

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 $\text{weight}/\text{outdegree}$ 。

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:

```
1 class CNClass
2 {
3 public:
4     CNClass(char* VerticesFilename, char* ArcsFilename);
5     ~CNClass();
6     void NewPaper();
7     void Withdraw(int index);
8     void Modify(int index);
9     int SearchPaper(std::string papersearch);
10    int SearchAuthor(std::string authorsearch);
11    int SearchPaperAuthor(std::string papersearch,
12        std::string authorsearch);
13    void DFS(int index);
14    void PrintAll ();
15 private:
16    void InputData(char* VerticesFilename, char* ArcsFilename);
```

```

17     int numVertices;
18     double arcs[(1 + maxsize) / 2 * maxsize]; //compressed type
19     VertexNode Vertices[maxsize];
20 };

```

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：

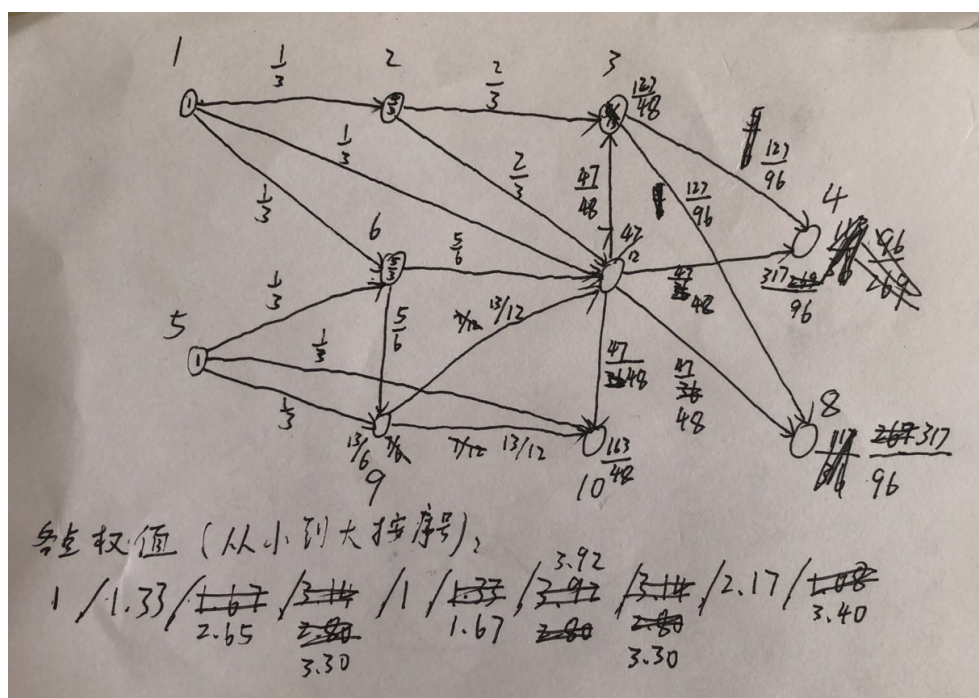


图 1: 第一个图 DAG1

Listing 1: VertexData1.txt

1	10
2	Market_Maker
3	Nicholas
4	1

5	#
6	Market_Maker
7	Tung_Chan
8	1
9	#
10	Market_Making
11	Yagna
12	1
13	#
14	Multi-Agent
15	Patel
16	1
17	#
18	Reinforcement_Learning
19	Knuth
20	1
21	#
22	High-frequency
23	Nicholas
24	1
25	#
26	Market_making
27	Lim
28	1
29	#
30	Reinforcement_Learning
31	Dijkstra
32	1
33	#
34	Machine_Learning
35	Lim
36	1
37	#

```
38 High-frequency
39 Tom
40 1
41 #
```

Listing 2: ArcData1.txt

```
1 10
2 0
3 0 1 0 0 0 1 1 0 0 0
4 0 0 1 0 0 0 1 0 0 0
5 0 0 0 1 0 0 0 1 0 0
6 0 0 0 0 0 0 0 0 0 0
7 0 0 0 0 0 1 0 0 1 1
8 0 0 0 0 0 0 1 0 1 0
9 0 0 1 1 0 0 0 1 0 1
10 0 0 0 0 0 0 0 0 0 0
11 0 0 0 0 0 0 1 0 0 1
12 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

```

1 10
2 Multi-Agent
3 Patel
4 3.30208
5 #
6 Reinforcement_Learning
7 Dijkstra
8 3.30208
9 #
10 Market_Making
11 Yagna
12 2.64583
13 #
14 High-frequency
15 Tom
16 3.39583

```



```

17 #
18 Market_making
19 Lim
20 3.91667
21 #
22 Market_Maker
23 Tung_Chan
24 1.33333
25 #
26 Machine_Learning
27 Lim
28 2.16667
29 #
30 High-frequency
31 Nicholas
32 1.66667
33 #
34 Market_Maker
35 Nicholas
36 1
37 #
38 Reinforcement_Learning
39 Knuth
40 1
41 #

```

Listing 4: ArcData2.txt

