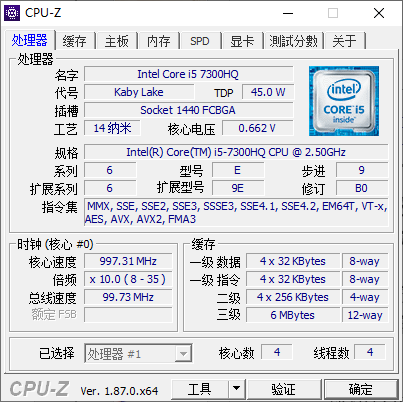
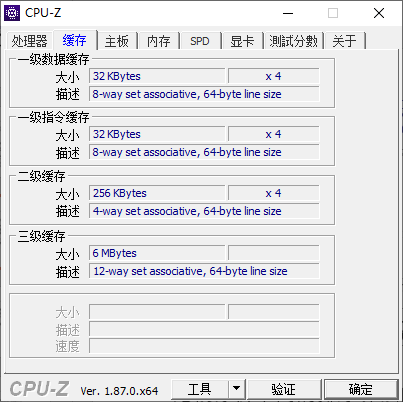
实验报告

1. 实验内容：

实验1：实现求有向图的强连通分量的算法。有向图的顶点数 N 的取值分别为： 9、27、81、243、729， 弧的数目为 Nlog\_3N, 统计算法所需运行时间 ，画出时间曲线。

实验2：实现求所有点对最短路径的Johnson算法。有向图的顶点数 N 的取值分别为: 27、81、243、729 ，同一顶点数目对应两种弧的数目： Nlog\_5N和Nlog\_7N（取下整）, 统计算法所需运行时间 ，画出时间曲线。

二．实验环境

笔记本使用两年半



1. 实验方法和步骤

将main()写在一个lab.c里，图相关算法写在graph.c（ex1）或graph.cpp(ex2)。

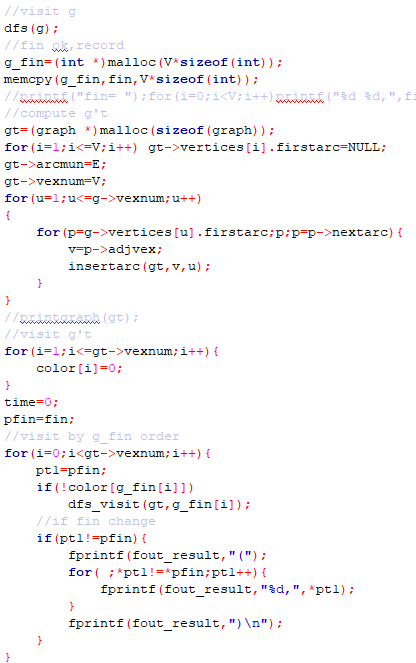
照旧使用字符串操作拼接文件名。

照旧使用QueryPerformance计时

求强联通分量的实现，我没有专门写一个函数，而是直接写在了主函数里，

我只调用了写好的dfs(),dfs\_visit()。这样可以使我输出分量更为顺滑。

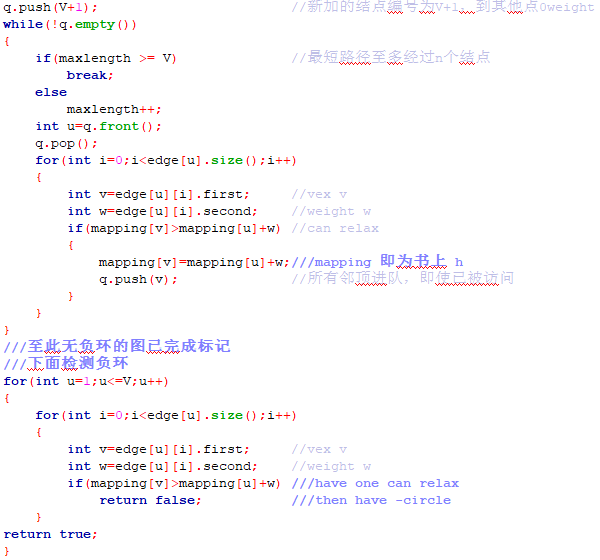
计时开始与结束分别在这一段前后。



强联通分量的实现参考书上伪码，图的实现及遍历相关部分使用了我之前写过的代码，但是太久没用过和新写了一趟没好大差别T\_T。

在第二个实验中，由于用邻接表存储的图需要用到优先队列等操作，我放弃了手写这部分，投向c++的怀抱，使用已有的数据结构。（不是我不想自己实现，这样的话其它部分的代码就要远多于我要实现的两个算法了，本末倒置）~~（其实就是快考试了没时间）~~

adjust() 函数主体部分，求用以调整wight的mapping（也就是书上的h），并且在最大路径长度 >= V-1 时再跑一趟判断是否能 relax 是则有负环，返回false；否则返回true。



