵达时间： 15:25
机型： 733
购买价格： 2221

【2】

姓名： Huang
证件号： 669443545313548646315
航班号： CA1883
航空公司： 中国国际航空公司
出发地： 北京
起飞机场： 首都国际机场
目的地： 上海
着陆机场： 浦东国际机场
起飞时间： 7:55
抵达时间： 10:05
机型： 738
购买价格： 1927

【3】

姓名： Wang
证件号： 864654153435768453
航班号： C76541

6、订票(无航班或者满载时均有提示，选用 2 中 CA1339 航班)

请输入航班号：
CA1339
=====

航班号：CA1339
航空公司：中国国际航空公司
出发地：北京
起飞机场：首都国际机场
目的地：广州
着陆机场：新白云国际机场
起飞时间：12:00
抵达时间：15:00
机型：738
票价：2872
折扣：0.2
折后票价：2297.6
载客量：220
已售：215
余票：5

===== 预订成功 =====
姓名：LIXY
证件号：332123201705054341
航班号：CA1339
航空公司：中国国际航空公司
出发地：北京
起飞机场：首都国际机场
目的地：广州
着陆机场：新白云国际机场
起飞时间：12:00
抵达时间：15:00
机型：738
购买价格：2297

=====

请输入姓名：
LIXY

=====

请输入证件号：
332123201705054341
=====

预定成功后程序会修改航班当前人数

7、退票 (以 5 中输入为例子,截图仅提供按照姓名退票)

=====

【8】

姓名：LIXY
证件号：332123201705054341
航班号：CA1339
航空公司：中国国际航空公司
出发地：北京
起飞机场：首都国际机场
目的地：广州
着陆机场：新白云国际机场
起飞时间：12:00
抵达时间：15:00
机型：738
购买价格：2297

=====

请输入姓名：
LIXY
=====

===== 退票成功 =====

姓名：LIXY
证件号：332123201705054341
航班号：CA1339
航空公司：中国国际航空公司
出发地：北京
目的地：广州
购买价格：2297

=====

【1】通过姓名退票
【2】通过证件号退票
【3】通过以上序号退票

8、查询城间航线并排序 (时间、票价) 以长春——北京的航班为例
1)按起飞时间排序

请选择排序方式
1) 出发时间
2) 折后票价
0) 退出
1

请输入出发城市和到达城市：
长春 北京

航班号	航空公司	出发地	起飞机场	目的地	着陆机场	起飞时间	抵达时间	机型	票价	折扣	折后票价	载客量	已
CZ6143	中国南方航空公司	长春	长春龙嘉国际机场	北京	首都国际机场	7:50	9:40	320	2699	0.23	2078.23	100	96
CA1662	中国国际航空公司	长春	长春龙嘉国际机场	北京	首都国际机场	8:00	9:50	JET	3032	0.1	2728.8	280	27
CZ6145	中国南方航空公司	长春	长春龙嘉国际机场	北京	首都国际机场	8:40	10:25	321	3176	0.4	1905.6	290	28
CA1610	中国国际航空公司	长春	长春龙嘉国际机场	北京	首都国际机场	10:45	12:35	738	2288	0.4	1372.8	250	24
CA1650	中国南方航空公司	长春	长春龙嘉国际机场	北京	首都国际机场	12:10	13:50	JET	1919	0.299	1345.22	270	26
CA1664	中国国际航空公司	长春	长春龙嘉国际机场	北京	首都国际机场	14:45	16:35	738	1695	0.4	1017	180	17
CZ6149	中国南方航空公司	长春	长春龙嘉国际机场	北京	首都国际机场	16:00	18:00	321	3157	0.5	1578.5	280	27
CA1630	中国国际航空公司	长春	长春龙嘉国际机场	北京	首都国际机场	18:10	19:55	JET	2300	0.1	2070	290	28
CZ6141	中国南方航空公司	长春	长春龙嘉国际机场	北京	首都国际机场	18:30	20:00	319	2904	0.33	1945.68	160	15
CZ6179	中国南方航空公司	长春	长春龙嘉国际机场	北京	首都国际机场	20:20	22:15	319	2547	0.1	2292.3	250	24
CA1648	中国国际航空公司	长春	长春龙嘉国际机场	北京	首都国际机场	21:30	23:25	JET	1400	0.1	1260	220	21

2)按折后票价排序

请选择排序方式

- 1) 出发时间
- 2) 折后票价
- 0) 退出

2

请输入出发城市和到达城市:

