**第七题：交通咨询系统**

**需求分析与概要设计**

**原有需求分析：**

根据题意，题目要求设计出一个这样的交通咨询系统：用户通过输入起始地点和终止地点以及自己的出发时间，可以得到最快到达时间和相应的旅费以及最少到达费用以及相应的时间。

①**城市信息存储**

**分析**：首先肯定要先存储城市的信息。对于城市的信息，我们觉得可以用向量（vector），＋类来解决，分别定义城市、飞机和列车三个类，城市类中包含城市的名称、该城市的地方性特色（因为该系统主要面向游客）；而城市与城市之间的距离强调的是两者之间的关联，保存在文件当中（按照 （A,B）：xxxmeter， 的方式分行存储），后续存储作为图的边。

**设计**：将城市的信息和城市与城市之间的距离存储到文件 city.txt 和文件 citylink.txt 中，在程序中建立city的vector数组，通过 ifsrtream 将文件中的城市信息存储到 vector中，利用 vector 的 size（）算法得到当前向量的长度，根据当前的长度建立邻接矩阵，矩阵每个点存储一个值，代表城市与城市之间的距离，点与点之间边的信息通过读取citylink.txt 文件来确定。

**说明**：这里用vector数组存储，是为了防止定义一个长度较大（如500）的数组会浪费空间，或者定义一个长度过小的数组，添加信息超出了他的限度；城市与城市之间的信息一定是每个城市都有的，所以这里的邻接矩阵每个点都一定会有值，不会有无穷大的情况。

实践后设计：city类中不需要使用图来保存，直接使用无序表即可解决

②**交通工具信息存储**

**分析**：接下来要存储的就是交通工具的信息。由于交通工具反映的也是城市与城市之间的关联，所以火车和列车的信息也存储到文件当中。

火车的信息按照 k152：（A,B）xxxtime1,xxxhour，xxxtime2,xxxmoney，xxxmeter，（B,C）xxxtime1,xxxhour，xxxtime2，xxxmoney,xxxmeter，（C,D）xxxtime1,xxxhour，xxxtime2，xxxmoney ，xxxmeter，这里第一个xxxtime1表示在该出发地的起始时间，之后的xxxhour表示两个城市之间的通行时间，这里xxxtime2表示到达某短程终点站的时间，每个xxxmoney表示两个站之间的通行价格，并且以换行的方式的形式存储，我们根据生活中的经验火车在每个站停留的时间大概在10分钟左右，我们简化问题，每个站都停留10分钟，所以xxxtime2（本站）=xxxtime1（本站）+xxxhour ，xxxtime1（下一个站的）=xxxtime2（上一个站的）+10min 。

飞机的信息按照 CZ3823：（A,B）xxxtime1,xxxhour,xxxtime2,xxxmoney，基于生活的经验，飞机一般是不会有中点站的，对于现实生活中的中途休息，在这里我们不做考虑。

经过上述的分析，我们发现了它不同于上面城市之间距离的点，在这里，两个城市之间可以有多种交通方式，也就是如果用一条边来连接，他会有多个权值，这个时候我们经过思考，认为还是可以用邻接矩阵来解决。

**设计**：此时，矩阵中每个点（即边信息）存储的应该是一个向量 vector ，即可以包含两个城市之间的多个交通信息，而 vector 中每个元素是 unordered\_map 类型的，每个元素即交通工具编号和过程两个信息（可变动），需要注意，交通工具编号是键，而值还需要在建立一个vector(5)，用来存储上面说的5种信息。

③**最短时间与最短路费**

**分析**：最优决策的关键是利用Dijkstra算法求出该地点到另外一个地点的在某种标准下的最小值，所以最短路程和最短时间的区别就在于选谁作为标准。但还有一个麻烦的在于，最小的值会因为游客的需求有所不同，如果游客想要直达而不想中转，或者游客接受中转，并且游客接受中转时是否接受换乘不同的交通工具，他们算出来的最小值是会有区别的。而不论是否中转，我们考虑到我们日常生活中使用高德地图、百度地图这些软件，当我们输入出发地和目的地时，他是会提供两地之间所有的到达方式的，所以下面的设计思路我们会根据游客的不同需求，对可行方式进行筛选与排序并同时输出最优的出行方式。

**设计**：首先游客需要输入出发的时间（精确到某一天即可），如果游客想显示直达的信息，直接对文件信息处理即可，首先筛选出满足出发时间的交通工具。如果是飞机，只用关注其出发地和目的地是否和预设一致即可；如果是火车，最好在邻接矩阵上处理，因为一辆列车他的起始点可能和预设的不一样，但中点站是可能满足预设的。

在算最优方案时，对于飞机，将不同航班之间的权重进行比对即可；对于火车，在邻接矩阵基础上，看（A,B）点的 vector 值中火车的权重，通过比对即可得到最优方案。

如果游客想显示的是有中转站的信息，有中转站意味着交通工具会发生变化。这个时候我们需要基于图的Dijkstra算法，但不同于Dijkstra算法的在于他需要加上限制条件，根据原本题目的要求，在中转站的等候时间：飞机至少二小时，火车至少一小时，所以在判断某点最短值的时候，如果下一次乘坐的是飞机，则下一次的xxxtime1至少等于本次的这一次的xxxtime2 + 120min，如果下一次乘坐的是火车，则下一次的xxxtime1至少等于本次的这一次的xxxtime2 + 60min，所以要筛选出符合要求的交通方式。

