实验报告

一、完成度说明

基本功能全部实现

二、详细说明

（一）、 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

1.数据结构：

使用邻接表存储，且直接使用机场ID作为下标访问，所以时间复杂度为O(n+e)。因为是有向图，所以空间复杂度为O(n+e)

2.算法：

基本思想：深度优先遍历用递归实现，由于是从某一特定顶点出发，所以只要求该顶点所在的连通分量即可。广度优先遍历用队列实现，利用队列先进先出的特点。

时间复杂度：由于用邻接表作为存储结构，深度优先遍历查找邻接点所需时间为O(e)，所以时间复杂度为O(n+e)。广度优先同理，也是O(n+e)。

（二）、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

1.数据结构：

使用邻接矩阵存储，时间复杂度为O(e)，空间复杂度为O(n^2)

2.算法：

基本思想：（这里认为中转次数为k是限制必须中转k次）根据离散数学的知识，邻接矩阵的k次幂（其中加法使用逻辑加）得到的矩阵就可以标识两个机场是否能在中转k次时联通，所以直接求邻接矩阵的幂并输出即可。而对于没有中转次数限制的情况，调用1中的深度优先遍历得到所有可以到达的点，将其在邻接矩阵中的对应位置置为1即可。

时间复杂度：对于有限制的情况，调用矩阵乘法，时间复杂度为O(n^3)；对于无限制的情况，调用深度优先遍历，但还要遍历邻接矩阵进行输出，时间复杂度为O(n^2)

（三）、求任意两个机场之间的最短飞行时间（含转机停留时间）

1.数据结构：

用map类型的数组实现了邻接表,还用到了STL中的set容器，以及优先队列。map对应的结构是红黑树，其查找、插入、删除等操作的时间复杂度为O(logn)，二叉堆时间复杂度为O(n)。使用map构建邻接表，要执行n+e次查找、插入等操作，所以空间复杂度为O(n+e)，时间复杂度为O((n+e)\*logn)

2.算法：

基本思想：本题需要考虑时序关系，所以拆点建图，新的节点类型包括机场ID和所处时刻。用map类型数组和nextedge类建立邻接表存储，可以实现用结点直接访问对应的边而不必查找。Dijikstra算法采用优先队列来优化，以其与源点（出发点）的距离为排序依据建立优先队列，那么队首元素就是权值最小的元素，省去了查找的时间。由于出发点有多个出发时刻，所以for循环进行遍历，找到所有出发时刻中到达目的顶点的最短用时。

时间复杂度：每次从堆中取出的最小值就是要用的顶点，这样堆中元素共O(n)个，更新和取出的操作有O(e)次，因此时间复杂度为O(e\*logn)

（四）、仅限直飞或1次中转，求任意两个机场的航线

1.数据结构：

直飞的情况一边读取一边输出，不涉及数据结构，空间复杂度为O(1)，时间复杂度为O(n)。一次中转的情况使用优先队列存储，空间复杂度为O(n)，时间复杂度为O(n\*logn)

2.算法：

基本思想：对于直飞的情况，直接在读入表格时判断出发地和目的地是否满足要求，满足要求则直接输出即可。对于一次中转的情况，先用优先队列存下所有航班（按照起飞时间早的优先）然后从队首元素开始依次出队，直至找到从给定出发地出发的航班为止，将其记为a，复制该优先队列，在第二层循环中，先将队列元素依次出队直至出发时间晚于a的出发时间，然后寻找出发地是a的目的地，且目的地为给定目的地的航班b。由于该队列是时间优先，此时航班ab组成的航班既满足要求，又在时间上合法，是答案之一。重复这个过程得到所有航班。

时间复杂度：由于共有两层循环嵌套，每次从堆中取出的最小值的时间复杂度为O(logn)，所以时间复杂度为O((n\*logn)^2)

（五）、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

1.数据结构：

存储结构与问题3相同，不再赘述，空间复杂度为O(n+e)，时间复杂度为O((n+e)\*logn)

2.算法：

基本思想：本题考虑时序关系，拆点建图，存储结构与问题3相同，只是在读取表格数据时就利用所给的不同的限制条件进行读取，只读取符合要求的。（这里认为起飞时段和降落时段的限制是对所有航班的，也包括中转过程中的航班）。在寻找航线是否存在是，由于只需要找到一条航线即可，所以用广度优先搜索（BFS）从指定出发地开始搜索，直到第一次找到目的地即可跳出循环，比较其中转次数是否满足所给条件，满足才输出路径。

