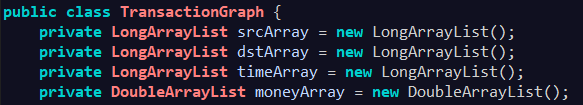
**一.将.csv文件中的数据读取并且按照设计的数据结构存储。**

对应边的.csv文件数据有4列，分别表示起始点id, 终止点id，时间戳， 交易金额。

四列数据分别用四个动态数组集合中保存



这四个集合封装成一个类的属性

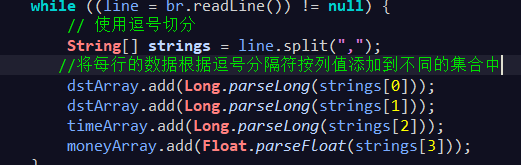
**二.将.CSVwe文件按照行进行读取，每读入一行就将每列的数据根据逗号分隔符进行切割分别添加到对应的集合中。**

例如起始点id添加到集合srcArray集合中

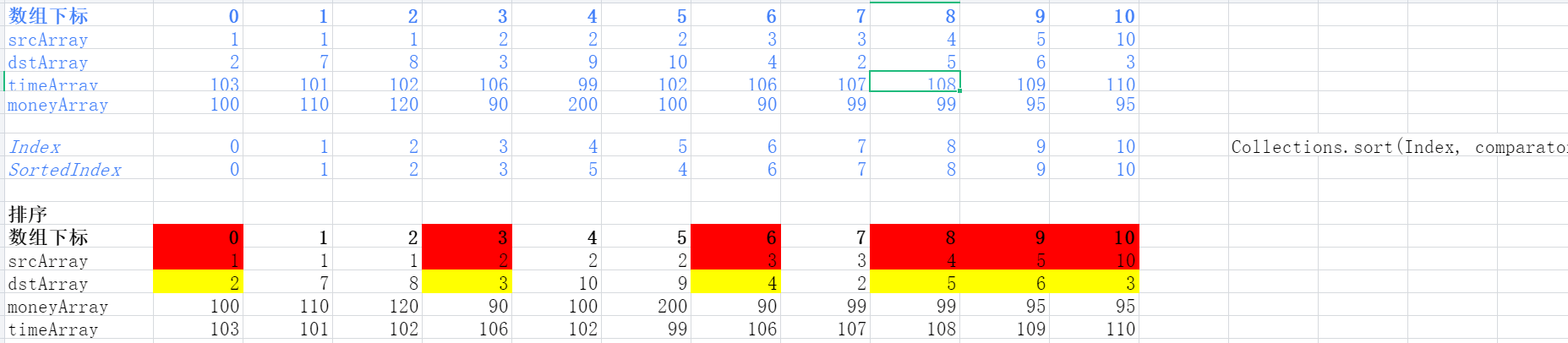
终止点id添加到集合dstArray集合中

时间戳添加到集合timeArray集合中

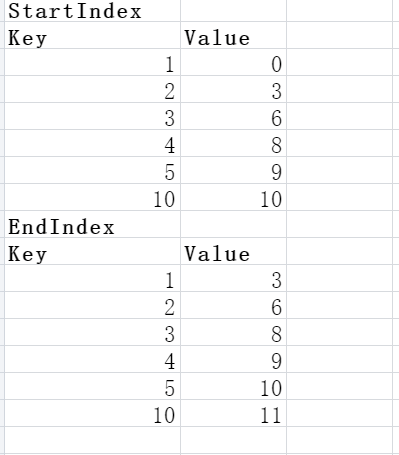
交易金添加到集合moneyArray集合中



**三.数据读入集合中后，先将起始点集合排序，起始点相同的再按照交易金额集合排序**



**四.建立两个起始点id集合的hashmap。对于StartIndex,Key表示起始点id，value表示第一次该点出现的索引下标。对于EndIndex,key表示起始点id，value表示最后一次该点出现的索引下标。**



**五.进行环路的查找**

遍历的算法：BFS(深度优先遍历)

约束条件：

（1）环的长度范围在3-6条边

（2）环上每条边有属性时间戳，时间戳按照递增的顺序。

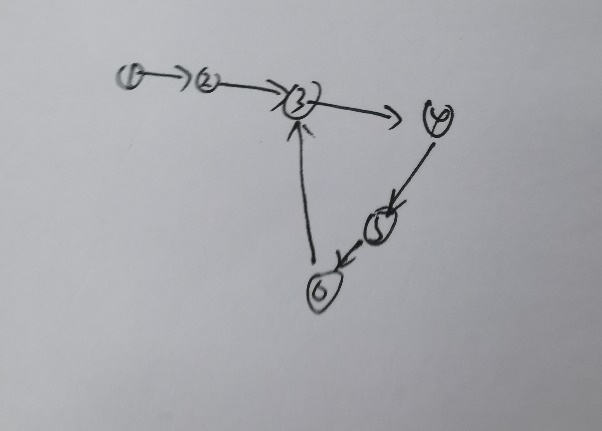
（3）环上每条边还有交易金额的属性，下一条边的交易金额要在上一条金额的90%--110%范围内。

（4）环上的每个顶点只能出现一次

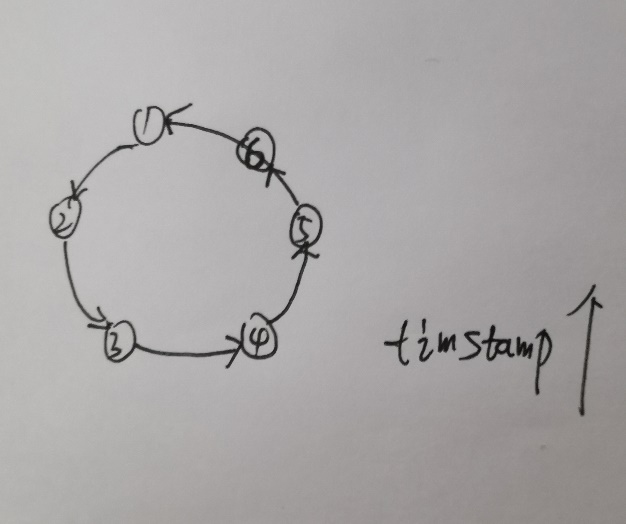
可能会影响查询效率的问题：

（a）环路重复遍历，例如找长度为6的环路的时候，可能里面已经包含长度为3或者4的环路了，当再需要找长度为3或者4的环路时又需要重新遍历一次。

下图中当节点6遍历到3时，应该回溯到6？



(b) 一条环路从不同的起点开始遍历会被重复遍历多次。



**代码实现时，根据交易金额的约束进行遍历的减枝**