参考文献引用网络分析

胡恩[1]

摘要

评估学术论文影响力的重要因素之一是被引次数。本文参考 PageRank 算法，设计了对具有基本影响力（即能被加入分析系统）论文，根据参考文献引用网络，进行论文影响力分析的系统。本文算法的设计基于有向无环图（DAG），运用已学算法拓扑排序（Topological Sorting）完成了图的压缩存储，参考了 PageRank 算法的思想。算法能根据文献引用，分析文献的影响力（weight）和引用关系，并且能够完成“增、删、改、查、DFS、写入数据、读取数据、打印信息”等操作，其中，“增、删、非递归 DFS、读取数据并建立系统”操作是算法难点。

关键字：有向无环图、拓扑排序、PageRank 算法、非递归 DFS

1 引言

在学术界，评估学术论文影响力的重要因素之一是被引次数。在某一领域，某一论文被引次数越多，该论文的影响力也就越大。对于研究人员，为了迅速了解某一领域的前沿研究方向，最好的方法是先阅读当期影响力较大的论文。于是，分析论文影响力的工作必不可少。

本人希望根据所学的数据结构与算法，设计一种能够根据论文之间的引用关系，建立参考文献引用网络的系统，进而分析论文的影响力，即权重，以此巩固数据结构与算法的设计能力。

2 研究方法

2.1 算法基于的数据结构

一篇论文可能引用多篇论文，也可能被多篇论文引用，故算法的设计基于图的数据结构。而引用关系是有向的，故图是有向的。由于后发表的论文只能引用先发表的论文，故在有向图中，不存在回路（环），故图是有向无环图。由于论文和引用关系，我们可以令一个顶点代表一篇论文，一条弧代表一个引用关系。我们需要分析论文的影响力，因此图的顶点应有相应权重。为方便讨论，先设置弧也有相应权重，具体意义后谈。

综上，算法基于的数据结构应为一个顶点和弧均带权的有向无环网，简记为 DAG。

2.2 顶点权值（论文影响力）确定

如何确定顶点的权值（即论文的影响力）是算法最关键的问题。本文参考 PageRank 算法的思想设计了一种简化的算法。

由算法要求，首先我们需要假设进入分析系统的论文必须具有基本的影响力（类似 SCI 期刊），否则，若任何论文都可以进入系统进行分析，大量质量低劣的论文将会严重影响分析系统。顶点权值的确定分为以下几步：

（1）由假设，我们设置每个顶点的初始权值为 1，然后根据引用关系，建立有向无环图，此时弧应不带权值。

（2）对有向无环图进行拓扑排序，拓扑序列的意义在于，若顶点 A 在顶点 B 之前，则 A 不可能被 B 引用。

（3）根据拓扑序列的性质，我们可以由拓扑序列由前到后逐步更新顶点的权值，并且只需一趟更新。根据这样的步骤更新顶点权值 A：首先对 A 的所有入弧的权值权值求和，记为 S，若无入弧则和为 0；其次，设置顶点 A 的权值 weight 为 1+S；然后，求得 A 的出度 outdegree；最后，设置 A 的所有出弧权值为 weight/outdegree。

2.3 增删顶点

结合现实，增加一个顶点意味着有一篇论文新发表。由于这篇论文是最新发表的，所以它不可能被其他论文引用，对于到 DAG 中，即该顶点无入弧。于是根据拓扑序列的性质，我们只需在拓扑序列的最前端添加一

个顶点，建立该顶点对其他顶点的引用关系，然后更新它的所有子孙及其邻接弧的权值即可。

结合现实，删除一个顶点意味着撤回一篇论文，参考 2018 年心脏干细胞学术造假事件。撤回一篇具有影响力的论文往往意味着学术界的巨大震动，被撤回论文的所有祖先（即引用被撤回论文和引用被撤回论文的论文和⋯的论文）都会受到影响。在 DAG 中，删除一个顶点意味着也要删除所有它的祖先顶点。这涉及到大规模数据的删除和顶点表和邻接下三角阵的大规模数据移动。由于研究时间有限，本人未能设计出删除顶点的算法。

2.4 数据结构基于的存储结构

由于我们需要快速查找顶点的出入弧，且根据上节讨论，删除操作较少，增加操作只需在数据边缘增加，DAG 采用邻接矩阵存储弧集，顺序表存储顶点集。

在顺序表中，根据拓扑序列由后往前的顺序，排列顶点和邻接矩阵的序号。由于拓扑序列的性质，邻接矩阵是一个下三角矩阵，故我们采用下三角矩阵的压缩存储形式存储邻接矩阵。

顶点顺序表的数据元素为：

string paper(论文名)

string author(作者名)

double weight(权值)

int indegree(入度)

int outdegree(出度)

邻接矩阵，对角线元素为 0，无弧处元素为 0，其余为弧的权值。

2.5 读取数据并建立 DAG 的算法

最初读取数据并建立 DAG 的算法（即构造函数），是最重要而且是最困难的算法。

构造函数对数据文件有格式要求。根据以下步骤建立 DAG：

（1）读取顶点数

（2）读取：是否已经经过拓扑排序和设置权值的标志

（3）若是则可直接读入

（4）若否，则进入下列程序：

（5）建立临时的顶点顺序表 tempVex 和临时的非三角 bool 型邻接矩阵 tempArcs

（6）将数据读入 tempVex 和 tempArcs

（7）根据 tempArcs 获得拓扑序列 toposeries

（8）根据 toposeries 录入顶点顺序表

（9）根据 toposeries 从下至上设置邻接下三角阵和顶点顺序表的权值（具体见代码，注释很详细）

（10）释放 tempVex 和 tempArcs

另外，增加顶点算法只需，在顶点顺序表末尾添加顶点、在弧邻接下三角矩阵最低端添加弧，再执行第（9）步即可（详见代码）。

3 实验结果

数据： 为了更好的实验，理想化数据。在附录 VertexData1.txt 和 ArcData1.txt 中记录了 10 篇虚构的 market-making 领域的论文基础论文，另外为测试增加顶点功能，添加 2 篇论文。写入结果放在 VertexData2.txt 和 ArcData2.txt 中。Citations-Network 类声明文件如下：

```
class CNClass
{
public:
        CNClass(char* VerticesFilename, char* ArcsFilename);
        ~CNClass();
        void NewPaper();
        void Withdraw(int index);
        void Modify(int index);
        int SearchPaper(std::string papersearch);
        int SearchAuthor(std::string authorsearch);
        int SearchPaperAuthor(std::string papersearch,
        std::string authorsearch);
        void DFS(int index);
        void PrintAll();
private:
        void InputData(char* VerticesFilename, char* ArcsFilename);
        int numVertices;
        double arcs[(1 + maxsize) / 2 * maxsize]; //compressed type
        VertexNode Vertices[maxsize];
};
```

3.1 构造函数（读取和建立）与析构函数（写入）实验结果

给定图：

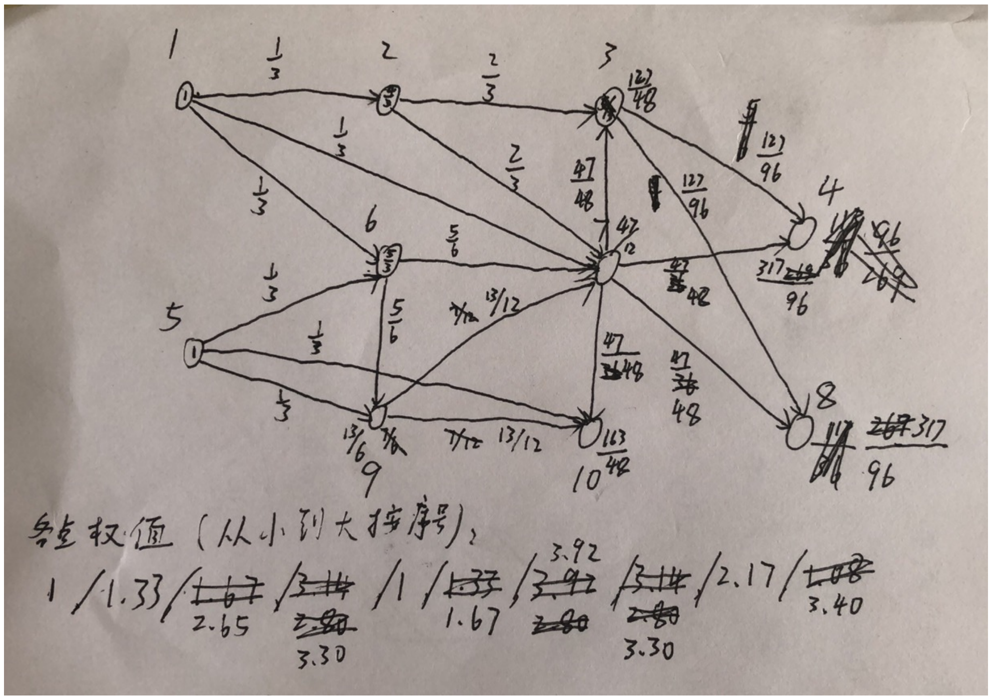


图 1: 第一个图 DAG1

Listing 1: VertexData1.txt

```
10
Market_Maker
Nicholas
1
```

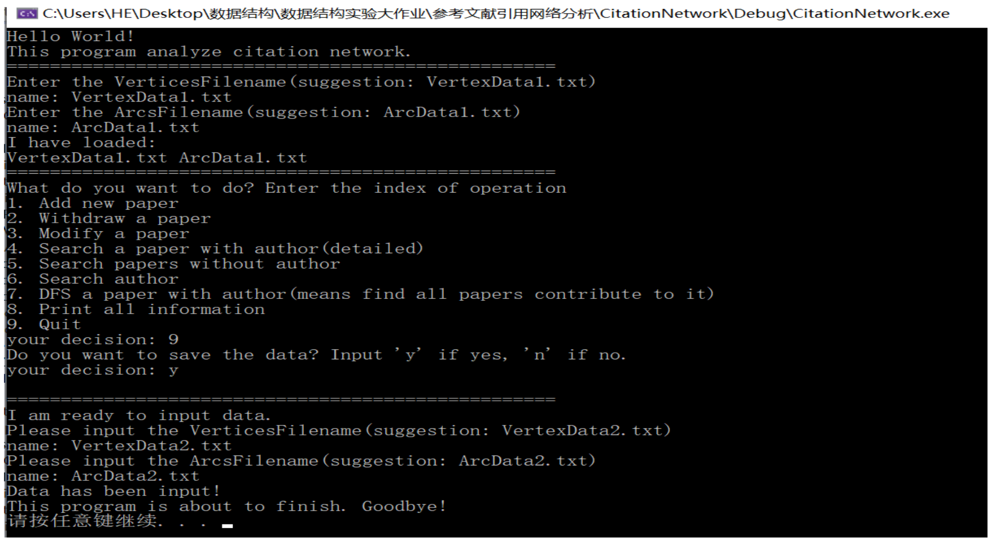
```
#
Market_Maker
Tung_Chan
1
#
Market_Making
Yagna
1
#
Multi-Agent
Patel
1
#
Reinforcement_Learning
Knuth
1
#
High-frequency
Nicholas
1
#
Market_making
Lim
1
#
Reinforcement_Learning
 Dijstra
1
#
Machine_Learning
Lim
1
#
```

```
High-frequency
Tom
1
#
```

