实验四 图算法

袁雨 PB20151804

一、实验设备和环境

- mac，vscode，g++

二、实验内容和要求

（一）实验内容

- **实验4.1：Johnson算法**

  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 ，每个顶点作为起点引出的边的条数取值分别为：$log_5 N$、$log_7 N$ （取下整）。图的输入规模总共4*2=8个，若同一个N，边的两种规模取值相等，则按后面输出要求输出两次，并在报告里说明。（不允许多重边，可以有环。）

（二）实验要求

1.编程要求

- C/C++

2. 目录格式

实验需建立根文件夹，文件夹名称为： 编号-姓名-学号-project4， 在根 文件夹下需包括实验报告和ex1实验文件夹，实验文件夹包含3个子文件夹：

- input文件夹：存放输入数据
- src文件夹：源程序
- output文件夹：存放输出数据

实验4.1 Johnson算法

- ex1/input/

  每种输入规模分别建立txt文件，文件名称为input11.txt, input12.txt,……,input42.txt （第一个数字为顶点数序号（27、81、243、729），第二个数字为弧数目序号（ log5N、log7N ））；生成的有向图信息分别存放在对应数据规模的txt文件中；每行存放一对结点i,j序号（数字表示）和wij，表示存在一条结点i指向结点j的边，边的权值为wij，权值范围为[-10,50],取整数。Input文件中为随机生成边以及权值，实验首先应判断输入图是否包含一个权重为负值的环路，如果存在，删除负环的一条边，消除负环，实验输出为处理后数据的实验结果，并在实验报告中说明。

- ex1/output/

- result.txt：输出对应规模图中所有点对之间的最短路径包含结点序列及路径长，不同规模写到不同的 txt文件中，因此共有8个txt文件，文件名称为result11.txt,result12.txt, ……,result42.txt;每行存一结点的对的最短路径，同一最短路径的结点序列用一对括号括起来输出到对应的txt文件中，并输 出路径长度。若图非连通导致节点对不存在最短路径，该节点对也要单独占一行说明。
- time.txt：运行时间效率的数据，不同规模的时间都写到同个文件。
- example：对顶点为27，边为54的所有点对最短路径实验输出应为：(1,5,2 20)(1,5,9,3 50) ……，执行结果与运行时间的输出路径分别为：
  - output/result11.txt
  - output/time.txt

3.实验报告

- 实验设备和环境、实验内容及要求、方法和步骤、结果与分析。
- 比较实际复杂度和理论复杂度是否相同，给出分析。

三、实验方法和步骤

参照教材 Johnson 算法的部分。如果图 G=(V, E) 所有的边权重皆为非负值，我们可以通过对每个结点运行一次 Dijkstra 算法来找到所有结点对之间的最短路径；如果图包含权重为负值的边，但没有权重为负值的环路，那么只要计算出一组新的非负权重值， 然后使用同样的方法即可。Johnson 算法在运行中需要使用 Dijkstra 算法和 Bellman-Ford 算法作为自己的子程序。这两个算法的实现同样参照教材。

Dijkstra 算法使用模板类 priority_queue，其使用的数据结构为二叉堆。并根据数据结构构建比较函数 cmp 。

扩展 Bellman-Ford 算法，若判断出输入图包含一个权重为负值的环路，则删除负环的一条边，消除负环。具体地，若 BellmanFord 函数在第|V|次 RELAX 操作时找到的第一处更新为 G[v].d > G[u].d + adj->w，则说明节点u在负环上。接下来循环进行 RELAX 操作，直到找到一个节点 x 指向 u ，满足 G[u].d > G[x].d + adj->w，则此处的 adj 就是负环上的边节点，将该边删掉。在 Johnson 函数中，循环调用 BellmanFord 函数，直到函数返回true，即图中不存在负环。

在主函数中，随机生成有向图信息（排除自环和多重边），然后在图上运行 Johnson 算法，并输出到 output 。使用 clock() 等函数完成计时，输出到 time 。

