程序说明及用例分析测试

# 测试用例说明

我们把测试用例分为三个文件：topo.csv，demand.csv，result.csv。

1、topo.csv：存储无向图信息，每一行为： 边编号，节点1，节点2，边权重； 节点编号从0开始，边编号从1开始；

2、demand.csv：限制条件信息

第一行： 起点，终点；

第二行：必经节点编号，以逗号隔开；

第三行：必经边集合，边的两个节点用逗号隔开，边中间用|隔开；没有必经边此行填NA；

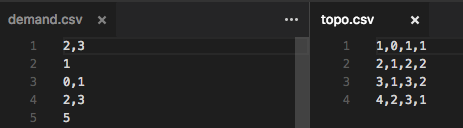
第四行：禁止边集合，边的两个节点用逗号隔开，边中间用|隔开;没有必经边此行填NA；

第五行：最多经过的节点数；

3. result.csv：存储运行结果

程序运行时读取这三个文件，topo.csv为无向图信息，demand.csv为限制信息，输出结果存在result.csv中。

如下图所示，为一个简单case，及其文件表示形式，表示起点2，终点3，必经点1，必经边（0，1），禁止边（2，3），节点限制5的一个case。



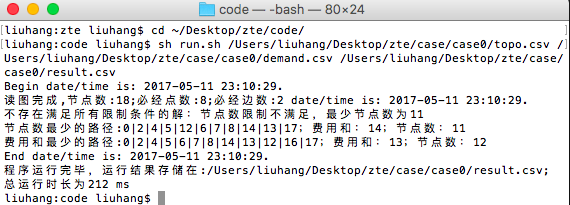
# 程序说明

我们的程序读取图信息和限制信息，如果得到满足所有条件的最优解，输出；如果不能满足所有条件，输出近似解，并给出不能满足的原因。由于我们使用了两种算法，所以给出两个程序，具体说明如下。

## 指派模型结合分支定界算法

### 使用说明

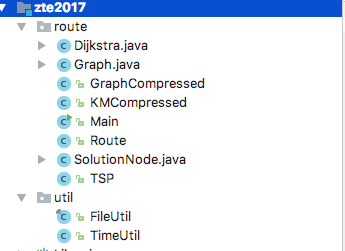
运行环境：Linux /mac osx系统，程序用Java语言编写，运行环境需预先安装java运行环境(jdk/jre 1.8版本)；我们提供运行脚本，使用如下：在终端中直接运行命令；run.sh /xxx/topo.csv /xxx/demand.csv /xxx/result.csv；如下图所示，为运行官方用例时的情况：



此外，源代码在code/src文件夹下，我们提供一键编译打包的脚本，直接运行build.sh即可得到jar包。

### 源代码说明

源代码结构如下：



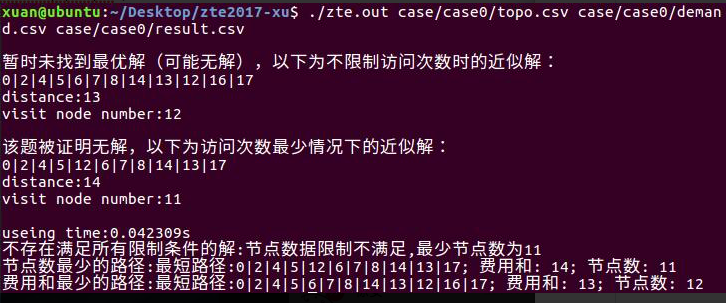
各代码文件说明:

1. Main.java为程序入口，通过main函数带参数运行，调用Route.searchRoute()运行算法；
2. Graph.java操作文件，构建原始图和读取限制条件信息；
3. TSP.java 为求解TSP模型代码，主要包括分支定界法的实现(branchAndBound函数)；
4. KMCompressed.java为用KM算法求解指派问题的实现；
5. Route.java整体算法入口，实现迭代求解，控制算法流程；
6. Dijkstra.java为堆优化的Dijkstra算法实现；
7. GraphCompressed.java为对原始图进行压缩，并构建新图；
8. SolutionNode.java为分支定界法的解节点；
9. TileUtil.java和Timeutil.java为工具函数，读取文件和计算时间；

## 线性规划结合lpsolve求解器

本程序用C++编写，并调用了lpsolve5.5.2.5库。运行环境为**Linux64位系统**（其他系统，如Linux32位系统或win32/64需重新进行编译，详见/code/lp/README.txt）。

程序使用如下：在Linux64位系统中（如ubuntu 16.04），在终端中，进入lp文件夹下，运行：./zte.out /xxx/topo.csv /xxx/demand.csv /xxx/result.csv，如下图所示，为运行官方例子时的情况：



# 用例分析及设计

我们总共设计了20个测试用例，各个测试用例的情况如下：

1. case0: 官方赛题上所给的用例；
2. case1-case3: 验证算法及程序正确性的用例，其中
   1. case1: 用于测试是否考虑节点可重复经过，本用例必须重复经过节点才能得到可行解；
   2. case2: 用于测试算法对限制条件“路径经过点数不超过k”的响应，考虑节点限制得到的解中路径不能经过重复点，且路径长度比不考虑节点限制小；
   3. case3：测试验证，考虑节点可重复经过的情况下，是否可以得到费用值更小的解；
3. case4-case20: 程序性能验证，验证程序可以跑多大规模的case。
   1. case4-case6为中等规模case，无向图点数在1000以内；
   2. case7-case20为大规模case，无向图点数在1000-2000；