Listing 2: ArcData1.txt

```
10
0
0 1 0 0 0 1 1 0 0 0
0 0 1 0 0 0 1 0 0 0
0 0 0 1 0 0 0 1 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 1 0 0 1 1
0 0 0 0 0 0 1 0 1 0
0 0 1 1 0 0 0 1 0 1
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 1 0 0 1
0 0 0 0 0 0 0 0 0 0
```

实验结果（读取未经过拓扑排序和设置权重的图）：


C:\Users\HE\Desktop\数据结构\数据结构实验大作业\参考文献引用网络分析\CitationNetwork\Debug\CitationNetwork.exe

```
Hello World!
This program analyze citation network.
==================================================
Enter the VerticesFilename(suggestion: VertexData1.txt)
name: VertexData1.txt
Enter the ArcsFilename(suggestion: ArcData1.txt)
name: ArcData1.txt
I have loaded:
VertexData1.txt ArcData1.txt
==================================================
What do you want to do? Enter the index of operation
1. Add new paper
2. Withdraw a paper
3. Modify a paper
4. Search a paper with author(detailed)
5. Search papers without author
6. Search author
7. DFS a paper with author(means find all papers contribute to it)
8. Print all information
9. Quit
your decision: 9
Do you want to save the data? Input 'y' if yes, 'n' if no.
your decision: y

==================================================
I am ready to input data.
Please input the VerticesFilename(suggestion: VertexData2.txt)
name: VertexData2.txt
Please input the ArcsFilename(suggestion: ArcData2.txt)
name: ArcData2.txt
Data has been input!
This program is about to finish. Goodbye!
请按任意键继续. . .
```

图 2: 实验结果控制台

Listing 3: VertexData2.txt

```
10
Multi−Agent
Patel
3.30208
#
Reinforcement_Learning
 Dijstra
3.30208
#
Market_Making
Yagna
2.64583
#
High−frequency
Tom
3.39583
```

```
#
Market_making
Lim
3.91667
#
Market_Maker
Tung_Chan
1.33333
#
Machine_Learning
Lim
2.16667
#
High-frequency
Nicholas
1.66667
#
Market_Maker
Nicholas
1
#
Reinforcement_Learning
Knuth
1
#
```

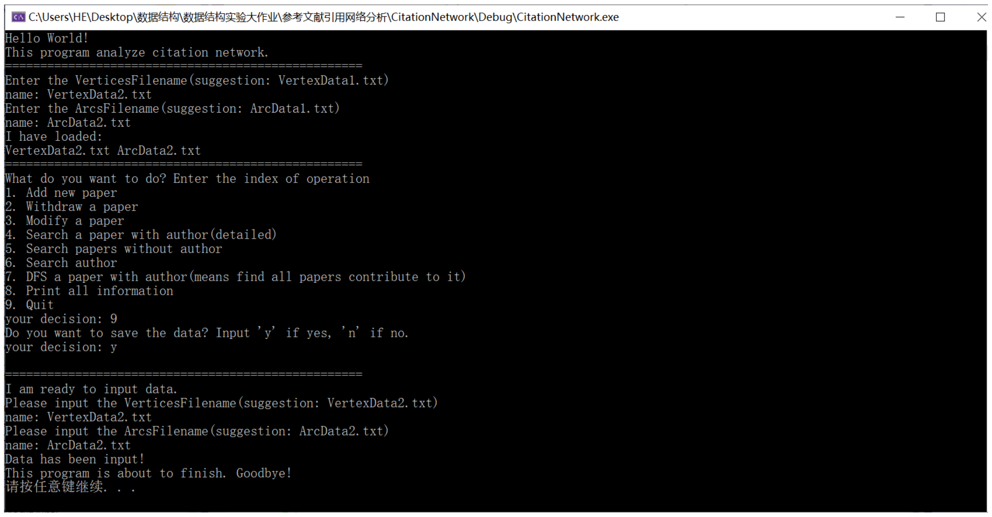
Listing 4: ArcData2.txt

```
10
1
0
0 0
1.32292 1.32292 0
0 0 0 0
```

```
0.979167 0.979167 0.979167 0.979167 0
0 0 0.666667 0 0.666667 0
0 0 0 1.08333 1.08333 0 0
0 0 0 0 0.833333 0 0.833333 0
0 0 0 0 0.333333 0.333333 0 0.333333 0
0 0 0 0.333333 0 0 0.333333 0.333333 0 0
```

实验分析：读取未经过排序和设置权重的数据，输出结果经过比较完全正确！

实验结果（读取已经过拓扑排序和设置权重的图）：



```
C:\Users\HE\Desktop\数据结构\数据结构实验大作业\参考文献引用网络分析\CitationNetwork\Debug\CitationNetwork.exe
Hello World!
This program analyze citation network.
==================================================
Enter the VerticesFilename(suggestion: VertexData1.txt)
name: VertexData2.txt
Enter the ArcsFilename(suggestion: ArcData1.txt)
name: ArcData2.txt
I have loaded:
VertexData2.txt ArcData2.txt
==================================================
What do you want to do? Enter the index of operation
1. Add new paper
2. Withdraw a paper
3. Modify a paper
4. Search a paper with author(detailed)
5. Search papers without author
6. Search author
7. DFS a paper with author(means find all papers contribute to it)
8. Print all information
9. Quit
your decision: 9
Do you want to save the data? Input 'y' if yes, 'n' if no.
your decision: y

==================================================
I am ready to input data.
Please input the VerticesFilename(suggestion: VertexData2.txt)
name: VertexData2.txt
Please input the ArcsFilename(suggestion: ArcData2.txt)
name: ArcData2.txt
Data has been input!
This program is about to finish. Goodbye!
请按任意键继续. . .
```

图 3: 实验结果控制台

Listing 5: VertexData2.txt

```
10
Multi−Agent
Patel
3.30208
#
Reinforcement_Learning
```

```
Dijstra
3.30208
#
Market_Making
Yagna
2.64583
#
High-frequency
Tom
3.39583
#
Market_making
Lim
3.91667
#
Market_Maker
Tung_Chan
1.33333
#
Machine_Learning
Lim
2.16667
#
High-frequency
Nicholas
1.66667
#
Market_Maker
Nicholas
1
#
Reinforcement_Learning
Knuth
```

```
1
#
```

实验分析：读取已经过排序和设置权重的数据，输出结果经过比较完全正确！

3.2 NewPaper 函数实验结果

给定图：

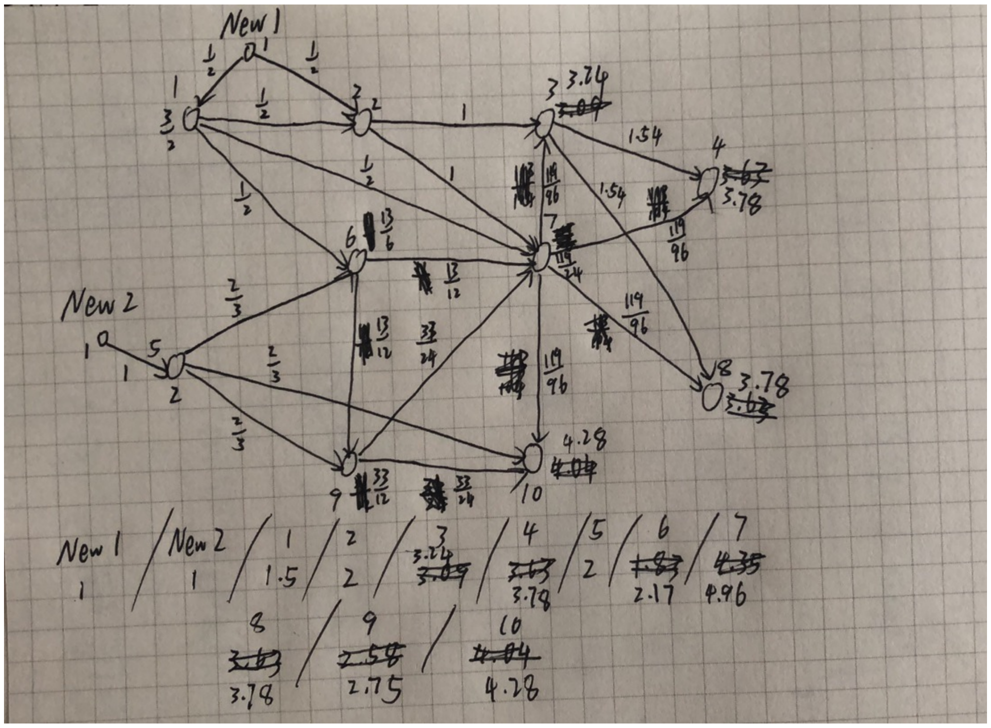
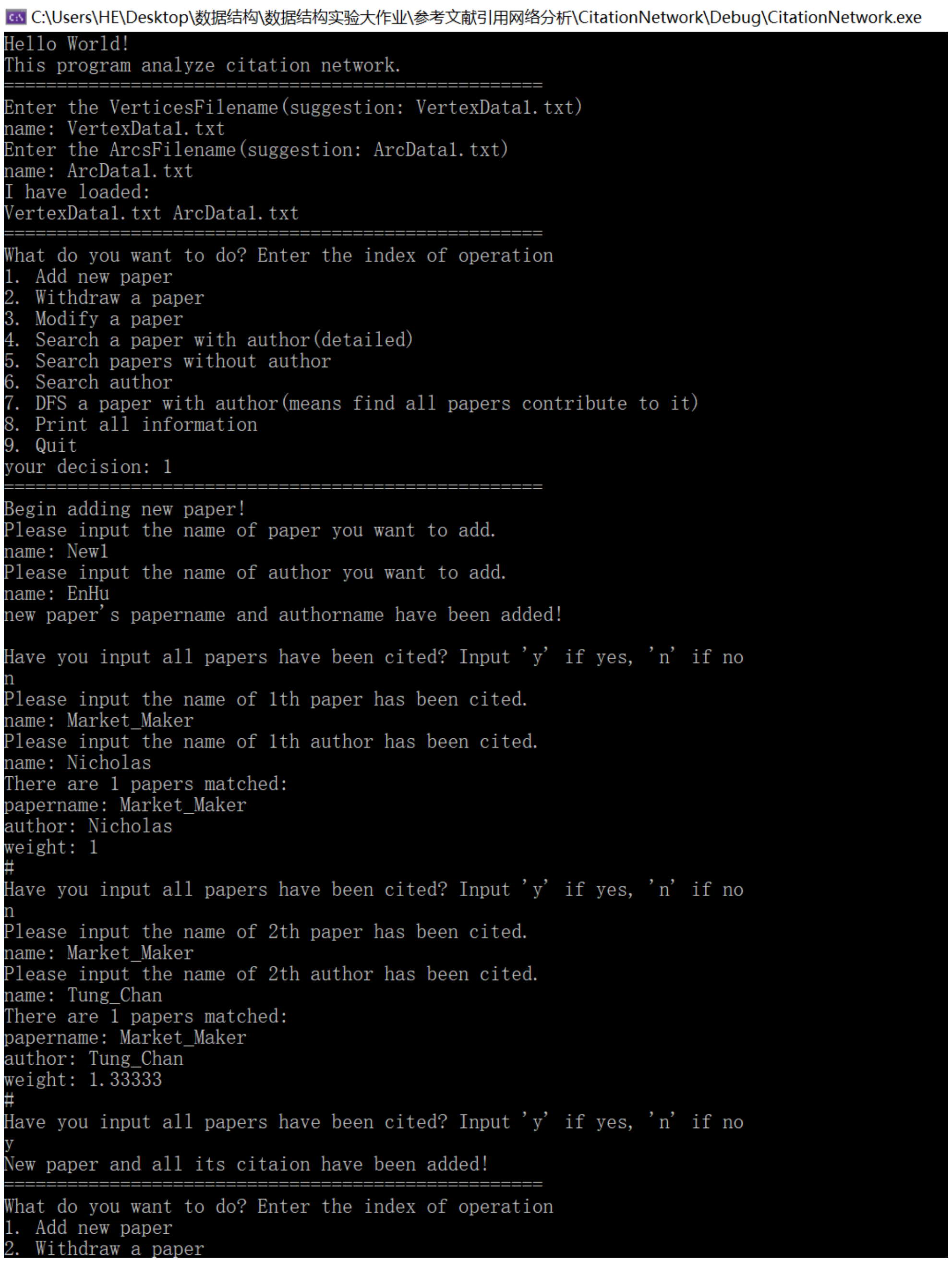