四、实验结果与分析

1.实验结果

- 目录结构

```
5-袁雨-PB20151804-project4\ex1
  input
    input11.txt
    input12.txt
    input21.txt
    input22.txt
    input31.txt
    input32.txt
    input41.txt
    input42.txt
  output
    output11.txt
    output12.txt
    output21.txt
    output22.txt
    output31.txt
    output32.txt
    output41.txt
    output42.txt
    time.txt
  src
    Johnson.cpp
```

以顶点为27，边为54为例。

- input11.txt

ex1 > input > input11.txt

```
1 6 45
1 20 1
2 7 -4
2 9 39
3 12 -3
3 27 41
4 11 -1
4 22 7
5 26 23
5 20 35
6 2 -9
6 11 1
7 15 17
7 12 3
8 25 19
8 17 -8
9 12 34
9 26 0
10 11 6
10 9 6
11 7 12
11 22 5
12 20 8
12 16 1
13 8 41
13 25 40
14 5 41
14 20 -2
15 24 8
15 5 8
16 4 4
16 6 25
17 18 50
17 12 22
18 17 35
18 26 19
19 16 41
19 20 -5
20 7 23
20 14 -5
21 3 20
21 18 29
22 21 47
22 6 43
23 9 44
23 10 30
24 26 -1
24 12 -8
25 23 -3
25 10 0
26 7 42
26 11 -6
27 8 8
27 17 35
```

- output11.txt

因篇幅原因只截图部分。

ex1 > output > output11.txt

```
(1,27,13,2 43)
(1,27,13,9,10,3 40)
(1,27,23,18,12,4 94)
(1,27,13,9,10,3,14,17,5 133)
(1,27,23,18,6 80)
(1,27,13,9,10,3,7 61)
(1,27,13,9,10,8 46)
(1,27,13,9 9)
(1,27,13,9,10 8)
There is no path from 1 to 11.
(1,27,23,18,12 68)
(1,27,13 -6)
(1,27,13,9,10,3,14 74)
There is no path from 1 to 15.
There is no path from 1 to 16.
(1,27,13,9,10,3,14,17 87)
(1,27,23,18 34)
(1,27,13,9,10,3,14,17,5,22,25,19 198)
(1,27,13,9,20 52)
(1,27,13,9,10,8,21 53)
(1,27,13,9,10,3,14,17,5,22 177)
(1,27,23 0)
There is no path from 1 to 24.
(1,27,13,9,10,3,14,17,5,22,25 188)
(1,27,13,9,10,3,7,26 60)
(1,27 -9)
(2,6,7,26,8,21,23,1 137)
(2,13,9,10,3 85)
(2,6,7,26,8,18,12,4 181)
(2,6,14,17,5 111)
(2,6 43)
(2,6,7 77)
(2,6,7,26,8 77)
(2,13,9 54)
(2,13,9,10 53)
There is no path from 2 to 11.
(2,6,7,26,8,18,12 155)
(2,13 39)
(2,6,14 52)
There is no path from 2 to 15.
```

- time.txt

ex1 > output > time.txt

```
0.000764
0.000519
0.008044
0.009005
0.080201
0.097029
0.772595
1.051523
```