```

1 10
2 1
3 0
4 0 0
5 1.32292 1.32292 0
6 0 0 0 0

```

```

7 0.979167 0.979167 0.979167 0.979167 0
8 0 0 0.666667 0 0.666667 0
9 0 0 0 1.08333 1.08333 0 0
10 0 0 0 0 0.833333 0 0.833333 0
11 0 0 0 0 0.333333 0.333333 0 0.333333 0
12 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

```

1 10
2 Multi-Agent
3 Patel
4 3.30208
5 #
6 Reinforcement_Learning

```

7	Dijkstra
8	3.30208
9	#
10	Market_Making
11	Yagna
12	2.64583
13	#
14	High-frequency
15	Tom
16	3.39583
17	#
18	Market_making
19	Lim
20	3.91667
21	#
22	Market_Maker
23	Tung_Chan
24	1.33333
25	#
26	Machine_Learning
27	Lim
28	2.16667
29	#
30	High-frequency
31	Nicholas
32	1.66667
33	#
34	Market_Maker
35	Nicholas
36	1
37	#
38	Reinforcement_Learning
39	Knuth

```

40 | 1
41 | #

```

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：

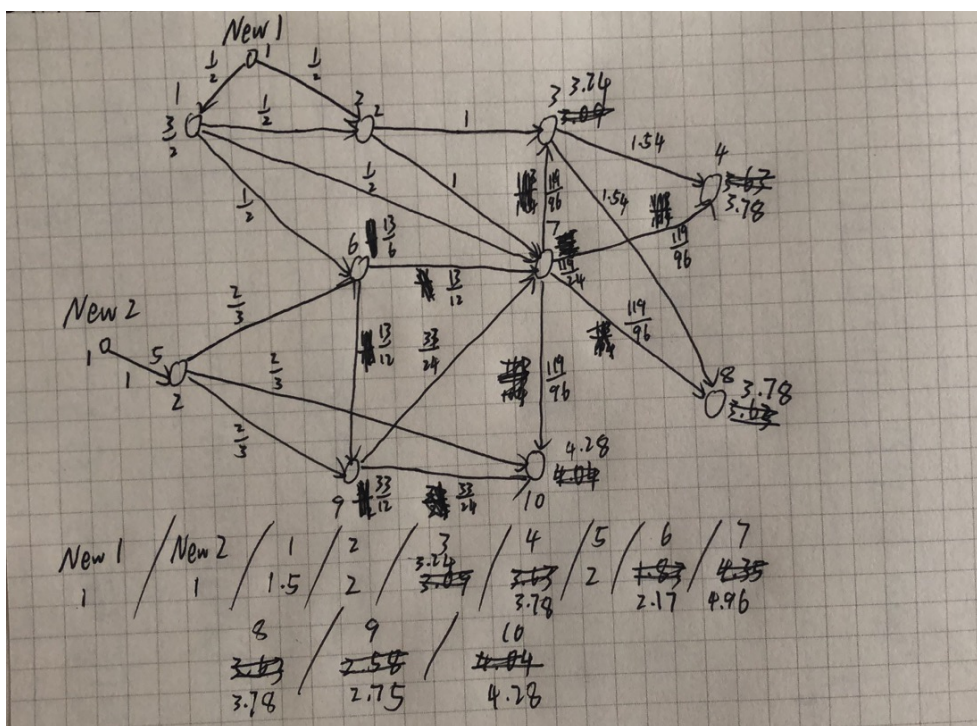


图 4: 在第一个图 DAG1 中增加顶点 New1 和 New2

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 papename and authername 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:
papename: 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:
papename: 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

```
1 12
2 Multi-Agent
3 Patel
4 3.85938
5 #
6 Reinforcement_Learning
7 Dijkstra
8 3.85938
9 #
10 Market_Making
11 Yagna
12 3.23958
13 #
14 High-frequency
15 Tom
16 4.28125
17 #
18 Market_making
19 Lim
20 4.95833
21 #
22 Market_Maker
23 Tung_Chan
24 2
25 #
26 Machine_Learning
27 Lim
28 2.75
29 #
30 High-frequency
31 Nicholas
32 2.16667
```

```
33 #
34 Market_Maker
35 Nicholas
36 1.5
37 #
38 Reinforcement_Learning
39 Knuth
40 2
41 #
42 New1
43 EnHu
44 1
45 #
46 New2
47 EnHu
48 1
49 #
```

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\Ht\Desktop\数据结构\数据结构头验大作业\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: Dijkstra
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\Citatic
your decision: 8
=====
Start PrintAll!
All informations about papers:
papername: Multi-Agent
author: Patel
weight: 3.30208
papername: Reinforcement_Learning
author: Dijkstra
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：

```

=====
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: Dijkstra
#
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