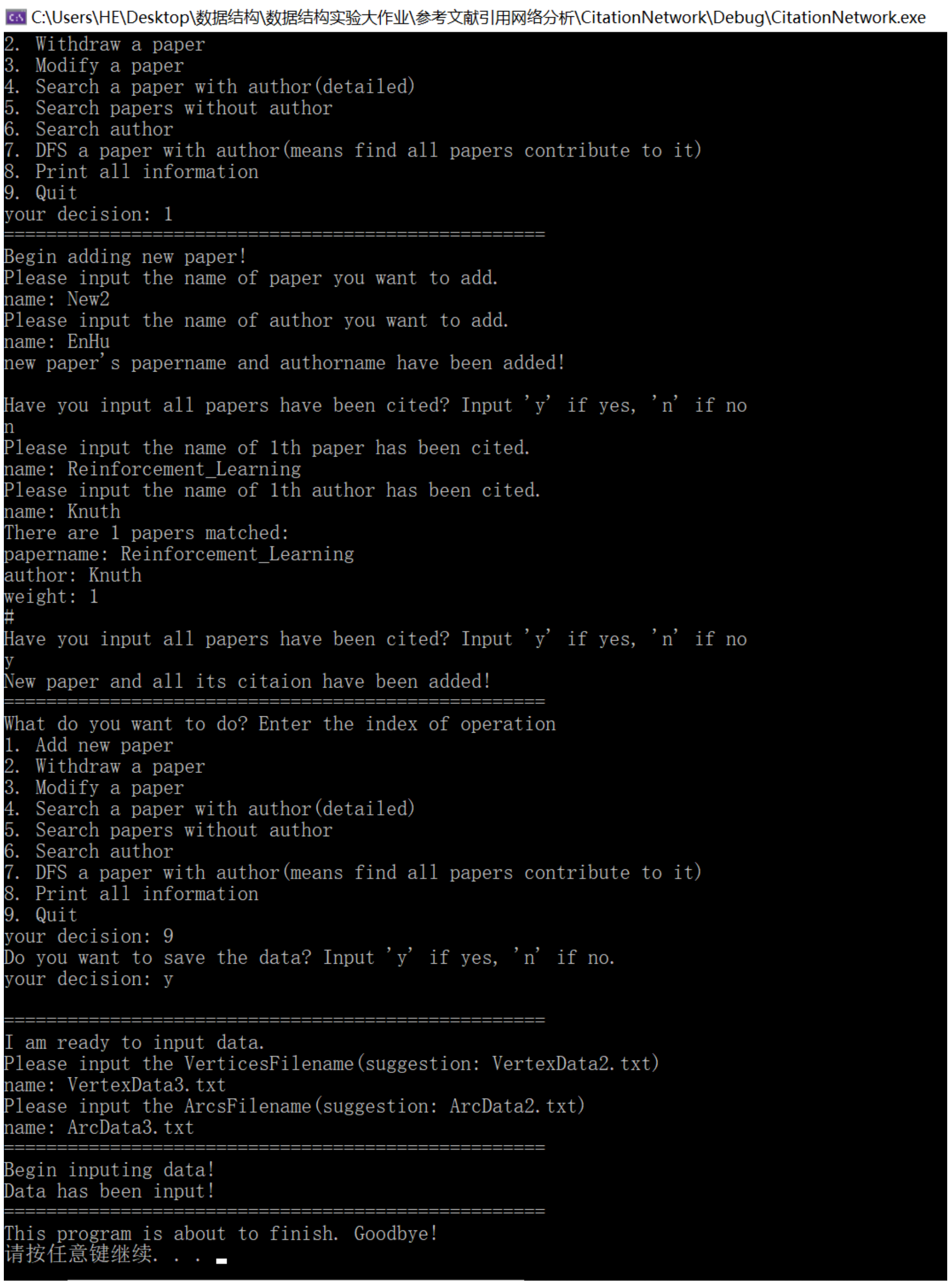
图 4: 在第一个图 DAG1 中增加顶点 New1 和 New2

实验结果（更改后的数据存储于 VertexData3.txt 和 ArcData3.txt 中）：



```
C:\Users\HE\Desktop\数据结构\数据结构实验大作业\参考文献引用网络分析\CitationNetwork\Debug\CitationNetwork.exe
Hello World!
This program analyze citation network.
===================================================
Enter the VerticesFilename(suggestion: VertexData1.txt)
name: VertexData1.txt
Enter the ArcsFilename(suggestion: ArcData1.txt)
name: ArcData1.txt
I have loaded:
VertexData1.txt ArcData1.txt
===================================================
What do you want to do? Enter the index of operation
1. Add new paper
2. Withdraw a paper
3. Modify a paper
4. Search a paper with author(detailed)
5. Search papers without author
6. Search author
7. DFS a paper with author(means find all papers contribute to it)
8. Print all information
9. Quit
your decision: 1
===================================================
Begin adding new paper!
Please input the name of paper you want to add.
name: New1
Please input the name of author you want to add.
name: EnHu
new paper's papername and authorname have been added!

Have you input all papers have been cited? Input 'y' if yes, 'n' if no
n
Please input the name of 1th paper has been cited.
name: Market_Maker
Please input the name of 1th author has been cited.
name: Nicholas
There are 1 papers matched:
papername: Market_Maker
author: Nicholas
weight: 1
#
Have you input all papers have been cited? Input 'y' if yes, 'n' if no
n
Please input the name of 2th paper has been cited.
name: Market_Maker
Please input the name of 2th author has been cited.
name: Tung_Chan
There are 1 papers matched:
papername: Market_Maker
author: Tung_Chan
weight: 1.33333
#
Have you input all papers have been cited? Input 'y' if yes, 'n' if no
y
New paper and all its citaion have been added!
===================================================
What do you want to do? Enter the index of operation
1. Add new paper
2. Withdraw a paper
```

图 5: 发表新论文 New1 和 New2



C:\Users\HE\Desktop\数据结构\数据结构实验大作业\参考文献引用网络分析\CitationNetwork\Debug\CitationNetwork.exe

```
2. Withdraw a paper
3. Modify a paper
4. Search a paper with author(detailed)
5. Search papers without author
6. Search author
7. DFS a paper with author(means find all papers contribute to it)
8. Print all information
9. Quit
your decision: 1
===================================================
Begin adding new paper!
Please input the name of paper you want to add.
name: New2
Please input the name of author you want to add.
name: EnHu
new paper's papername and authorname have been added!

Have you input all papers have been cited? Input 'y' if yes, 'n' if no
n
Please input the name of 1th paper has been cited.
name: Reinforcement_Learning
Please input the name of 1th author has been cited.
name: Knuth
There are 1 papers matched:
papername: Reinforcement_Learning
author: Knuth
weight: 1
#
Have you input all papers have been cited? Input 'y' if yes, 'n' if no
y
New paper and all its citaion have been added!
===================================================
What do you want to do? Enter the index of operation
1. Add new paper
2. Withdraw a paper
3. Modify a paper
4. Search a paper with author(detailed)
5. Search papers without author
6. Search author
7. DFS a paper with author(means find all papers contribute to it)
8. Print all information
9. Quit
your decision: 9
Do you want to save the data? Input 'y' if yes, 'n' if no.
your decision: y

===================================================
I am ready to input data.
Please input the VerticesFilename(suggestion: VertexData2.txt)
name: VertexData3.txt
Please input the ArcsFilename(suggestion: ArcData2.txt)
name: ArcData3.txt
===================================================
Begin inputing data!
Data has been input!
===================================================
This program is about to finish. Goodbye!
请按任意键继续. . .
```

图 6: 发表新论文 New1 和 New2

Listing 6: VertexData3.txt

```
12
Multi-Agent
 Patel
3.85938
#
Reinforcement_Learning
 Dijstra
3.85938
#
Market_Making
Yagna
3.23958
#
High-frequency
Tom
4.28125
#
Market_making
Lim
4.95833
#
Market_Maker
Tung_Chan
2
#
Machine_Learning
Lim
2.75
#
High-frequency
 Nicholas
2.16667
```

```
#
Market_Maker
Nicholas
1.5
#
Reinforcement_Learning
Knuth
2
#
New1
EnHu
1
#
New2
EnHu
1
#
```