还有一点要清楚的是Dijkstra算法比较的是边的权值，也就是邻接矩阵的某点值，但这里我们定义的邻接矩阵某点值对应的是一个哈希表，此时比对的就应该是xxxhour，将该点的各个xxxhour比对得出最小值作为Dijkstra算法比较的对象。上面是以时间作为标准，如果是以费用作为标准，则比对的就应该是xxxmoney（但仍须满足中转等待时间筛选条件）

**说明**：我们有想过更换交通工具（比如从火车变成飞机）可能会导致由于等待需要的时间不同而出现最短时间计算有问题，但其实通过筛选后，他们就在一个起跑线上了，不会有比较的错误。

④**昼夜更替**

**分析：**所谓“昼夜更替“强调时间维度的计量上，这是第三个问题的说明，为什么Dijkstra算法要比较xxxhour而不是xxxtime2，因为当某种交通工具运行时间跨越了从当天的24:00到第二天，在比较的时候，第二天的比如 6:00虽然表面上是比23:43小的，但由于他是在第二天，实际他是更大的，为了简化比较的方式使之不涉及日期转换，比较xxxhour会更加方便。

“昼夜更替”不仅仅是源自于问题，在算最终到达的时间也有用，xxxtime2+xxxhour可能会超过当天的日期，此时还是需要将日期进行转换。

⑥**友好界面**

**分析：**根据题意，“主程序可以有系统界面、菜单；也可用命令提示方式；选择功能模块执行，要求在程序运行过程中可以反复操作。”，我们采用 do-while 的方式，通过输入不同命令对应的数字，来执行相关的操作，只用不输入指定的退出数字，程序便会一直执行下去。

**拓展需求分析：**

①**自驾出行**

**分析：**我们发现，关于时间的比较，直接利用列车或者飞机的运行时间相关信息即可，题目要求存储的城市之间的里程并没有利用到，为了不浪费空间，我们想到了自驾，而自驾的时间就用 （城市之间的里程÷自驾车速）即可得到，自驾的费用就用 （城市之间的里程×单位距离油费），我们简化问题，通过查询资料得到一个大致的家用车在高速路段的自驾车速与单位距离油费（并一直保持不变）。但这和实际情况肯定会有一定的偏差，不仅仅是因为自驾车速与单位距离油费的理想化，还有一个原因就是两个城市之间的高速路段里程和城市之间的距离并不是直接相等的，我们在这里都简单化处理。

**说明：**自驾出行的时间和费用也会参与和列车、飞机的时间和费用的比较，由游客自行选择是否显示。

②**所有路径显示**

**分析：**我们在原有需求分析中的问题4中提到了日常地图软件的所有路径显示，这里我们给出解决方法。所有的路径，我们联想到的就是图的两种探索方法深度优先探索和广度优先探索，我们在这里采用深度优先探索来解决两点之间的所有路径。但同样的，在每次探索的时候，还是要受专程等待时间的约束，只有满足时间要求的专程才予以考虑。

**设计：**采用DFS，设置一个visited的unordered\_map，他的值是某城市，他的键为0或1，0的表示还未访问，1表示已访问（这里不用vector是因为vector的位置需要用整形数据来表示），从游客设置的城市出发，置他的键值为1，找到他的一个未访问的邻接点，再从该点出发，置该城市的键值为1，继续找到他的未访问的邻接点，循环往复，直到找到了一个键为目的地即可。但这只是找到一条路径，想要找到所有的路径，需要回溯重置，当找到目标点后，回退他的前一个点，置此点的键值为0，然后再探索他的其他邻接点寻找路径，并且还要在该点基础上继续回退到再前一个点，置他的键值为0，找其他的路径，以此类推，找到所有的路径。由于存在多路径，我们需要引入vector path 来实时保存两点之间的路径，对于满足要求的，再将path中记录的点存储到vector<vector<vector>>（A->B->C->D,A->B->D）中进行统一输出。这里探索的时候，每个点找他的临界点，依据的是前面我们定义好的边的信息再加上专程等待时间的约束，这里我们得到的路径是指城市之间的联通，最后每个城市之间肯定会有多种交通方式，我们将会输出所有的通行方式，具体实现的话，就是对三维向量外两维进行遍历并依据邻接表（邻接表中每个点也是一个vector，所以总共是一个三维向量）再次遍历得到所有的出行方式，具体的输出将采用 A->B->C->D + 各段的通行信息。这里我们将会引入总时间计算函数，其实前面也会用到，便可以得到所有出行方式的总时间。

**说明：**其实得到所有出行方式的总时间后经过排序便可以得到最优出行方式，但由于题目要求使用Dijkstra算法，我们还是设计了Dijkstra算法。

③**自主设计中点站**

**分析：**由上面2的分析后，这个点就容易解决，游客可以选择输入想要经过的中点站（由于中点站之间有必要的停留时间，游客可以在此期间内购买当地的特产，但停留时间的设置暂且不考虑），于是便在所有的通行路径中寻找是否有，既满足出发地、起始地、中点站的，同时又满足出发时间、专程等待时间这几个条件的路径，然后有满足的输出即可。

④**夜间出行顾虑**

**分析：**日常生活中，我们还会遇到：我们的父母担心我们夜间出发会有安全隐患或者我们自己太晚甚至凌晨到达目的地后住宿不便的情形。也就是在2的基础上，我们得到了所有的出行方式后，再对出发时间和到达时间进行筛选即可，设置一个出发时间和到达时间范围，一一筛查再输出即可。

**显示距离**