本文件中共有四个类：Edge, Graph, IntegerComparator 和 AssigTwoz5172701。Edge、Graph、IntegerComparator都实现了Serializable接口。

Edge类表示有向图中的每一条边，它包含起点start, 终点target 和 权重weight.

Graph类表示有向图，它包含邻接表adjList(一个关于Edge的列表)，顶点数verNum, 边数edgeNum, 计算单源最短路径的源顶点start, 存储源顶点到顶点i之间距离的数据dis, 存储顶点i关于源顶点的前驱节点数据pre。

我们可以使用方法public Graph(int vnum)方法创建一个有向图对象；使用方法public void insertEdge(String startStr, String targetStr, String weightStr)向有向图对象中插入一条边；使用方法public void traverse()查看这个有向图；使用方法dijkstra(String startStr)设置源顶点，并计算出到各个顶点的最短路径与距离；使用方法public String getSortedStringResult()得到已排序的符合输出格式的字符串；使用方法public LinkedList<Tuple2<Integer, Tuple3<String, Integer, String>>> getSortedTupleResult()得到已排序的关于最短距离和最短路径的元组；使用方法public LinkedList<Tuple2<Integer, Tuple3<String, Integer, String>>> getUnsortedTupleResult()得到未排序的关于最短距离和最短路径的元组。

IntegerComparator类是自定义的关于Integer对象的比较器。

1. 检验程序的参数个数是否为三个，如果不是则直接退出程序。否则获取参数中源顶点、输入文件路径和输出文件路径的值，然后创建JavaSparkContext。
2. 读取输入文件，创建一个JavaRDD<String>。
3. 将每一行分解为含有三个String的元组，得到一个JavaRDD<Tuple3<String, String, String>>
4. 创建HashSet对象用于计算顶点个数。
5. 计算顶点个数。依次将上一步中得到的每一个元组中的start和target插入到set中，最终set的最大值即为顶点个数。
6. 创建有向图对象，并遍历（2）中得到的集合，依次添加到graph中
7. 使用dijkstra(String startStr)方法执行算法，计算graph的单源最短路径。
8. 将getUnsortedTupleResult()得到的结果创建为JavaPairRDD<Integer, Tuple3<String, Integer, String>>。
9. Sort by distance and get the formatted tuple list then save to file