```
1 //Copyright (c) 2020 En Hu. All rights reserved.
2
3 #pragma once
4 #include<iostream>
5 #include<string>
6 #include<fstream>
7
8 #ifndef MAXSIZE
9 #define MAXSIZE
10 const int maxsize = 20;
11 #endif // !MAXSIZE
12
13 struct VertexNode
14 {
15     std::string paper;
16     std::string author;
17     double weight;
18     int indegree, outdegree;
19 };
20
21 class CNClass
22 {
23 public:
24     CNClass(char* VerticesFilename, char* ArcsFilename);
25     ~CNClass();
26     void NewPaper();
27     void Withdraw(int index);
28     void Modify(int index);
29     int SearchPaper(std::string papersearch);
30     int SearchAuthor(std::string authorsearch);
31     int SearchPaperAuthor(std::string papersearch, std::string
        authorsearch, bool printcite);
32     void DFS(int index);
33     void PrintAll();
34 private:
```



```

35     void InputData(char* VerticesFilename, char* ArcsFilename);
36     int numVertices;
37     double arcs[(1 + maxsize) / 2 * maxsize]; //compressed type
38     VertexNode Vertices[maxsize];
39 };

```

Listing 8: CitationNetwork.cpp

```

1  //Copyright (c) 2020 En Hu. All rights reserved.
2
3  #include <iostream>
4  #include "CNClass.h"
5
6  int main()
7  {
8      using namespace std;
9      cout << "Hello World!\n";
10     cout << "This program analyze citation network.\n";
11     char split[] = "===== "
12     ;
13     cout << split << endl;
14     //char VerticesFilename[] = "VertexData1.txt", ArcsFilename[] = "
15     ArcData1.txt";
16     char VerticesFilename[50], ArcsFilename[50];
17     cout << "Enter the VerticesFilename(suggestion: VertexData1.txt)" <<
18     endl << "name: ";
19     cin >> VerticesFilename;
20     cout << "Enter the ArcsFilename(suggestion: ArcData1.txt)" << endl
21     << "name: ";
22     cin >> ArcsFilename;
23     CNClass CN1(VerticesFilename, ArcsFilename);
24     cout << split << endl;
25     bool finished = false;
26     while (!finished)
27     {
28         int id_operation;
29         cout << "What do you want to do? Enter the index of
30         operation" << endl;
31         cout << "1. Add new paper" << endl

```

```

27         << "2. Withdraw a paper" << endl
28         << "3. Modify a paper" << endl
29         << "4. Search a paper with author(detailed)" << endl
30         << "5. Search papers without author" << endl
31         << "6. Search author" << endl
32         << "7. DFS a paper with author(means find all papers
           contribute to it)" << endl
33         << "8. Print all information" << endl
34         << "9. Quit" << endl;
35     cout << "your decision: ";
36     cin >> id_operation;
37     string papername, authorname; int index;
38     switch (id_operation)
39     {
40     case 1:
41         cout << split << endl;
42         cout << "Begin adding new paper! " << endl;
43         CN1.NewPaper();
44         cout << split << endl;
45         break;
46     case 2:
47         char decision;
48         cout << "Do you really want to withdraw the paper?
           Input 'y' if yes, else if no." << endl;
49         cin >> decision;
50         if (decision != 'y') break;
51         else
52         {
53             cout << split << endl;
54             cout << "Begin withdrawing! " << endl;
55             cout << "Please input the name of paper you
           want to withdraw. " << endl
56                 << "name: ";
57             cin >> papername;
58             cout << "Please input the name of author you
           want to withdraw. " << endl
59                 << "name: ";
60             cin >> authorname;

```

```

61         index = CN1.SearchPaperAuthor(papername,
62             authorname, false);
63         if (index == -1)      cout << "I cannot
64             find the paper. " << endl;
65         else if (index > maxsize)      cout << "
66             There is more than one paper. I have
67             not solve this problem now. " << endl;
68         else CN1.Withdraw(index);
69         cout << split << endl;
70         break;
71     }
72 case 3:
73     cout << split << endl;
74     cout << "Please input the name of paper you want to
75         modify. " << endl
76         << "name: ";
77     cin >> papername;
78     cout << "Please input the name of author you want to
79         modify. " << endl
80         << "name: ";
81     cin >> authorname;
82     index = CN1.SearchPaperAuthor(papername, authorname,
83         false);
84     if (index == -1)      cout << "I cannot find the
85         paper. " << endl;
86     else if (index > maxsize)      cout << "There is
87         more than one paper. I have not solve this
88         problem now. " << endl;
89     else CN1.Modify(index);
90     cout << split << endl;
91     break;
92 case 4:
93     cout << split << endl;
94     cout << "Start SearchPaperAuthor! " << endl;
95     cout << "Please input the name of paper. " << endl
96         << "name: ";
97     cin >> papername;
98     cout << "Please input the name of author you want. "
99         << endl

```