2.实验分析

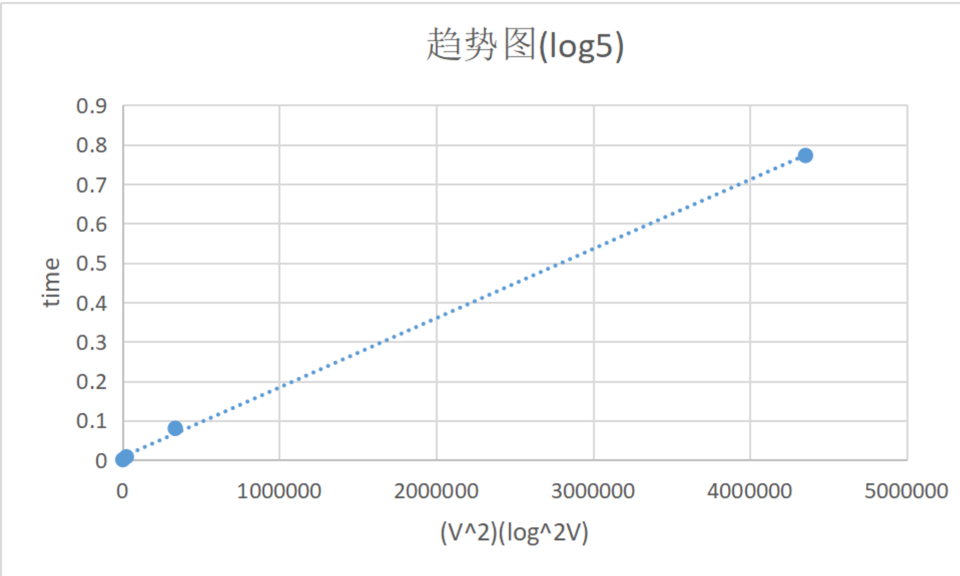
本次实验中使用二叉最小堆实现的Johnson算法。因为删除负环边会受随机数据的影响、变量赋值会受编译器优化的影响等，对运行时间的影响较大，故将其视作预处理过程，不算入运行时间，则剩余部分的理论运行时间为 $O(VElgV)$ 。又$E = VlgV$，故理论运行时间为$O(V^2lg^2V)$。

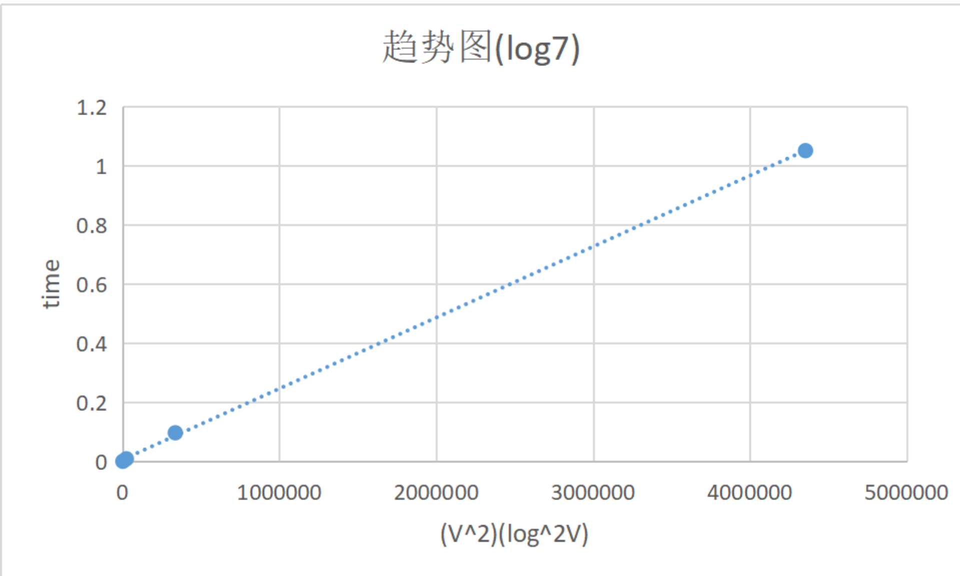
实验中得到的数据如下表。

顶点数N	原边数	消除负环删除的边数	N^2log^2N	运行时间(s)
27	$Nlog_5N$=27*2=54	0	1494	0.000764
27	$Nlog_7N$=27*1=27	0	1494	0.000519
81	$Nlog_5N$=81*2=162	5	23897	0.008044
81	$Nlog_7N$=81*2=162	4	23897	0.009005
243	$Nlog_5N$=243*3=729	79	336055	0.080201
243	$Nlog_7N$=243*2=486	0	336055	0.097029
729	$Nlog_5N$=729*4=2916	663	4355270	0.772595
729	$Nlog_7N$=729*3=2187	155	4355270	1.051523

两种边的规模log_5N、 log_7N 相差常数倍，故将数据分成两组，即序号11、21、31、41为一组，序号12、22、32、42为一组，分别作图观察。

以V^2lg^2V为横轴，程序运行时间time为纵轴，作图如下：





可见两图中V^2lg^2V均与运行时间time近似成线性关系，符合理论复杂度。