为了判断数据中是否有负环，我专门在ex2/input/ 目录下写了个.c用来判断，若有，那么我丢弃掉这份数据，重新生成input\*\*.txt 进行实验。

确保在正常实验时不会有非法数据。

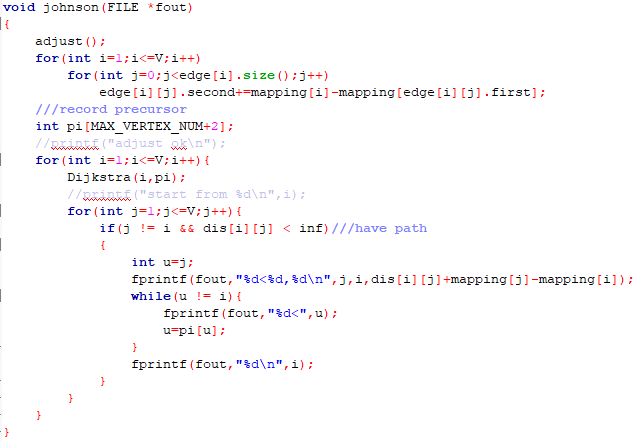
%随机生成太难没有负环了，手动改变部分负值，虽然这样会破坏部分随机性，但我试了十几趟怎么还是有负环T\_T。

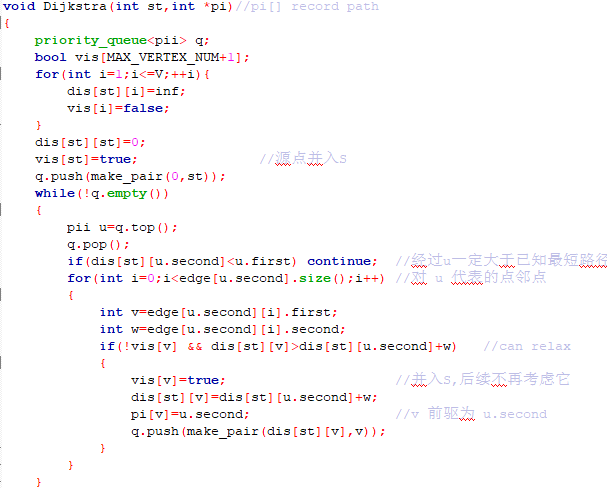
%随机生成使用matlab。

我将输入输出写在了johnson算法的里面，导致算法运行时间很大，基本都是在处理输出，这样时间的数据就没有代表性了，但是我实在不想把我正常运行的算法再改一趟了TT

下为johnson 及 Dijkstra 算法

Johnson先做一次调整，得到mapping后更改所有边的权值以进行dijkstra

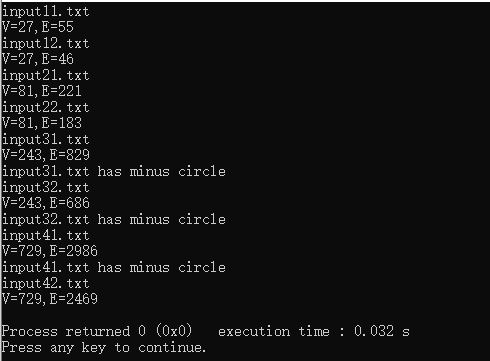


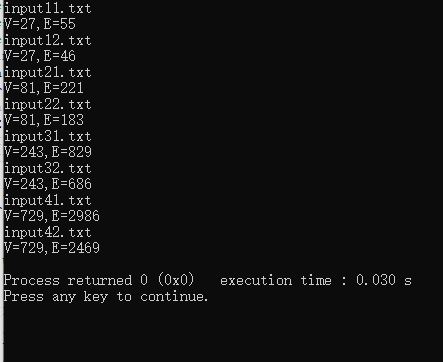


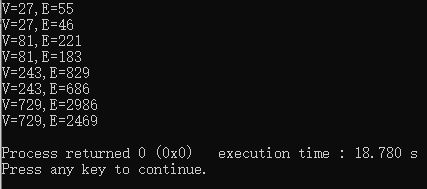
算法实现基本根据书上伪码，使用一些c++的结构以减少代码量。

还有，输出所有点对是逆序的，最短路径的文件实在是比较大，我将ex2输出专门打包为一个压缩包，让助教可以下载快点。

1. 实验结果与分析

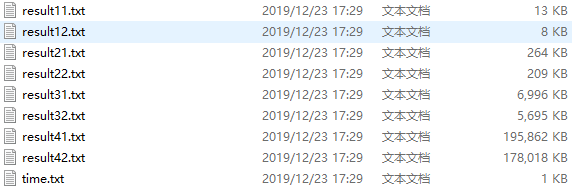
原数据

修改后的数据

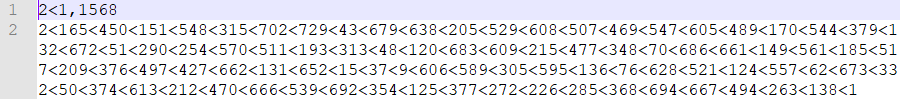


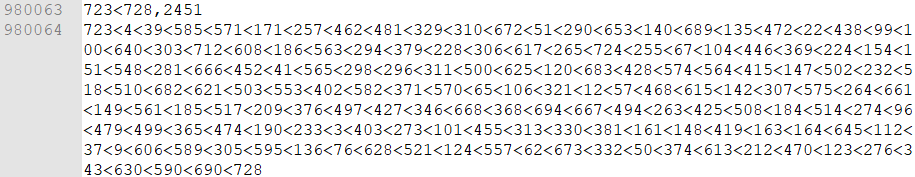
但是为了输出路径

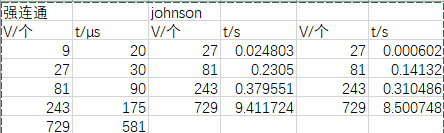
运行时间还是很大



ex2的输出画风，







对于ex1，运行时间曲线还较为正常，与理论分析O（V+E）没有太大差距

但对于johnson算法，这个图就根本不能看

理论上，对于E=O(N\*log(N))

Johnson算法运行时间O(V^2\*log(V))