```

89         << "name: ";
90         cin >> authorname;
91         CN1.SearchPaperAuthor(papername, authorname, true);
92         cout << split << endl;
93         break;
94     case 5:
95         cout << split << endl;
96         cout << "Start SearchPaper! " << endl;
97         cout << "Please input the name of paper you want. "
98             << endl
99             << "name: ";
100        cin >> papername;
101        CN1.SearchPaper(papername);
102        cout << split << endl;
103        break;
104    case 6:
105        cout << split << endl;
106        cout << "Start SearchAuthor! " << endl;
107        cout << "Please input the name of author you want. "
108            << endl
109            << "name: ";
110        cin >> authorname;
111        CN1.SearchAuthor(authorname);
112        cout << split << endl;
113        break;
114    case 7:
115        cout << split << endl;
116        cout << "Start DFS! " << endl;
117        cout << "Please input the name of paper you want to
118            DFS. " << endl
119            << "name: ";
120        cin >> papername;
121        cout << "Please input the name of author you want to
122            DFS. " << endl
123            << "name: ";
124        cin >> authorname;
125        index = CN1.SearchPaperAuthor(papername, authorname,
126            false);

```

```

122         if (index == -1)          cout << "I cannot find the
                                     paper. " << endl;
123         else if (index > maxsize)  cout << "There is
                                     more than one paper. I have not solve this
                                     problem now. " << endl;
124         else CN1.DFS(index);
125         cout << split << endl;
126         break;
127     case 8:
128         cout << split << endl;
129         cout << "Start PrintAll! " << endl;
130         CN1.PrintAll();
131         cout << split << endl;
132         break;
133     case 9:
134         finished = true;
135         break;
136     default:
137         cout << "Wrong index! Please enter again. " << endl;
138         break;
139     }
140 }
141 }
142
143 // 运行程序: Ctrl + F5 或调试“开始执行不调试 >()”菜单
144 // 调试程序: F5 或调试“开始调试”菜单 >

```

Listing 9: CNClass.cpp

```

1  //Copyright (c) 2020 En Hu. All rights reserved.
2  #include "CNClass.h"
3
4  CNClass::CNClass(char* VerticesFilename, char* ArcsFilename)
5  {
6      using namespace std;
7      //load files
8      ifstream VexFile, ArcFile;
9      cout << "I have loaded: " << endl;
10     cout << VerticesFilename << ' ' << ArcsFilename << endl;

```

```

11 VexFile.open(VerticesFilename);
12 ArcFile.open(ArcsFilename);
13 //get number of vertices and the sign of is-sorted-and-weighted
14 int t1, t2;//temporary int-type data
15 VexFile >> t1; ArcFile >> t2;
16 if (t1 != t2) throw("number of Vertices is not equal");
17 else if (t1 < 0) throw("number of Vertices cannot less than 0
    ");
18 else if (t1 > maxsize) throw("overflow");
19 else numVertices = t1;
20 bool isSortedWeighted;
21 ArcFile >> t1;
22 switch (t1)
23 {
24 case 0:
25     isSortedWeighted = false;
26     break;
27 case 1:
28     isSortedWeighted = true;
29     break;
30 default:
31     throw("error type of sign of is-triangular-matrix");
32     break;
33 }
34 if (isSortedWeighted)//if the files has been sorted-and-weighted
35 {
36     //get data of paper, author, and weights
37     string paper, author, finished; double weights;
38     for (int i = 0; i < numVertices; i++)
39     {
40         VexFile >> paper;
41         VexFile >> author;
42         VexFile >> weights;
43         Vertices[i].paper = paper;
44         Vertices[i].author = author;
45         Vertices[i].weight = weights;
46         VexFile >> finished;
47         if (finished == "#") continue;
48         else throw("error: the type of VexFile is wrong");

```

```

49     }
50     //get data of arcs
51     for (int i = 0; i < (1 + numVertices) * numVertices / 2; i
        ++ ) ArcFile >> arcs[i];
52     //initialize empty room
53     for (int i = (1 + numVertices) * numVertices / 2; i <
        maxsize; i++) arcs[i] = 0;
54 }
55 else//if not, then start from scratch
56 {
57     //initialize arcs matrix
58     for (int i = 0; i < (1 + maxsize) / 2 * maxsize; i++)
59         arcs[i] = 0;
60     VertexNode* tempVex = new VertexNode[numVertices];//
        temporary vertexdata
61     //temporaray arcdata
62     bool** tempArcs = new bool* [numVertices];
63     for (int i = 0; i < numVertices; i++)
64         tempArcs[i] = new bool[numVertices];
65     //get data of vertex
66     string paper, author, finished; double weights;
67     for (int i = 0; i < numVertices; i++)
68     {
69         VexFile >> paper;
70         VexFile >> author;
71         VexFile >> weights;
72         tempVex[i].paper = paper;
73         tempVex[i].author = author;
74         tempVex[i].weight = 1;
75         VexFile >> finished;
76         if (finished == "#") continue;
77         else throw("error: the type of VexFile is wrong");
78     }
79     //get data of arcs
80     for (int i = 0; i < numVertices; i++)
81         for (int j = 0; j < numVertices; j++)
82         {
83             ArcFile >> t1;
84             if (t1 > 0)

```