各个测试用例的详细介绍，见每个case文件夹下的用例说明.txt/用例说明.png，每个case的运行结果见相应文件夹中的result.csv。

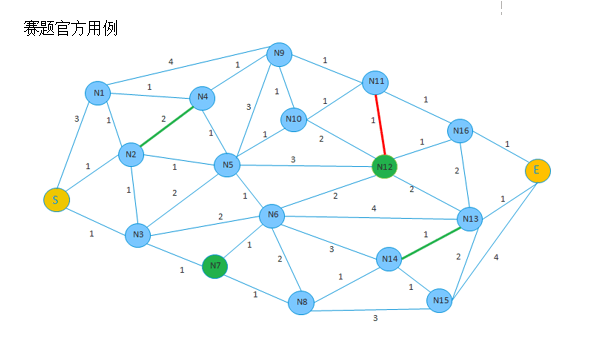
# 程序测试

程序测试主要包括以下三个部分：

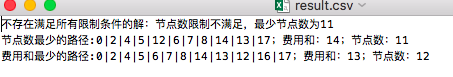
1. 对赛题中所述示例的测试；
2. 对程序正确性及各种特殊情况的测试；
3. 对程序性能的测试，看看程序能跑多少节点、多少必经点、多少必经边的图以及运行的时间。

测试环境如下：CPU：intel酷睿i7；操作系统：Linux Ubuntu 16.04 64位；内存：8G；内核：4核。

## 官方case



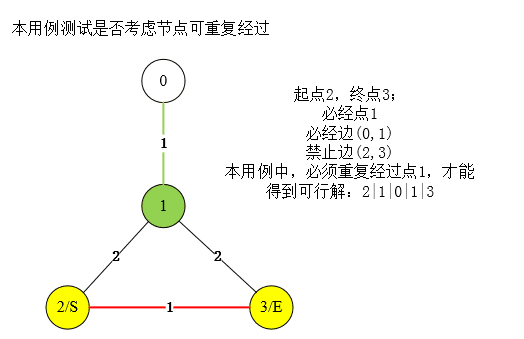
我们把赛题上所描述的示例制作为用例(case/case0)，起点谁0，终点设为17，包含18个节点，2条必经边，1条禁忌边，2个必经节点，限制经过9个节点。运行结果如下：



满足所有限制条件的解是不存在的，输出结果中给出了解不存在的原因，即节点限制不满足。同时，我们给出了两个近似解，费用和最少的路径和经过节点数最少的路径。

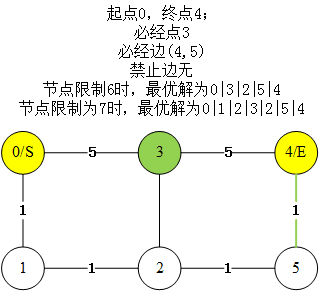
## 正确性测试

### case1



本用例用于测试算法是否考虑节点可重复经过，本例中节点1必须重复经过才能得到可行解。经测试，我们使用的算法可以得到可行解2|1|0|1|3，所以，**我们的算法重分考虑了必须经过重复点才能得到可行解的可能性**。

### case2



本用例用于测试算法对限制条件“路径经过点数不超过k”的响应，考虑节点限制得到的最优解的费用不是最小的且路径中没有重复点（如上图所示）。 所以，**我们的算法充分考虑了节点数的限制，并进行了正确的处理。**

### case3

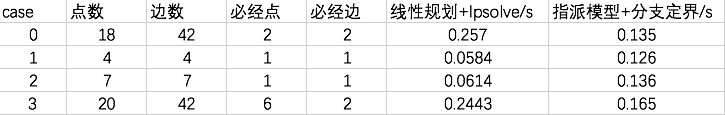
本用例用于测试验证：考虑节点可重复经过的情况下，是否可以得到费用值更小的解。本用例的情况如下：

节点可重复经过时路径：2|15|18|3|11|4|13|7|13|17|6|5|9|14|19|0|1|0|19；费用和：103；节点数：19

节点不可重复经时路径：2|15|10|12|14|9|17|6|5|3|11|7|13|4|1|0|19；费用和：126；节点数：17

所以，经过验证：**考虑重复经过，有可能得到费用值最小的解**。

Case0-case3的测试情况汇总如下：



后两栏为两种算法运行的时间。**经过测试，两种算法均能通过正确性测试，说明算法及模型正确有效。**

## 性能测试

根据我们的算法，影响性能主要由三点因素：无向图的的点数，必经点数（包括起点终点），必经边数。Case4-case20为各种不同规模的用例，测试情况如下



如上图，为对两种算法的性能测试。经过测试，我们发现：线性规划+lpsolve求解器在必经点规模达到20时，很难得到可行解，这主要受限于lpsolve求解器的性能；**指派模型结合分支定界的方法，表现出非常良好的性能，在目前的测试环境下，5s内可求解出100个必经点100条必经边的case;10s内可求解100必经点200必经边的case；30s内可求解100必经点300-400必经边的case。**