结果分析：手算精确值完全符合！估计值误差在理想范围内！

3.3 Modify 和 PrintAll 函数实验结果

给定图仍为第一张图。

实验结果（更改后的数据存储于 VertexData3.txt 和 ArcData3.txt 中）：

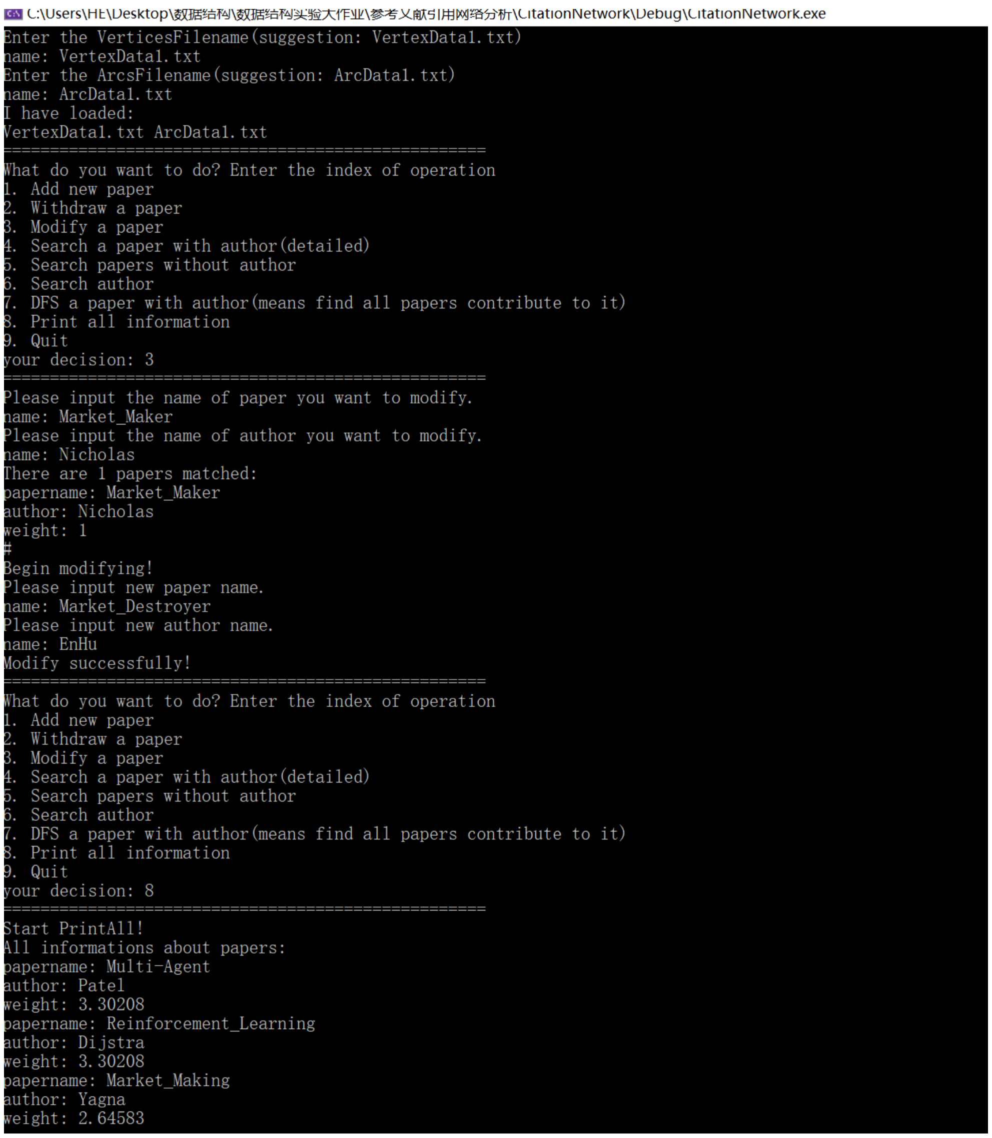
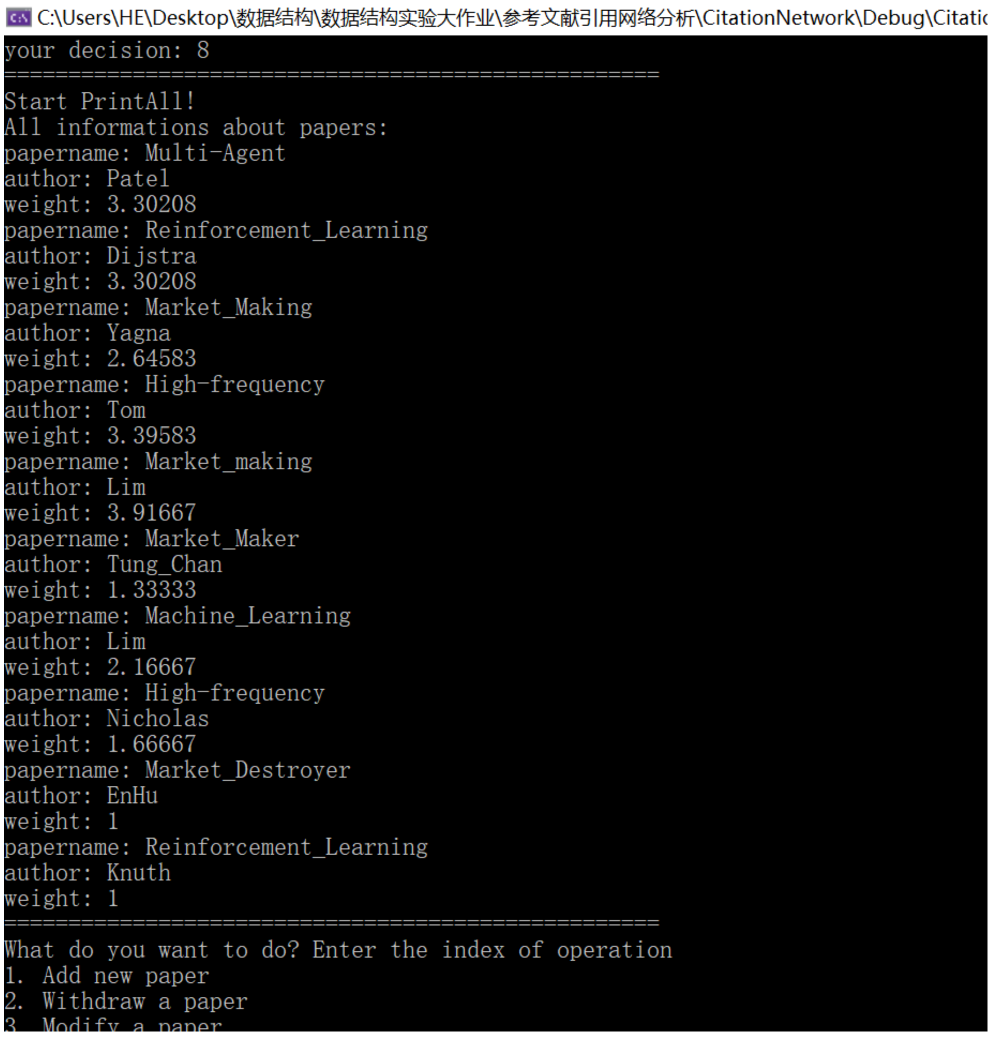

```
C:\Users\HE\Desktop\数据结构\数据结构实验大作业\参考文献引用网络分析\CitationNetwork\Debug\CitationNetwork.exe
Enter the VerticesFilename(suggestion: VertexData1.txt)
name: VertexData1.txt
Enter the ArcsFilename(suggestion: ArcData1.txt)
name: ArcData1.txt
I have loaded:
VertexData1.txt ArcData1.txt
==================================================
What do you want to do? Enter the index of operation
1. Add new paper
2. Withdraw a paper
3. Modify a paper
4. Search a paper with author(detailed)
5. Search papers without author
6. Search author
7. DFS a paper with author(means find all papers contribute to it)
8. Print all information
9. Quit
your decision: 3
==================================================
Please input the name of paper you want to modify.
name: Market_Maker
Please input the name of author you want to modify.
name: Nicholas
There are 1 papers matched:
papername: Market_Maker
author: Nicholas
weight: 1
#
Begin modifying!
Please input new paper name.
name: Market_Destroyer
Please input new author name.
name: EnHu
Modify successfully!
==================================================
What do you want to do? Enter the index of operation
1. Add new paper
2. Withdraw a paper
3. Modify a paper
4. Search a paper with author(detailed)
5. Search papers without author
6. Search author
7. DFS a paper with author(means find all papers contribute to it)
8. Print all information
9. Quit
your decision: 8
==================================================
Start PrintAll!
All informations about papers:
papername: Multi-Agent
author: Patel
weight: 3.30208
papername: Reinforcement_Learning
author: Dijstra
weight: 3.30208
papername: Market_Making
author: Yagna
weight: 2.64583
```

图 7: Modify 和 PrintAll 函数实验结果控制台 1



```
C:\Users\HE\Desktop\数据结构\数据结构实验大作业\参考文献引用网络分析\CitationNetwork\Debug\Citatio
your decision: 8
====================================================
Start PrintAll!
All informations about papers:
papername: Multi-Agent
author: Patel
weight: 3.30208
papername: Reinforcement_Learning
author: Dijstra
weight: 3.30208
papername: Market_Making
author: Yagna
weight: 2.64583
papername: High-frequency
author: Tom
weight: 3.39583
papername: Market_making
author: Lim
weight: 3.91667
papername: Market_Maker
author: Tung_Chan
weight: 1.33333
papername: Machine_Learning
author: Lim
weight: 2.16667
papername: High-frequency
author: Nicholas
weight: 1.66667
papername: Market_Destroyer
author: EnHu
weight: 1
papername: Reinforcement_Learning
author: Knuth
weight: 1
====================================================
What do you want to do? Enter the index of operation
1. Add new paper
2. Withdraw a paper
3. Modify a paper
```