```

85         tempArcs[i][j] = true;
86         else tempArcs[i][j] = false;
87     }
88     //get indegree
89     for (int j = 0; j < numVertices; j++)
90     {
91         int count = 0;
92         for (int i = 0; i < numVertices; i++)
93             if (tempArcs[i][j]) count++;
94         tempVex[j].indegree = count;
95     }
96     //toposort
97     int* s = new int[numVertices]; //stack, which stores index of
98     vertices
99     int* toposeries = new int[numVertices]; //stores toposeries
100     int top = -1, count = 0;
101     for (int i = 0; i < numVertices; i++)
102         if (tempVex[i].indegree == 0)
103             s[++top] = i;
104     while (top != -1)
105     {
106         t1 = s[top--];
107
108         //cout << tempVex[t1];
109         toposeries[count++] = t1;
110
111         for (int j = 0; j < numVertices; j++)
112             if (tempArcs[t1][j])
113             {
114                 tempVex[j].indegree--;
115                 if (tempVex[j].indegree == 0)
116                     s[++top] = j;
117             }
118     }
119     if (count < numVertices) throw("there is a ring");
120
121     //int* inversetopo = new int[numVertices]; //stores inverse
122     of toposeries
123     //for (int i = 0; i < numVertices; i++)

```



```

122         //      inversetopo[toposeries[i]] = i;
123
124         //input paper and author
125         for (int i = 0; i < numVertices; i++)
126         {
127             Vertices[numVertices - 1 - i].paper = tempVex[
128                 toposeries[i]].paper;
129             Vertices[numVertices - 1 - i].author = tempVex[
130                 toposeries[i]].author;
131         }
132
133         //weigh vertices and arcs
134         for (int i = numVertices - 1; i >= 0; i--)
135         {
136             //sum of weights of inarcs
137             double sum = 0;
138             for (int j = i + 1; j < numVertices; j++)
139                 sum = sum + arcs[(1 + j) * j / 2 + i];
140             //weigh vertex
141             Vertices[i].weight = sum + 1;
142             //calculate outdegree
143             Vertices[i].outdegree = 0;
144             for (int j = 0; j < numVertices; j++)
145                 if (tempArcs[toposeries[numVertices - i -
146                     1]][j])
147                     Vertices[i].outdegree++;
148             //set the weights of outarcs
149             for (int j = 0; j < i; j++)
150                 if (tempArcs[toposeries[numVertices - i -
151                     1]][toposeries[numVertices - j - 1]])
152                     arcs[(1 + i) * i / 2 + j] = Vertices
153                         [i].weight / Vertices[i].
154                         outdegree;
155         }
156
157         VexFile.close(); ArcFile.close();
158         //delete[] inversetopo;
159         delete[] toposeries;
160         delete[] tempVex;

```

```

155         for (int i = 0; i < numVertices; i++)
156             delete[] tempArcs[i];
157         delete[] tempArcs;
158     }
159 }
160
161 CNClass::~CNClass()
162 {
163     using namespace std;
164     bool isdecided = false;
165     while (!isdecided)
166     {
167         cout << "Do you want to save the data? Input 'y' if yes, 'n'
168             if no. " << endl
169             << "your decision: ";
170         char decision;
171         cin >> decision;
172         cout << endl;
173         if (decision == 'y')
174         {
175             char VerticesFilename[50], ArcsFilename[50];
176             char split[] = "
=====
";
177             cout << split << endl << "I am ready to input data.
" << endl;
178             cout << "Please input the VerticesFilename(
suggestion: VertexData2.txt)" << endl << "name:
";
179             cin >> VerticesFilename;
180             cout << "Please input the ArcsFilename(suggestion:
ArcData2.txt)" << endl << "name: ";
181             cin >> ArcsFilename;
182             InputData(VerticesFilename, ArcsFilename);
183             isdecided = true;
184             cout << "This program is about to finish. Goodbye! "
<< endl;
185         }
186     }
187     else if (decision == 'n')

```

```

186         {
187             isdecided = true;
188             cout << "This program is about to finish. Goodbye! "
                  << endl;
189         }
190         else
191             cout << "Error decision! Please decided again. " <<
                  endl;
192     }
193     system("pause");
194 }
195
196 void CNClass::NewPaper()
197 {
198     using namespace std;
199     if (numVertices == maxsize)
200     {
201         cout << "overflow";
202         return;
203     }
204     //add vertex
205     string papername, authorname;
206     cout << "Please input the name of paper you want to add. " << endl
          << "name: ";
207     cin >> papername;
208     cout << "Please input the name of author you want to add. " << endl
          << "name: ";
209     cin >> authorname;
210     Vertices[numVertices].paper = papername;
211     Vertices[numVertices].author = authorname;
212     Vertices[numVertices].weight = 1;
213     cout << "new paper's papername and authorname have been added! " <<
          endl << endl;
214
215     //add arcs
216     int count = 0;
217     int citedid[maxsize];
218     for (int i = 0; i < maxsize; i++)        citedid[i] = -1;
219     bool finishedsign = false;
220     while (!finishedsign)
221

```