时间复杂度：使用队列进行广度优先遍历，时间复杂度为O(n+e)。

（六）、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.数据结构：

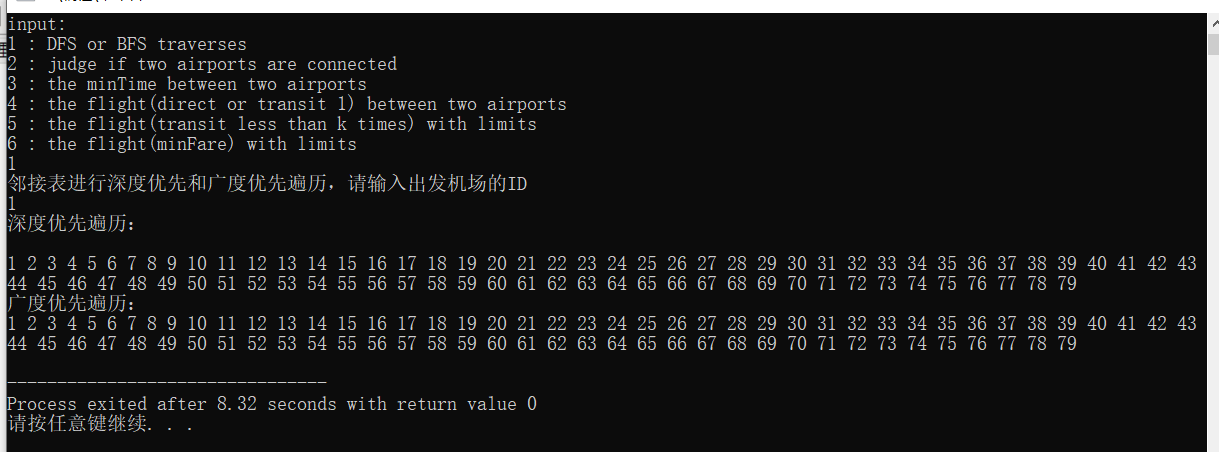
存储结构与问题3相同，不再赘述，空间复杂度为O(n+e)，时间复杂度为O((n+e)\*logn)