图 8: Modify 和 PrintAll 函数实验结果控制台 2

结果分析：Modify 成功，PrintAll 运行正常。

3.4 Search 函数实验结果

给定图仍为第一张图。

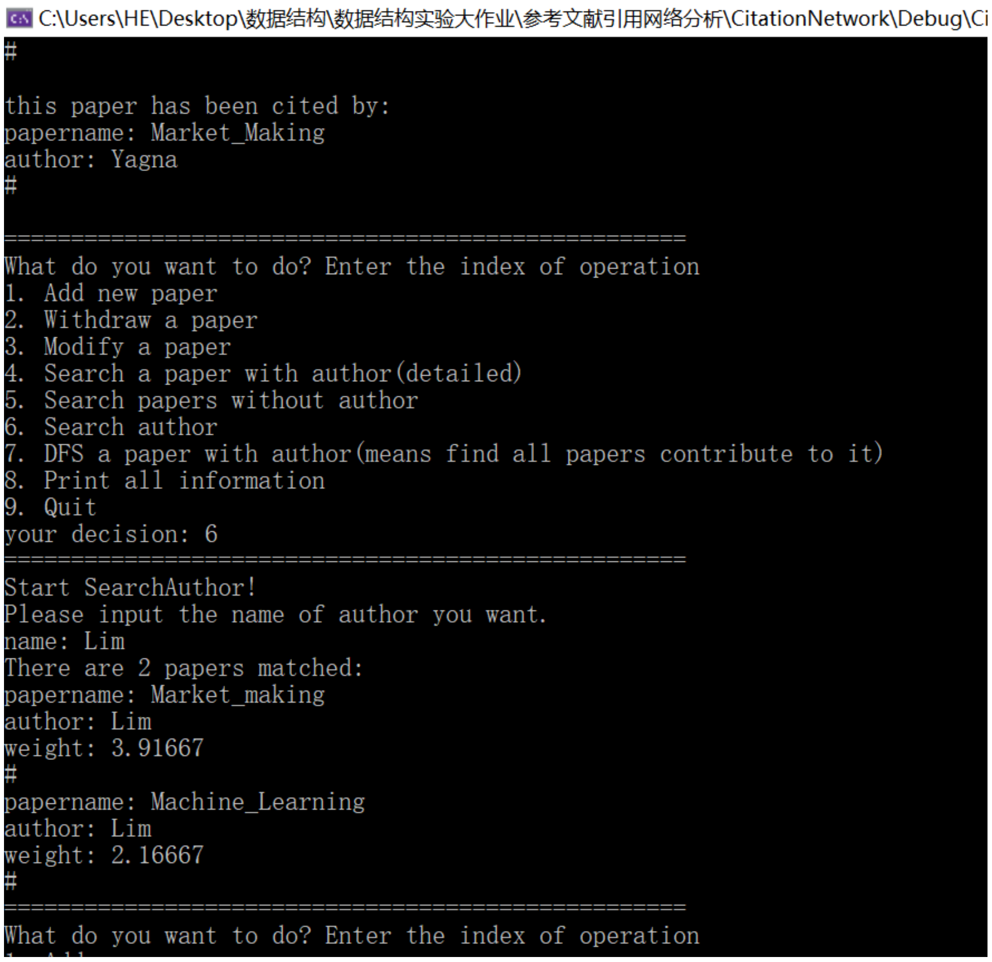
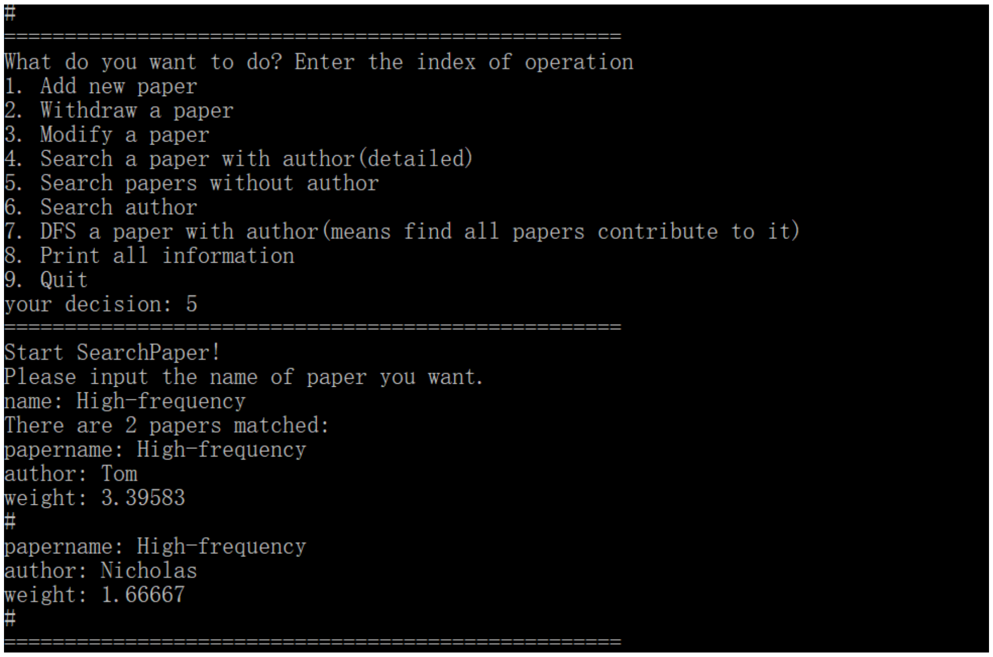
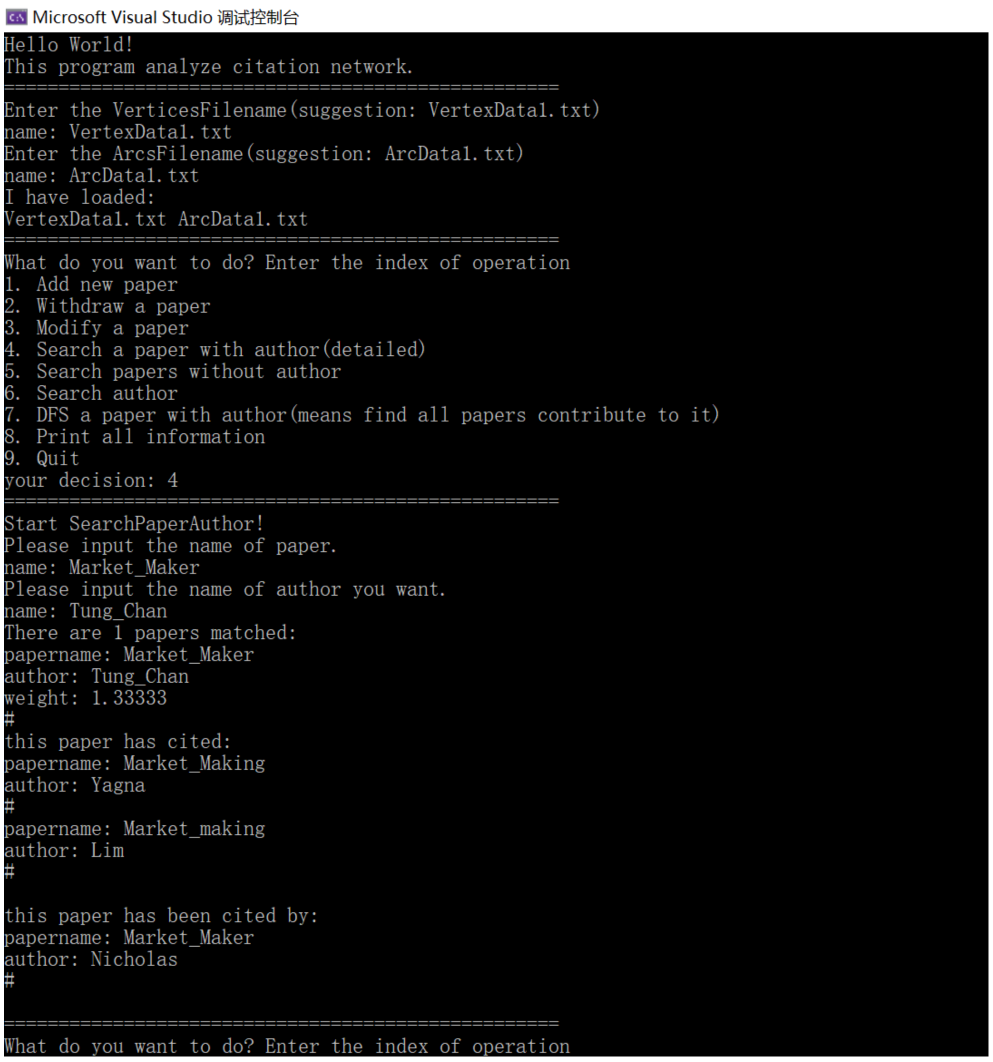

```
C:\Users\HE\Desktop\数据结构\数据结构实验大作业\参考文献引用网络分析\CitationNetwork\Debug\Ci
#

this paper has been cited by:
papername: Market_Making
author: Yagna
#

====================================================
What do you want to do? Enter the index of operation
1. Add new paper
2. Withdraw a paper
3. Modify a paper
4. Search a paper with author(detailed)
5. Search papers without author
6. Search author
7. DFS a paper with author(means find all papers contribute to it)
8. Print all information
9. Quit
your decision: 6
====================================================
Start SearchAuthor!
Please input the name of author you want.
name: Lim
There are 2 papers matched:
papername: Market_making
author: Lim
weight: 3.91667
#
papername: Machine_Learning
author: Lim
weight: 2.16667
#
====================================================
What do you want to do? Enter the index of operation
```

图 9: SearchAuthor:



```
#
====================================================
What do you want to do? Enter the index of operation
1. Add new paper
2. Withdraw a paper
3. Modify a paper
4. Search a paper with author(detailed)
5. Search papers without author
6. Search author
7. DFS a paper with author(means find all papers contribute to it)
8. Print all information
9. Quit
your decision: 5
====================================================
Start SearchPaper!
Please input the name of paper you want.
name: High-frequency
There are 2 papers matched:
papername: High-frequency
author: Tom
weight: 3.39583
#
papername: High-frequency
author: Nicholas
weight: 1.66667
#
====================================================
```

图 10: SearchPaper：



```
Microsoft Visual Studio 调试控制台
Hello World!
This program analyze citation network.
===================================================
Enter the VerticesFilename(suggestion: VertexData1.txt)
name: VertexData1.txt
Enter the ArcsFilename(suggestion: ArcData1.txt)
name: ArcData1.txt
I have loaded:
VertexData1.txt ArcData1.txt
===================================================
What do you want to do? Enter the index of operation
1. Add new paper
2. Withdraw a paper
3. Modify a paper
4. Search a paper with author(detailed)
5. Search papers without author
6. Search author
7. DFS a paper with author(means find all papers contribute to it)
8. Print all information
9. Quit
your decision: 4
===================================================
Start SearchPaperAuthor!
Please input the name of paper.
name: Market_Maker
Please input the name of author you want.
name: Tung_Chan
There are 1 papers matched:
papername: Market_Maker
author: Tung_Chan
weight: 1.33333
#
this paper has cited:
papername: Market_Making
author: Yagna
#
papername: Market_making
author: Lim
#

this paper has been cited by:
papername: Market_Maker
author: Nicholas
#

===================================================
What do you want to do? Enter the index of operation
```