```

222     {
223         char finished;
224         cout << "Have you input all papers have been cited? Input 'y
                ' if yes, 'n' if no" << endl;
225         cin >> finished;
226         if (finished == 'y')    finishedsign = true;
227         else if (finished == 'n')
228         {
229             cout << "Please input the name of " << (count + 1)
                << "th paper has been cited. " << endl
                << "name: ";
230             cin >> papername;
231             cout << "Please input the name of " << (count + 1)
                << "th author has been cited. " << endl
                << "name: ";
232             cin >> authorname;
233             int i = SearchPaperAuthor(papername, authorname,
                false);
234             if (i >= 0 && i < maxsize)
235                 citedid[count++] = i;
236             else if (i == -1)
237                 cout << "I cannot find the paper you have
                inputed. " << endl;
238             else
239                 cout << "More than one paper. I have not
                solve this problem. " << endl;
240         }
241     }
242     else cout << "Error type! Please input again. " << endl;
243 }
244 Vertices[numVertices].outdegree = count;
245 numVertices++;
246 for (int i = 0; i < numVertices - 1 && citedid[i] != -1; i++)
247     //arcs[numVertices - 1][citedid[i]] = 1 / count;
248     arcs[numVertices * (numVertices - 1) / 2 + citedid[i]] = 1.0
249     / double(count);
250
251 //update weights of vertices and arcs
252 for (int i = numVertices - 2; i >= 0; i--)
253 {

```

```

254         //sum of weights of inarcs
255         double sum = 0;
256         for (int j = i + 1; j < numVertices; j++)
257             sum = sum + arcs[(1 + j) * j / 2 + i];
258         //weigh vertex
259         Vertices[i].weight = sum + 1;
260         //calculate outdegree
261         Vertices[i].outdegree = 0;
262         for (int j = 0; j < i; j++)
263             if (arcs[(1 + i) * i / 2 + j] > 0) Vertices[i].
                outdegree++;
264         //set the weights of outarcs
265         double outarcweight = Vertices[i].weight / Vertices[i].
                outdegree;
266         for (int j = 0; j < i; j++)
267             if (arcs[(1 + i) * i / 2 + j] > 0) arcs[(1 + i)
                * i / 2 + j] = outarcweight;
268     }
269     cout << "New paper and all its citaion have been added! " << endl;
270 }
271
272 void CNClass::InputData(char* VerticesFilename, char* ArcsFilename)
273 {
274     using namespace std;
275     fstream Vexfile, Arcfile;
276     char split[] = "=====
                ";
277     cout << split << endl;
278     cout << "Begin inputing data! " << endl;
279     Vexfile.open(VerticesFilename); Arcfile.open(ArcsFilename);
280     //input arcs data
281     Arcfile << numVertices << '\n';
282     Arcfile << 1 << '\n';
283     for (int i = 0; i < numVertices; i++)
284     {
285         for (int j = 0; j <= i; j++)
286             Arcfile << arcs[(1 + i) * i / 2 + j] << ' ';
287         Arcfile << '\n';
288     }

```

```

289
290     //input vex data
291     Vexfile << numVertices << '\n';
292     for (int i = 0; i < numVertices; i++)
293     {
294         Vexfile << Vertices[i].paper << '\n';
295         Vexfile << Vertices[i].author << '\n';
296         Vexfile << Vertices[i].weight << '\n';
297         Vexfile << '#' << '\n';
298     }
299     cout << "Data has been input! " << endl;
300     cout << split << endl;
301 }
302
303 void CNClass::Withdraw(int index)
304 {
305     using namespace std;
306     //start withdrawing
307     cout << "Sorry! I have not design this algorithm. " << endl;
308 }
309
310 void CNClass::Modify(int index)
311 {
312     using namespace std;
313     cout << "Begin modifying! " << endl;
314     string papername, authername;
315     cout << "Please input new paper name. " << endl
316         << "name: ";
317     cin >> papername;
318     cout << "Please input new author name. " << endl
319         << "name: ";
320     cin >> authername;
321     Vertices[index].paper = papername;
322     Vertices[index].author = authername;
323     cout << "Modify successfully! " << endl;
324 }
325
326 int CNClass::SearchPaper(std::string papersearch)
327 {

```

```

328     using namespace std;
329     int* id = new int[numVertices];
330     for (int i = 0; i < numVertices; i++)    id[i] = -1;
331     int count = 0;
332     for (int i = 0; i < numVertices; i++)
333         if (Vertices[i].paper == papersearch)
334             id[count++] = i;
335     cout << "There are " << count << " papers matched: " << endl;
336     for (int i = 0, j = 0; i < numVertices && id[j] != -1; i++)
337         if (i == id[j])
338         {
339             cout << "papername: " << Vertices[i].paper << '\n';
340             cout << "author: " << Vertices[i].author << '\n';
341             cout << "weight: " << Vertices[i].weight << '\n';
342             cout << '#' << '\n';
343             j++;
344         }
345
346     delete[] id;
347     return count;
348 }
349
350 int CNClass::SearchAuthor(std::string authorsearch)
351 {
352     using namespace std;
353     int* id = new int[numVertices];
354     for (int i = 0; i < numVertices; i++)    id[i] = -1;
355     int count = 0;
356     for (int i = 0; i < numVertices; i++)
357         if (Vertices[i].author == authorsearch)
358             id[count++] = i;
359     cout << "There are " << count << " papers matched: " << endl;
360     for (int i = 0, j = 0; i < numVertices && id[j] != -1; i++)
361         if (i == id[j])
362         {
363             cout << "papername: " << Vertices[i].paper << '\n';
364             cout << "author: " << Vertices[i].author << '\n';
365             cout << "weight: " << Vertices[i].weight << '\n';
366             cout << '#' << '\n';

```