长春 北京

航班号	航空公司	出发地	起飞机场	目的地	着陆机场	起飞时间	抵达时间	机型	票价	折扣	折后票价	载客量	已
CA1664	中国国际航空公司	长春	长春龙嘉国际机场	北京	首都国际机场	14:45	16:35	738	1695	0.4	1017	180	17
CA1648	中国国际航空公司	长春	长春龙嘉国际机场	北京	首都国际机场	21:30	23:25	JET	1400	0.1	1260	220	21
CA1650	中国国际航空公司	长春	长春龙嘉国际机场	北京	首都国际机场	12:10	13:50	JET	1919	0.299	1345.22	270	26
CA1610	中国国际航空公司	长春	长春龙嘉国际机场	北京	首都国际机场	10:45	12:35	738	2288	0.4	1372.8	250	24
CZ6149	中国南方航空公司	长春	长春龙嘉国际机场	北京	首都国际机场	16:00	18:00	321	3157	0.5	1578.5	280	27
CZ6145	中国南方航空公司	长春	长春龙嘉国际机场	北京	首都国际机场	8:40	10:25	321	3176	0.4	1905.6	290	28
CZ6141	中国南方航空公司	长春	长春龙嘉国际机场	北京	首都国际机场	18:30	20:00	319	2904	0.33	1945.68	160	15
CA1630	中国国际航空公司	长春	长春龙嘉国际机场	北京	首都国际机场	18:10	19:55	JET	2300	0.1	2070	290	28
CZ6143	中国南方航空公司	长春	长春龙嘉国际机场	北京	首都国际机场	7:50	9:40	320	2699	0.23	2078.23	100	96
CZ6179	中国南方航空公司	长春	长春龙嘉国际机场	北京	首都国际机场	20:20	22:15	319	2547	0.1	2292.3	250	24
CA1662	中国国际航空公司	长春	长春龙嘉国际机场	北京	首都国际机场	8:00	9:50	JET	3032	0.1	2728.8	280	27

9、合理路线设计（两个城市名称，可以根据条件（如出发、到达时间）设计这两座城市之间可能的线路，进而推荐合理线路）以长春到北京 6：00——12：00 为例，提供中转方案。

请输入出发城市、到达城市、出发时间和到达时间：
长春 北京 6:00 12:00

总费用：1905 [北京-首都国际机场 CZ6145]
总费用：2078 [北京-首都国际机场 CZ6143]
总费用：2728 [北京-首都国际机场 CA1662]
总费用：4353 [大连-周水子机场 CZ637]
总费用：5122 [大连-周水子机场 CZ637]
->[北京-首都国际机场 CZ6139]
->[北京-首都国际机场 CA1698]

10、航线网络输出（输入一个城市，将输出这个城市到别的所有城市的路线）以城市作为顶点，航线作为边（若两个城市之间具有多条线路，则选取当前最低票价航线）。

从【长春 - 长春龙嘉国际机场】到:

【腾冲 - 驼峰机场】	总费用：2509	【重庆-江北国际机场 CZ6459】	->【腾冲-驼峰机场 CZ8169】
【运城 - 运城关公机场】	总费用：2629	【广州-新白云国际机场 CZ6341】	->【运城-运城关公机场 CA1392】
【长白山 - 长白山机场】	总费用：1966	【延吉-朝阳川机场 CZ6805】	->【长白山-长白山机场 CZ6393】
【阿克苏 - 阿克苏机场】	总费用：4785	【南京-禄口国际机场 CZ6545】	->【乌鲁木齐-地窝堡国际机场 CZ6964】->【阿克苏-阿克苏机场 CZ6863】
【阿勒泰 - 阿勒泰机场】	总费用：5963	【南京-禄口国际机场 CZ6545】	->【乌鲁木齐-地窝堡国际机场 CZ6964】->【阿勒泰-阿勒泰机场 CZ6841】
【安庆 - 安庆机场】	总费用：2971	【广州-新白云国际机场 CZ6341】	->【安庆-安庆机场 CZ3585】
【包头 - 包头机场】	总费用：2797	【包头-包头机场 CZ6239】	
【保山 - 云瑞机场】	总费用：4115	【广州-新白云国际机场 CZ6341】	->【保山-云瑞机场 CZ3477】
【北海 - 福城机场】	总费用：5422	【南京-禄口国际机场 CZ6545】	->【桂林-两江国际机场 CZ3254】 ->【三亚-凤凰国际机场 CZ6399】 ->【北海-福城机场 CZ2720】
【北京 - 首都国际机场】	总费用：1017	【北京-首都国际机场 CA1664】	
【北京 - 南苑机场】	总费用：5077	【南京-禄口国际机场 CZ6545】	->【桂林-两江国际机场 CZ3254】 ->【三亚-凤凰国际机场 CZ6399】 ->【北京-南苑机场 FM2278】
【常德 - 桃花源机场】	总费用：3003	【广州-新白云国际机场 CZ6341】	->【常德-桃花源机场 CZ3229】
【常州 - 奔牛机场】	总费用：2613	【广州-新白云国际机场 CZ6341】	->【常州-奔牛机场 CZ3815】
【成都 - 双流国际机场】	总费用：938	【成都-双流国际机场 CZ6441】	
【赤峰 - 赤峰机场】	总费用：2880	【北京-首都国际机场 CA1664】	->【赤峰-赤峰机场 CA1281】
【达县 - 达县机场】	总费用：4168	【广州-新白云国际机场 CZ6341】	->【达县-达县机场 CA4356】
【大理 - 大理机场】	总费用：3409	【重庆-江北国际机场 CZ6459】	->【景洪-西双版纳机场 CZ8137】 ->【大理-大理机场 8L9961】
【大连 - 周水子机场】	总费用：1740	【大连-周水子机场 CZ6652】	
【大庆 - 大庆萨尔图机场】	总费用：6355	【南京-禄口国际机场 CZ6545】	->【桂林-两江国际机场 CZ3254】 ->【三亚-凤凰国际机场 CZ6399】 ->【大庆-大庆萨尔图机场 CZ6725】
【大同 - 大同机场】	总费用：3201	【广州-新白云国际机场 CZ6341】	->【大同-大同机场 CZ3705】
【丹东 - 浪头机场】	总费用：5504	【南京-禄口国际机场 CZ6545】	->【桂林-两江国际机场 CZ3254】 ->【三亚-凤凰国际机场 CZ6399】 ->【丹东-浪头机场 CZ6755】
【德宏 - 德宏芒市机场】	总费用：3468	【昆明-巫家坝国际机场 CZ6239】	->【德宏-德宏芒市机场 8L9931】
【迪庆 - 迪庆香格里拉机场】	总费用：2428	【昆明-巫家坝国际机场 CZ6239】	->【迪庆-迪庆香格里拉机场 8L9991】
【鄂尔多斯 - 鄂尔多斯机场】	总费用：3666	【重庆-江北国际机场 CZ6459】	->【鄂尔多斯-鄂尔多斯机场 3U8869】
【恩施 - 许家坪机场】	总费用：2353	【广州-新白云国际机场 CZ6341】	->【恩施-许家坪机场 CZ6589】
【福州 - 长乐机场】	总费用：3003	【南京-禄口国际机场 CZ6545】	->【福州-长乐机场 CZ3825】
【贵阳 - 龙洞堡机场】	总费用：4599	【广州-新白云国际机场 CZ6341】	->【贵阳-龙洞堡机场 CZ6915】
【赣州 - 黄金机场】	总费用：3311	【广州-新白云国际机场 CZ6341】	->【赣州-黄金机场 CZ3637】
【广元 - 广元机场】	总费用：2908	【北京-首都国际机场 CA1664】	->【广元-广元机场 CA1457】
【广州 - 新白云国际机场】	总费用：1413	【广州-新白云国际机场 CZ6341】	
【贵阳 - 龙洞堡机场】	总费用：2816	【南京-禄口国际机场 CZ6545】	->【贵阳-龙洞堡机场 CZ3656】
【桂林 - 两江国际机场】	总费用：2200	【南京-禄口国际机场 CZ6545】	->【桂林-两江国际机场 CZ3254】
【哈尔滨 - 太平机场】	总费用：2434	【南京-禄口国际机场 CZ6545】	->【哈尔滨-太平机场 3U8839】
【哈密 - 哈密机场】	总费用：5141	【南京-禄口国际机场 CZ6545】	->【乌鲁木齐-地窝堡国际机场 CZ6964】->【哈密-哈密机场 CZ6689】

11、推荐最优线路（以机票价钱低作为最优量度标准，输入出发到达城市，输出最优线路）以长春到北京为例。

请输入出发城市和到达城市：
长春 北京

总费用：1017 [北京-首都国际机场 CA1664]

程序的使用说明：

首先，编译运行进入主界面，或直接打开可执行文件。

然后，按照提示输入相应编号进行操作，基本实现了大部分数据的处理。

是一个不错的飞机订票系统，能实现航班的录入与删除，航班信息浏览，查询，订票退票等基本功能。并且还有一系列高级应用，例如：排序、合理设计线路、输出、推荐最优线路等功能，基本涵盖了飞机订票过程中的基本问题。并且数据量够多，测试起来没有发现异常。

总结（包括小组总结和每位组员针对自己分工的总结）：

（对程序进行分析、评价运行效果，总结遇到的问题及解决办法）

小组总结：对程序的分析：这次的飞机订票管理系统，结合了数据结构的多方面知识，运用了图等数据结构来解决问题，程序较为复杂，但是有很精炼。运行效果达到了预期目标，能够顺利的解决一系列问题，以及程序代码有时候不理解。遇到的问题是：1）有些地方存在着不同的设计思路与算法，2）还有对某一个问题的理解有不同的想法，3）以及程序代码有时候不理解，4）如何进行团队项目。针对这些问题，1）我们组选取合适的数据结构与算法，对待不够优质的算法进行淘汰，选取更加有效的算法，2）而对待一个问题有不同的想法时，我们进行小组内部讨论，并结合生活实际，对这个问题得到一致的想法，然后再着手解决问题。3）对待代码不理解的问题，我们加以注释，并互相询问。4）我们学习了 git，把小组项目传到 github 进行操作。

成员总结：

--

注：各部分内容要求填写详尽，如空间不够可自行扩充。