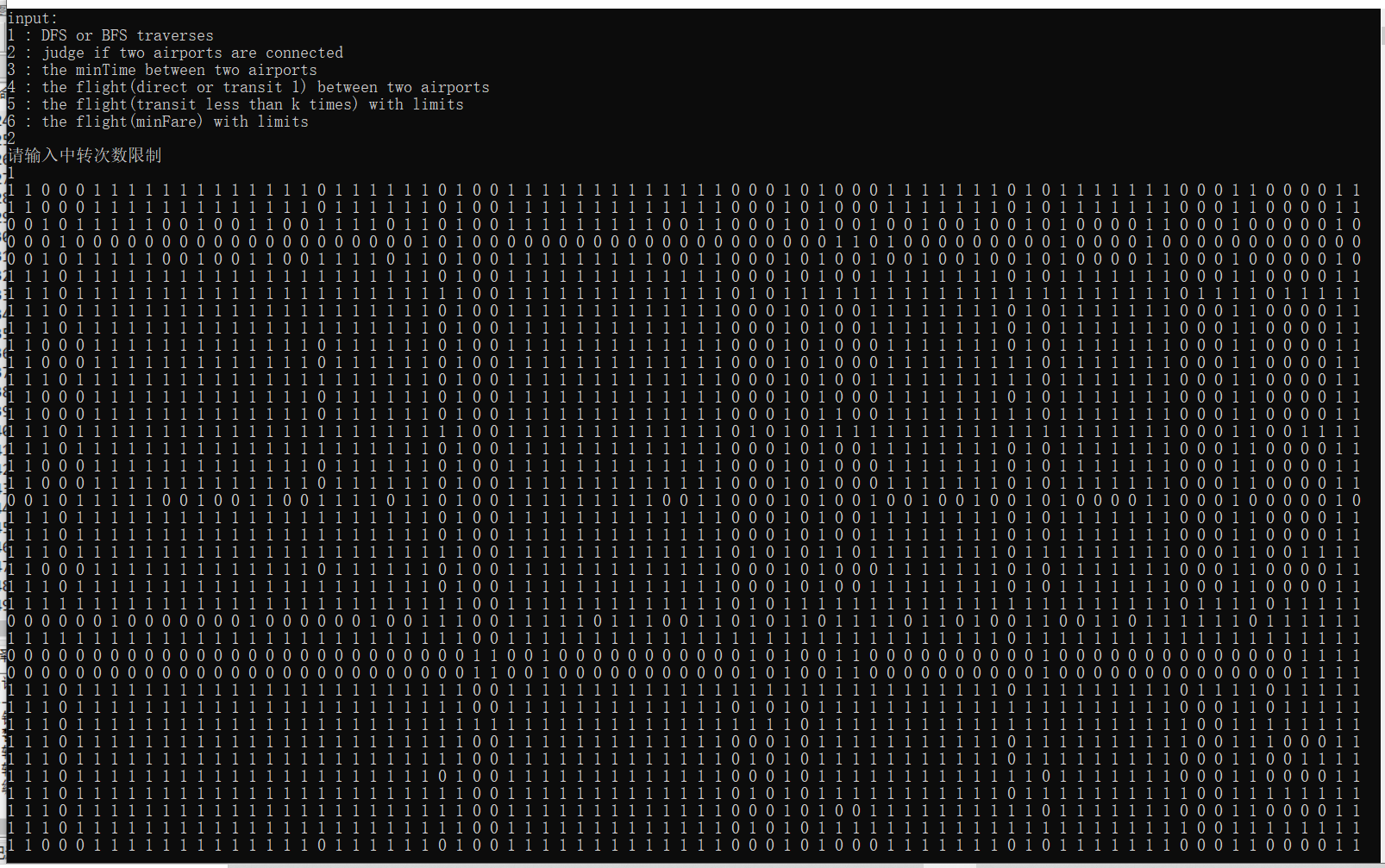
2.算法：

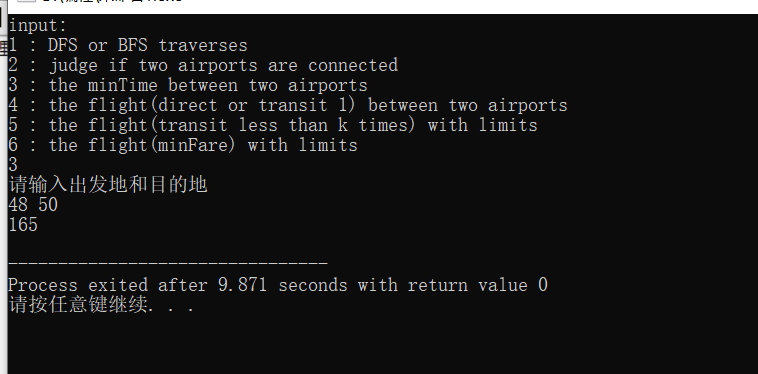
基本思想：图的读取、存储都与问题5相同，共用一个函数进行读取。搜索航费最低的路径，实质还是用dijkstra算法，只是权值变成了航费，同一机场的不同时刻对应的不同点之间权值为0。dijikstra算法可以套用问题3中的，但为了输出路径，另添map结构的path数组，来存放路径以便输出。

时间复杂度：使用的算法与问题3相同，所以不再重复分析过程，时间复杂度为O(e\*logn)

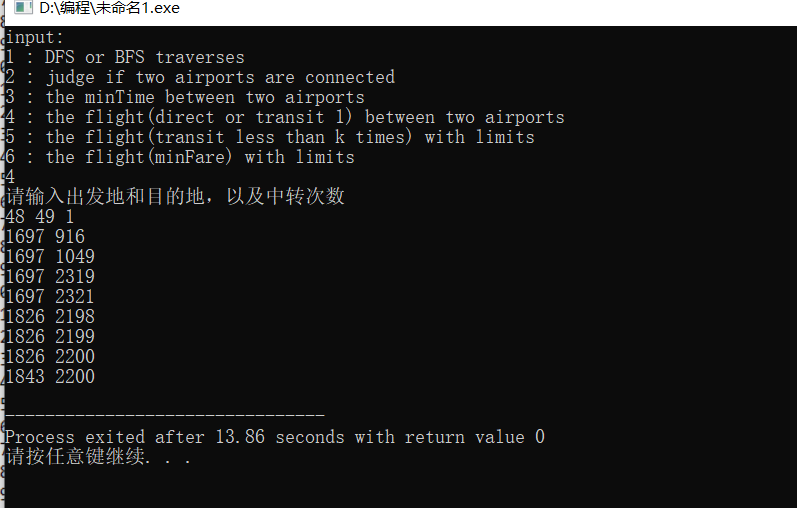
三、样例输出

1. 

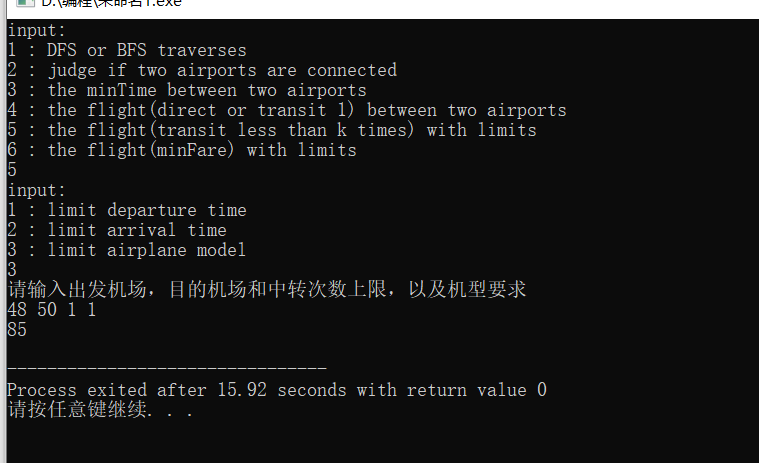
2. 

3. 

4.



5.



6.