```

367         j++;
368     }
369
370     delete[] id;
371     return count;
372
373 }
374
375 int CNClass::SearchPaperAuthor(std::string papersearch, std::string
authorsearch, bool printcite)
376 {
377     using namespace std;
378     int* id = new int[numVertices];
379     for (int i = 0; i < numVertices; i++)    id[i] = -1;
380     int count = 0;
381     for (int i = 0; i < numVertices; i++)
382         if (Vertices[i].paper == papersearch && Vertices[i].author
== authorsearch)
383             id[count++] = i;
384     cout << "There are " << count << " papers matched: " << endl;
385     for (int i = 0, j = 0; i < numVertices && id[j] != -1; i++)
386         if (i == id[j])
387         {
388             cout << "papername: " << Vertices[i].paper << '\n';
389             cout << "author: " << Vertices[i].author << '\n';
390             cout << "weight: " << Vertices[i].weight << '\n';
391             cout << '#' << '\n';
392             j++;
393         }
394     int result;
395     if (count == 1)
396     {
397         result = id[count - 1];
398         if (printcite)
399         {
400             cout << "this paper has cited: " << endl;
401             for (int i = 0; i < numVertices; i++)    id[i] = -1;
402             count = 0;
403             for (int i = 0; i < result; i++)

```



```

404         if (arcs[(1 + result) * result / 2 + i] > 0)
405             id[count++] = i;
406     for (int i = 0, j = 0; i < numVertices && id[j] !=
407         -1; i++)
408         if (i == id[j])
409         {
410             cout << "papername: " << Vertices[i]
411                 .paper << '\n';
412             cout << "author: " << Vertices[i].
413                 author << '\n';
414             cout << '#' << '\n';
415             j++;
416         }
417     cout << endl;
418     cout << "this paper has been cited by: " << endl;
419     for (int i = 0; i < numVertices; i++)    id[i] = -1;
420     count = 0;
421     for (int i = result + 1; i < numVertices; i++)
422         if (arcs[(1 + i) * i / 2 + result] > 0)
423             id[count++] = i;
424     for (int i = 0, j = 0; i < numVertices && id[j] !=
425         -1; i++)
426         if (i == id[j])
427         {
428             cout << "papername: " << Vertices[i]
429                 .paper << '\n';
430             cout << "author: " << Vertices[i].
431                 author << '\n';
432             cout << '#' << '\n';
433             j++;
434         }
435     cout << endl;
436     }
437 }
438 else if (count == 0) result = -1;
439 else result = maxsize + count;
440
441 delete[] id;
442 return result;

```

```

437 }
438
439 void CNClass::DFS(int v)
440 {
441     using namespace std;
442     cout << "Print all papers contributed to the paper you have inputed:
         " << endl;
443     bool visited[maxsize];
444     for (int i = 0; i < maxsize; i++)
445         visited[i] = false;
446     int s[maxsize]; int top = -1;
447     cout << "itself: " << endl;
448     cout << "papername: " << Vertices[v].paper << '\n';
449     cout << "author: " << Vertices[v].author << '\n';
450     cout << '#' << '\n';
451     cout << "others: " << endl;
452     visited[v] = true; s[++top] = v;
453     while (top != -1)
454     {
455         int j = 0;
456         v = s[top];
457         for (; j < v; j++)
458             if (arcs[(1 + v) * v / 2 + j] > 0 && !visited[j])
459             {
460                 cout << "papername: " << Vertices[j].paper
461                     << '\n';
462                 cout << "author: " << Vertices[j].author <<
463                     '\n';
464                 cout << '#' << '\n';
465                 visited[j] = true;
466                 s[++top] = j;
467                 break;
468             }
469         if (j == v)
470             top--;
471     }
472     cout << endl;

```

```

473 void CNClass::PrintAll()
474 {
475     using namespace std;
476     cout << "All informations about papers: " << endl;
477     for (int i = 0; i < numVertices; i++)
478     {
479         cout << "papername: " << Vertices[i].paper << '\n';
480         cout << "author: " << Vertices[i].author << '\n';
481         cout << "weight: " << Vertices[i].weight << '\n';
482     }
483 }

```

B 附录 2：数据

Listing 10: VertexData1.txt

```

1 10
2 Market_Maker
3 Nicholas
4 1
5 #
6 Market_Maker
7 Tung_Chan
8 1
9 #
10 Market_Making
11 Yagna
12 1
13 #
14 Multi-Agent
15 Patel
16 1
17 #
18 Reinforcement_Learning
19 Knuth
20 1
21 #
22 High-frequency

```