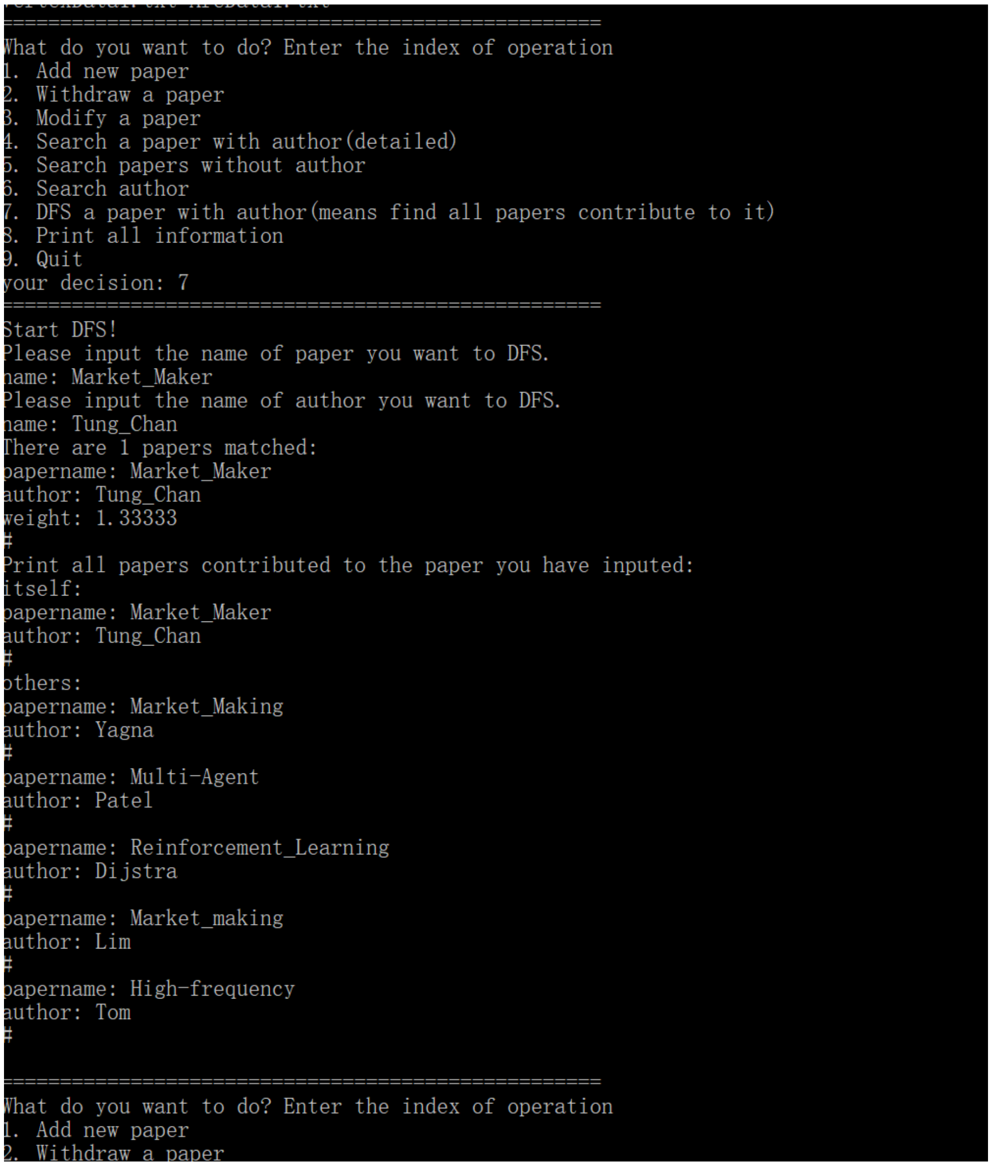
图 11: SearchPaperAuthor：

结果分析：三个查找函数全部运行正常

3.5 DFS 函数实验结果

给定图仍为第一张图。

实验结果：



```
vertexdata.txt arcdata.txt
===================================================
What do you want to do? Enter the index of operation
1. Add new paper
2. Withdraw a paper
3. Modify a paper
4. Search a paper with author(detailed)
5. Search papers without author
6. Search author
7. DFS a paper with author(means find all papers contribute to it)
8. Print all information
9. Quit
your decision: 7
===================================================
Start DFS!
Please input the name of paper you want to DFS.
name: Market_Maker
Please input the name of author you want to DFS.
name: Tung_Chan
There are 1 papers matched:
papername: Market_Maker
author: Tung_Chan
weight: 1.33333
#
Print all papers contributed to the paper you have inputed:
itself:
papername: Market_Maker
author: Tung_Chan
#
others:
papername: Market_Making
author: Yagna
#
papername: Multi-Agent
author: Patel
#
papername: Reinforcement_Learning
author: Dijstra
#
papername: Market_making
author: Lim
#
papername: High-frequency
author: Tom
#

===================================================
What do you want to do? Enter the index of operation
1. Add new paper
2. Withdraw a paper
```

图 12: DFS：

结果分析：DFS 正确！

4 讨论（算法分析）

优势：

（1）所有函数非递归，速度较快

（2）参考 PageRank 算法的思想，合理确定文献影响力（顶点权重）

（3）用拓扑排序，压缩存储下三角邻接矩阵，空间复杂度低

（4）能够执行文件的读取和写入

（5）允许查找多个同名论文或多个相同作者的论文

（6）能够查找论文的引用和被引用关系

不足：

（1）由于研究时间有限，仍未能设计有效的删除顶点的算法

（2）所有的查找算法只能搜索全名，不能搜索关键词

（3）某文献不同引用的引用强度是一样的，没有区分强引用和弱引用

（4）没有时效性，即论文发表时间不影响权重（影响力）

5 结论

本文为建立参考文献引用网络分析系统提供了一个基本的框架和思路，体现了作者数据结构与算法能力。

参考文献

https://en.wikipedia.org/wiki/PageRank

A 附录 1：代码

Listing 7: **CNClass.h**

```cpp
//Copyright (c) 2020 En Hu. All rights reserved.

#pragma once
#include<iostream>
#include<string>
#include<fstream>

#ifndef MAXSIZE
#define MAXSIZE
const int maxsize = 20;
#endif // !MAXSIZE

struct VertexNode
{
        std::string paper;
        std::string author;
        double weight;
        int indegree, outdegree;
};

class CNClass
{
public:
        CNClass(char* VerticesFilename, char* ArcsFilename);
        ~CNClass();
        void NewPaper();
        void Withdraw(int index);
        void Modify(int index);
        int SearchPaper(std::string papersearch);
        int SearchAuthor(std::string authorsearch);
        int SearchPaperAuthor(std::string papersearch, std::string
             authorsearch, bool printcite);
        void DFS(int index);
        void PrintAll();
private:
        void InputData(char* VerticesFilename, char* ArcsFilename);
        int numVertices;
        double arcs[(1 + maxsize) / 2 * maxsize];//compressed type
        VertexNode Vertices[maxsize];
};
```

Listing 8: **CitationNetwork.cpp**

```
//Copyright (c) 2020 En Hu. All rights reserved.

#include <iostream>
#include"CNClass.h"

int main()
{
    using namespace std;
    cout << "Hello World!\n";
    cout << "This program analyze citation network.\n";
        char split[] = "====================================================="
            ;
        cout << split << endl;
        //char VerticesFilename[] = "VertexData1.txt", ArcsFilename[] = "
            ArcData1.txt";
        char VerticesFilename[50], ArcsFilename[50];
        cout << "Enter the VerticesFilename(suggestion: VertexData1.txt)" <<
             endl << "name: ";
        cin >> VerticesFilename;
        cout << "Enter the ArcsFilename(suggestion: ArcData1.txt)" << endl
            << "name: ";
        cin >> ArcsFilename;
        CNClass CN1(VerticesFilename, ArcsFilename);
        cout << split << endl;
        bool finished = false;
        while (!finished)
        {
                int id_operation;
                cout << "What do you want to do? Enter the index of
                    operation" << endl;
                cout << "1. Add new paper" << endl
```

```
                        << "2. Withdraw a paper" << endl
                        << "3. Modify a paper" << endl
                        << "4. Search a paper with author(detailed)" << endl
                        << "5. Search papers without author" << endl
                        << "6. Search author" << endl
                        << "7. DFS a paper with author(means find all papers
                             contribute to it)" << endl
                        << "8. Print all information" << endl
                        << "9. Quit" << endl;
                cout << "your decision: ";
                cin >> id_operation;
                string papername, authorname; int index;
                switch (id_operation)
                {
                case 1:
                        cout << split << endl;
                        cout << "Begin adding new paper! " << endl;
                        CN1.NewPaper();
                        cout << split << endl;
                        break;
                case 2:
                        char decision;
                        cout << "Do you really want to withdraw the paper?
                            Input 'y' if yes, else if no." << endl;
                        cin >> decision;
                        if (decision != 'y')    break;
                        else
                        {
                                cout << split << endl;
                                cout << "Begin withdrawing! " << endl;
                                cout << "Please input the name of paper you
                                    want to withdraw. " << endl
                                        << "name: ";
                                cin >> papername;
                                cout << "Please input the name of author you
                                     want to withdraw. " << endl
                                        << "name: ";
                                cin >> authorname;
                index = CN1.SearchPaperAuthor(papername,
                    authorname, false);
                if (index == -1)        cout << "I cannot
                    find the paper. " << endl;
                else if (index > maxsize)       cout << "
                    There is more than one paper. I have
                    not solve this problem now. " << endl;
                else CN1.Withdraw(index);
                cout << split << endl;
                break;
            }
        case 3:
            cout << split << endl;
            cout << "Please input the name of paper you want to
                modify. " << endl
                << "name: ";
            cin >> papername;
            cout << "Please input the name of author you want to
                 modify. " << endl
                << "name: ";
            cin >> authorname;
            index = CN1.SearchPaperAuthor(papername, authorname,
                 false);
            if (index == -1)        cout << "I cannot find the
                paper. " << endl;
            else if (index > maxsize)       cout << "There is
                more than one paper. I have not solve this
                problem now. " << endl;
            else CN1.Modify(index);
            cout << split << endl;
            break;
        case 4:
            cout << split << endl;
            cout << "Start SearchPaperAuthor! " << endl;
            cout << "Please input the name of paper. " << endl
                << "name: ";
            cin >> papername;
            cout << "Please input the name of author you want. "
                 << endl
```

```
                                << "name: ";
                        cin >> authorname;
                        CN1.SearchPaperAuthor(papername, authorname, true);
                        cout << split << endl;
                        break;
                case 5:
                        cout << split << endl;
                        cout << "Start SearchPaper! " << endl;
                        cout << "Please input the name of paper you want. "
                            << endl
                        << "name: ";
                        cin >> papername;
                        CN1.SearchPaper(papername);
                        cout << split << endl;
                        break;
                case 6:
                        cout << split << endl;
                        cout << "Start SearchAuthor! " << endl;
                        cout << "Please input the name of author you want. "
                             << endl
                                << "name: ";
                        cin >> authorname;
                        CN1.SearchAuthor(authorname);
                        cout << split << endl;
                        break;
                case 7:
                        cout << split << endl;
                        cout << "Start DFS! " << endl;
                        cout << "Please input the name of paper you want to
                            DFS. " << endl
                                << "name: ";
                        cin >> papername;
                        cout << "Please input the name of author you want to
                             DFS. " << endl
                                << "name: ";
                        cin >> authorname;
                        index = CN1.SearchPaperAuthor(papername, authorname,
                             false);
                        if (index == -1)        cout << "I cannot find the
                            paper. " << endl;
                        else if (index > maxsize)       cout << "There is
                            more than one paper. I have not solve this
                            problem now. " << endl;
                        else CN1.DFS(index);
                        cout << split << endl;
                        break;
                case 8:
                        cout << split << endl;
                        cout << "Start PrintAll! " << endl;
                        CN1.PrintAll();
                        cout << split << endl;
                        break;
                case 9:
                        finished = true;
                        break;
                default:
                        cout << "Wrong index! Please enter again. " << endl;
                        break;
                }
        }
}

// 运行程序: Ctrl + F5 或调试 “开始执行不调试 >()” 菜单
// 调试程序: F5 或调试 “开始调试” 菜单 >
```