```

23 Nicholas
24 1
25 #
26 Market_making
27 Lim
28 1
29 #
30 Reinforcement_Learning
31 Dijkstra
32 1
33 #
34 Machine_Learning
35 Lim
36 1
37 #
38 High-frequency
39 Tom
40 1
41 #

```

Listing 11: **ArcData1.txt**

```

1 10
2 0
3 0 1 0 0 0 1 1 0 0 0
4 0 0 1 0 0 0 1 0 0 0
5 0 0 0 1 0 0 0 1 0 0
6 0 0 0 0 0 0 0 0 0 0
7 0 0 0 0 0 1 0 0 1 1
8 0 0 0 0 0 0 1 0 1 0
9 0 0 1 1 0 0 0 1 0 1
10 0 0 0 0 0 0 0 0 0 0
11 0 0 0 0 0 0 1 0 0 1
12 0 0 0 0 0 0 0 0 0 0

```

Listing 12: **VertexData2.txt**

```

1 10
2 Multi-Agent
3 Patel

```

4	3.30208
5	#
6	Reinforcement_Learning
7	Dijkstra
8	3.30208
9	#
10	Market_Making
11	Yagna
12	2.64583
13	#
14	High-frequency
15	Tom
16	3.39583
17	#
18	Market_making
19	Lim
20	3.91667
21	#
22	Market_Maker
23	Tung_Chan
24	1.33333
25	#
26	Machine_Learning
27	Lim
28	2.16667
29	#
30	High-frequency
31	Nicholas
32	1.66667
33	#
34	Market_Maker
35	Nicholas
36	1
37	#
38	Reinforcement_Learning
39	Knuth
40	1
41	#

Listing 13: **ArcData2.txt**

```

1 10
2 1
3 0
4 0 0
5 1.32292 1.32292 0
6 0 0 0 0
7 0.979167 0.979167 0.979167 0.979167 0
8 0 0 0.666667 0 0.666667 0
9 0 0 0 1.08333 1.08333 0 0
10 0 0 0 0 0.833333 0 0.833333 0
11 0 0 0 0 0.333333 0.333333 0 0.333333 0
12 0 0 0 0.333333 0 0 0.333333 0.333333 0 0

```

Listing 14: **VertexData3.txt**

```

1 12
2 Multi-Agent
3 Patel
4 3.85938
5 #
6 Reinforcement_Learning
7 Dijkstra
8 3.85938
9 #
10 Market_Making
11 Yagna
12 3.23958
13 #
14 High-frequency
15 Tom
16 4.28125
17 #
18 Market_making
19 Lim
20 4.95833
21 #
22 Market_Maker
23 Tung_Chan

```

```

24 | 2
25 | #
26 | Machine_Learning
27 | Lim
28 | 2.75
29 | #
30 | High-frequency
31 | Nicholas
32 | 2.16667
33 | #
34 | Market_Maker
35 | Nicholas
36 | 1.5
37 | #
38 | Reinforcement_Learning
39 | Knuth
40 | 2
41 | #
42 | New1
43 | EnHu
44 | 1
45 | #
46 | New2
47 | EnHu
48 | 1
49 | #

```

Listing 15: ArcData3.txt

```

1 | 12
2 | 1
3 | 0
4 | 0 0
5 | 1.61979 1.61979 0
6 | 0 0 0 0
7 | 1.23958 1.23958 1.23958 1.23958 0
8 | 0 0 1 0 1 0
9 | 0 0 0 1.375 1.375 0 0
10 | 0 0 0 0 1.08333 0 1.08333 0

```

```

11 0 0 0 0 0.5 0.5 0 0.5 0
12 0 0 0 0.666667 0 0 0.666667 0.666667 0 0
13 0 0 0 0 0 0.5 0 0 0.5 0 0
14 0 0 0 0 0 0 0 0 0 1 0 0

```

Listing 16: **VertexData4.txt**

```

1 10
2 Multi-Agent
3 Patel
4 3.30208
5 #
6 Reinforcement_Learning
7 Dijkstra
8 3.30208
9 #
10 Market_Making
11 Yagna
12 2.64583
13 #
14 High-frequency
15 Tom
16 3.39583
17 #
18 Market_making
19 Lim
20 3.91667
21 #
22 Market_Maker
23 Tung_Chan
24 1.33333
25 #
26 Machine_Learning
27 Lim
28 2.16667
29 #
30 High-frequency
31 Nicholas
32 1.66667

```



```
33 #
34 Market_Destroyer
35 EnHu
36 1
37 #
38 Reinforcement_Learning
39 Knuth
40 1
41 #
```

Listing 17: ArcData4.txt

```
1 10
2 1
3 0
4 0 0
5 1.32292 1.32292 0
6 0 0 0 0
7 0.979167 0.979167 0.979167 0.979167 0
8 0 0 0.666667 0 0.666667 0
9 0 0 0 1.08333 1.08333 0 0
10 0 0 0 0 0.833333 0 0.833333 0
11 0 0 0 0 0.333333 0.333333 0 0.333333 0
12 0 0 0 0.333333 0 0 0.333333 0.333333 0 0
```