Listing 9: **CNClass.cpp**

```
//Copyright (c) 2020 En Hu. All rights reserved.
#include "CNClass.h"

CNClass::CNClass(char* VerticesFilename, char* ArcsFilename)
{
        using namespace std;
        //load files
        fstream VexFile, ArcFile;
        cout << "I have loaded: " << endl;
        cout << VerticesFilename << ' ' << ArcsFilename << endl;
```

```
    VexFile.open(VerticesFilename);
    ArcFile.open(ArcsFilename);
    //get number of vertices and the sign of is-sorted-and-weighted
    int t1, t2;//temporary int-type data
    VexFile >> t1; ArcFile >> t2;
    if (t1 != t2)   throw("number of Vertices is not equal");
    else if (t1 < 0)        throw("number of Vertices cannot less than 0
        ");
    else if (t1 > maxsize)  throw("overflow");
    else numVertices = t1;
    bool isSortedWeighted;
    ArcFile >> t1;
    switch (t1)
    {
    case 0:
        isSortedWeighted = false;
        break;
    case 1:
        isSortedWeighted = true;
        break;
    default:
        throw("error type of sign of is-triangular-matrix");
        break;
    }
    if (isSortedWeighted)//if the files has been sorted-and-weighted
    {
        //get data of paper, author, and weights
        string paper, author, finished; double weights;
        for (int i = 0; i < numVertices; i++)
        {
            VexFile >> paper;
            VexFile >> author;
            VexFile >> weights;
            Vertices[i].paper = paper;
            Vertices[i].author = author;
            Vertices[i].weight = weights;
            VexFile >> finished;
            if (finished == "#")    continue;
            else throw("error: the type of VexFile is wrong");
				}
				//get data of arcs
				for (int i = 0; i < (1 + numVertices) * numVertices / 2; i
				    ++)   ArcFile >> arcs[i];
				//initialize empty room
				for (int i = (1 + numVertices) * numVertices / 2; i <
				    maxsize; i++)     arcs[i] = 0;
		}
		else//if not, then start from scratch
		{
				//initialize arcs matrix
				for (int i = 0; i < (1 + maxsize) / 2 * maxsize; i++)
					arcs[i] = 0;
				VertexNode* tempVex = new VertexNode[numVertices];//
				    temporary vertexdata
				//temporaray arcdata
				bool** tempArcs = new bool* [numVertices];
				for (int i = 0; i < numVertices; i++)
					tempArcs[i] = new bool[numVertices];
				//get data of vertex
				string paper, author, finished; double weights;
				for (int i = 0; i < numVertices; i++)
				{
					VexFile >> paper;
					VexFile >> author;
					VexFile >> weights;
					tempVex[i].paper = paper;
					tempVex[i].author = author;
					tempVex[i].weight = 1;
					VexFile >> finished;
					if (finished == "#")    continue;
					else throw("error: the type of VexFile is wrong");
				}
				//get data of arcs
				for (int i = 0; i < numVertices; i++)
					for (int j = 0; j < numVertices; j++)
					{
						ArcFile >> t1;
						if (t1 > 0)
```

```
                    tempArcs[i][j] = true;
                else tempArcs[i][j] = false;
            }
        //get indegree
        for (int j = 0; j < numVertices; j++)
        {
            int count = 0;
            for (int i = 0; i < numVertices; i++)
                if (tempArcs[i][j])     count++;
            tempVex[j].indegree = count;
        }
        //toposort
        int* s = new int[numVertices];//stack, which stores index of vertices
        int* toposeries = new int[numVertices];//stores toposeries
        int top = -1, count = 0;
        for (int i = 0; i < numVertices; i++)
            if (tempVex[i].indegree == 0)
                s[++top] = i;
        while (top != -1)
        {
            t1 = s[top--];

            //cout << tempVex[t1];
            toposeries[count++] = t1;

            for (int j = 0; j < numVertices; j++)
                if (tempArcs[t1][j])
                {
                    tempVex[j].indegree--;
                    if (tempVex[j].indegree == 0)
                        s[++top] = j;
                }
        }
        if (count < numVertices)        throw("there is a ring");

        //int* inversetopo = new int[numVertices];//stores inverse of toposeries
        //for (int i = 0; i < numVertices; i++)
        //      inversetopo[toposeries[i]] = i;

        //input paper and author
        for (int i = 0; i < numVertices; i++)
        {
                Vertices[numVertices - 1 - i].paper = tempVex[
                    toposeries[i]].paper;
                Vertices[numVertices - 1 - i].author = tempVex[
                    toposeries[i]].author;
        }

        //weigh vertices and arcs
        for (int i = numVertices - 1; i >= 0; i--)
        {
                //sum of weights of inarcs
                double sum = 0;
                for (int j = i + 1; j < numVertices; j++)
                        sum = sum + arcs[(1 + j) * j / 2 + i];
                //weigh vertex
                Vertices[i].weight = sum + 1;
                //calculate outdegree
                Vertices[i].outdegree = 0;
                for (int j = 0; j < numVertices; j++)
                        if (tempArcs[toposeries[numVertices - i -
                            1]][j])
                                Vertices[i].outdegree++;
                //set the weights of outarcs
                for (int j = 0; j < i; j++)
                        if (tempArcs[toposeries[numVertices - i -
                            1]][toposeries[numVertices - j - 1]])
                                arcs[(1 + i) * i / 2 + j] = Vertices
                                    [i].weight / Vertices[i].
                                    outdegree;
        }

        VexFile.close(); ArcFile.close();
        //delete[] inversetopo;
        delete[] toposeries;
        delete[] tempVex;
```

```
                for (int i = 0; i < numVertices; i++)
                        delete[] tempArcs[i];
                delete[] tempArcs;
        }
}

CNClass::~CNClass()
{
        using namespace std;
        bool isdecided = false;
        while (!isdecided)
        {
                cout << "Do you want to save the data? Input 'y' if yes, 'n' if no. " << endl
                        << "your decision: ";
                char decision;
                cin >> decision;
                cout << endl;
                if (decision == 'y')
                {
                        char VerticesFilename[50], ArcsFilename[50];
                        char split[] = "===================================================";
                        cout << split << endl << "I am ready to input data. " << endl;
                        cout << "Please input the VerticesFilename(suggestion: VertexData2.txt)" << endl << "name: ";
                        cin >> VerticesFilename;
                        cout << "Please input the ArcsFilename(suggestion: ArcData2.txt)" << endl << "name: ";
                        cin >> ArcsFilename;
                        InputData(VerticesFilename, ArcsFilename);
                        isdecided = true;
                        cout << "This program is about to finish. Goodbye! " << endl;
                }
                else if (decision == 'n')
                {
                        isdecided = true;
                        cout << "This program is about to finish. Goodbye! " << endl;
                }
                else
                        cout << "Error decision! Please decided again. " << endl;
        }
        system("pause");
}

void CNClass::NewPaper()
{
        using namespace std;
        if (numVertices == maxsize)
        {
                cout << "overflow";
                return;
        }
        //add vertex
        string papername, authorname;
        cout << "Please input the name of paper you want to add. " << endl
                << "name: ";
        cin >> papername;
        cout << "Please input the name of author you want to add. " << endl
                << "name: ";
        cin >> authorname;
        Vertices[numVertices].paper = papername;
        Vertices[numVertices].author = authorname;
        Vertices[numVertices].weight = 1;
        cout << "new paper's papername and authorname have been added! " << endl << endl;
        //add arcs
        int count = 0;
        int citedid[maxsize];
        for (int i = 0; i < maxsize; i++)       citedid[i] = -1;
        bool finishedsign = false;
        while (!finishedsign)
```

```
    {
        char finished;
        cout << "Have you input all papers have been cited? Input 'y' if yes, 'n' if no" << endl;
        cin >> finished;
        if (finished == 'y')    finishedsign = true;
        else if (finished == 'n')
        {
            cout << "Please input the name of " << (count + 1) << "th paper has been cited. " << endl
                << "name: ";
            cin >> papername;
            cout << "Please input the name of " << (count + 1) << "th author has been cited. " << endl
                << "name: ";
            cin >> authorname;
            int i = SearchPaperAuthor(papername, authorname, false);
            if (i >= 0 && i < maxsize)
                citedid[count++] = i;
            else if (i == -1)
                cout << "I cannot find the paper you have inputed. " << endl;
            else
                cout << "More than one paper. I have not solve this problem. " << endl;
        }
        else cout << "Error type! Please input again. " << endl;
    }
    Vertices[numVertices].outdegree = count;
    numVertices++;
    for (int i = 0; i < numVertices - 1 && citedid[i] != -1; i++)
        //arcs[numVertices - 1][citedid[i]] = 1 / count;
        arcs[numVertices * (numVertices - 1) / 2 + citedid[i]] = 1.0 / double(count);

    //update weights of vertices and arcs
    for (int i = numVertices - 2; i >= 0; i--)
    {
```

```cpp
                //sum of weights of inarcs
                double sum = 0;
                for (int j = i + 1; j < numVertices; j++)
                        sum = sum + arcs[(1 + j) * j / 2 + i];
                //weigh vertex
                Vertices[i].weight = sum + 1;
                //calculate outdegree
                Vertices[i].outdegree = 0;
                for (int j = 0; j < i; j++)
                        if (arcs[(1 + i) * i / 2 + j] > 0)      Vertices[i].outdegree++;
                //set the weights of outarcs
                double outarcweight = Vertices[i].weight / Vertices[i].outdegree;
                for (int j = 0; j < i; j++)
                        if (arcs[(1 + i) * i / 2 + j] > 0)      arcs[(1 + i) * i / 2 + j] = outarcweight;
        }
        cout << "New paper and all its citaion have been added! " << endl;
}

void CNClass::InputData(char* VerticesFilename, char* ArcsFilename)
{
        using namespace std;
        fstream Vexfile, Arcfile;
        char split[] = "====================================================";
        cout << split << endl;
        cout << "Begin inputing data! " << endl;
        Vexfile.open(VerticesFilename); Arcfile.open(ArcsFilename);
        //input arcs data
        Arcfile << numVertices << '\n';
        Arcfile << 1 << '\n';
        for (int i = 0; i < numVertices; i++)
        {
                for (int j = 0; j <= i; j++)
                        Arcfile << arcs[(1 + i) * i / 2 + j] << ' ';
                Arcfile << '\n';
        }
```

```

	//input vex data
	Vexfile << numVertices << '\n';
	for (int i = 0; i < numVertices; i++)
	{
		Vexfile << Vertices[i].paper << '\n';
		Vexfile << Vertices[i].author << '\n';
		Vexfile << Vertices[i].weight << '\n';
		Vexfile << '#' << '\n';
	}
	cout << "Data has been input! " << endl;
	cout << split << endl;
}

void CNClass::Withdraw(int index)
{
	using namespace std;
	//start withdrawing
	cout << "Sorry! I have not design this algorithem. " << endl;
}

void CNClass::Modify(int index)
{
	using namespace std;
	cout << "Begin modifying! " << endl;
	string papername, authorname;
	cout << "Please input new paper name. " << endl
		<< "name: ";
	cin >> papername;
	cout << "Please input new author name. " << endl
		<< "name: ";
	cin >> authorname;
	Vertices[index].paper = papername;
	Vertices[index].author = authorname;
	cout << "Modify successfully! " << endl;
}

int CNClass::SearchPaper(std::string papersearch)
{
        using namespace std;
        int* id = new int[numVertices];
        for (int i = 0; i < numVertices; i++)   id[i] = -1;
        int count = 0;
        for (int i = 0; i < numVertices; i++)
                if (Vertices[i].paper == papersearch)
                        id[count++] = i;
        cout << "There are " << count << " papers matched: " << endl;
        for (int i = 0, j = 0; i < numVertices && id[j] != -1; i++)
                if (i == id[j])
                {
                        cout << "papername: " << Vertices[i].paper << '\n';
                        cout << "author: " << Vertices[i].author << '\n';
                        cout << "weight: " << Vertices[i].weight << '\n';
                        cout << '#' << '\n';
                        j++;
                }

        delete[] id;
        return count;
}

int CNClass::SearchAuthor(std::string authorsearch)
{
        using namespace std;
        int* id = new int[numVertices];
        for (int i = 0; i < numVertices; i++)   id[i] = -1;
        int count = 0;
        for (int i = 0; i < numVertices; i++)
                if (Vertices[i].author == authorsearch)
                        id[count++] = i;
        cout << "There are " << count << " papers matched: " << endl;
        for (int i = 0, j = 0; i < numVertices && id[j] != -1; i++)
                if (i == id[j])
                {
                        cout << "papername: " << Vertices[i].paper << '\n';
                        cout << "author: " << Vertices[i].author << '\n';
                        cout << "weight: " << Vertices[i].weight << '\n';
                        cout << '#' << '\n';
                        j++;
                }

        delete[] id;
        return count;

}

int CNClass::SearchPaperAuthor(std::string papersearch, std::string
     authorsearch, bool printcite)
{
        using namespace std;
        int* id = new int[numVertices];
        for (int i = 0; i < numVertices; i++)   id[i] = -1;
        int count = 0;
        for (int i = 0; i < numVertices; i++)
                if (Vertices[i].paper == papersearch && Vertices[i].author
                     == authorsearch)
                        id[count++] = i;
        cout << "There are " << count << " papers matched: " << endl;
        for (int i = 0, j = 0; i < numVertices && id[j] != -1; i++)
                if (i == id[j])
                {
                        cout << "papername: " << Vertices[i].paper << '\n';
                        cout << "author: " << Vertices[i].author << '\n';
                        cout << "weight: " << Vertices[i].weight << '\n';
                        cout << '#' << '\n';
                        j++;
                }
        int result;
        if (count == 1)
        {
                result = id[count - 1];
                if (printcite)
                {
                        cout << "this paper has cited: " << endl;
                        for (int i = 0; i < numVertices; i++)   id[i] = -1;
                        count = 0;
                        for (int i = 0; i < result; i++)
                            if (arcs[(1 + result) * result / 2 + i] > 0)
                                id[count++] = i;
                        for (int i = 0, j = 0; i < numVertices && id[j] != -1; i++)
                            if (i == id[j])
                            {
                                cout << "papername: " << Vertices[i].paper << '\n';
                                cout << "author: " << Vertices[i].author << '\n';
                                cout << '#' << '\n';
                                j++;
                            }
                        cout << endl;
                        cout << "this paper has been cited by: " << endl;
                        for (int i = 0; i < numVertices; i++)   id[i] = -1;
                        count = 0;
                        for (int i = result + 1; i < numVertices; i++)
                            if (arcs[(1 + i) * i / 2 + result] > 0)
                                id[count++] = i;
                        for (int i = 0, j = 0; i < numVertices && id[j] != -1; i++)
                            if (i == id[j])
                            {
                                cout << "papername: " << Vertices[i].paper << '\n';
                                cout << "author: " << Vertices[i].author << '\n';
                                cout << '#' << '\n';
                                j++;
                            }
                        cout << endl;
                }
        }
        else if (count == 0) result = -1;
        else result = maxsize + count;

        delete[] id;
        return result;
```

```
}

void CNClass::DFS(int v)
{
        using namespace std;
        cout << "Print all papers contributed to the paper you have inputed:
             " << endl;
        bool visited[maxsize];
        for (int i = 0; i < maxsize; i++)
                visited[i] = false;
        int s[maxsize]; int top = -1;
        cout << "itself: " << endl;
        cout << "papername: " << Vertices[v].paper << '\n';
        cout << "author: " << Vertices[v].author << '\n';
        cout << '#' << '\n';
        cout << "others: " << endl;
        visited[v] = true; s[++top] = v;
        while (top != -1)
        {
                int j = 0;
                v = s[top];
                for (; j < v; j++)
                        if (arcs[(1 + v) * v / 2 + j] > 0 && !visited[j])
                        {
                                cout << "papername: " << Vertices[j].paper
                                     << '\n';
                                cout << "author: " << Vertices[j].author <<
                                     '\n';
                                cout << '#' << '\n';
                                visited[j] = true;
                                s[++top] = j;
                                break;
                        }
                if (j == v)
                        top--;
        }
        cout << endl;
}

```

```
void CNClass::PrintAll()
{
        using namespace std;
        cout << "All informations about papers: " << endl;
        for (int i = 0; i < numVertices; i++)
        {
                cout << "papername: " << Vertices[i].paper << '\n';
                cout << "author: " << Vertices[i].author << '\n';
                cout << "weight: " << Vertices[i].weight << '\n';
        }
}
```

B 附录 2：数据

Listing 10: **VertexData1.txt**

```
10
Market_Maker
Nicholas
1
#
Market_Maker
Tung_Chan
1
#
Market_Making
Yagna
1
#
Multi-Agent
Patel
1
#
Reinforcement_Learning
Knuth
1
#
High-frequency
```

```
Nicholas
1
#
Market_making
Lim
1
#
Reinforcement_Learning
Dijstra
1
#
Machine_Learning
Lim
1
#
High-frequency
Tom
1
#
```

Listing 11: **ArcData1.txt**

```
10
0
0 1 0 0 0 1 1 0 0 0
0 0 1 0 0 0 1 0 0 0
0 0 0 1 0 0 0 1 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 1 0 0 1 1
0 0 0 0 0 0 1 0 1 0
0 0 1 1 0 0 0 1 0 1
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 1 0 0 1
0 0 0 0 0 0 0 0 0 0
```

Listing 12: **VertexData2.txt**

```
10
Multi-Agent
Patel
```

```
3.30208
#
Reinforcement_Learning
Dijstra
3.30208
#
Market_Making
Yagna
2.64583
#
High-frequency
Tom
3.39583
#
Market_making
Lim
3.91667
#
Market_Maker
Tung_Chan
1.33333
#
Machine_Learning
Lim
2.16667
#
High-frequency
Nicholas
1.66667
#
Market_Maker
Nicholas
1
#
Reinforcement_Learning
Knuth
1
#
```

Listing 13: **ArcData2.txt**

```
10
1
0
0 0
1.32292 1.32292 0
0 0 0 0
0.979167 0.979167 0.979167 0.979167 0
0 0 0.666667 0 0.666667 0
0 0 0 1.08333 1.08333 0 0
0 0 0 0 0.833333 0 0.833333 0
0 0 0 0 0.333333 0.333333 0 0.333333 0
0 0 0 0.333333 0 0 0.333333 0.333333 0 0
```

Listing 14: **VertexData3.txt**

```
12
Multi-Agent
Patel
3.85938
#
Reinforcement_Learning
Dijstra
3.85938
#
Market_Making
Yagna
3.23958
#
High-frequency
Tom
4.28125
#
Market_making
Lim
4.95833
#
Market_Maker
Tung_Chan
```

```
2
#
Machine_Learning
Lim
2.75
#
High-frequency
Nicholas
2.16667
#
Market_Maker
Nicholas
1.5
#
Reinforcement_Learning
Knuth
2
#
New1
EnHu
1
#
New2
EnHu
1
#
```

Listing 15: **ArcData3.txt**

```
12
1
0
0 0
1.61979 1.61979 0
0 0 0 0
1.23958 1.23958 1.23958 1.23958 0
0 0 1 0 1 0
0 0 0 1.375 1.375 0 0
0 0 0 0 1.08333 0 1.08333 0
```

```
0 0 0 0 0.5 0.5 0 0.5 0
0 0 0 0.666667 0 0 0.666667 0.666667 0 0
0 0 0 0 0 0.5 0 0 0.5 0 0
0 0 0 0 0 0 0 0 0 1 0 0
```

Listing 16: **VertexData4.txt**

```
10
Multi-Agent
Patel
3.30208
#
Reinforcement_Learning
Dijstra
3.30208
#
Market_Making
Yagna
2.64583
#
High-frequency
Tom
3.39583
#
Market_making
Lim
3.91667
#
Market_Maker
Tung_Chan
1.33333
#
Machine_Learning
Lim
2.16667
#
High-frequency
Nicholas
1.66667
```

```
#
Market_Destroyer
EnHu
1
#
Reinforcement_Learning
Knuth
1
#
```

Listing 17: **ArcData4.txt**

```
10
1
0
0 0
1.32292 1.32292 0
0 0 0 0
0.979167 0.979167 0.979167 0.979167 0
0 0 0.666667 0 0.666667 0
0 0 0 1.08333 1.08333 0 0
0 0 0 0 0.833333 0 0.833333 0
0 0 0 0 0.333333 0.333333 0 0.333333 0
0 0 0 0.333333 0 0 0.333333 0.333333 0 0
```