Lab07

Description

Small World Phenomenon

In The Small-World Phenomenon: An Algorithmic Perspective, Jon Kleinberg writes:

> Long a matter of folklore, the *small-world phenomenon* -- the principle that we are all linked by short chains of acquaintances -- was inaugurated as an area of experimental study in the social sciences through the pioneering work of Stanley Milgram in the 1960's. This work was among the first to make the phenomenon quantitative, allowing people to speak of the *six degrees of separation* between any two people in the United States. Since then, a number of network models have been proposed as frameworks in which to study the problem analytically. One of the most refined of these models was formulated in recent work of Watts and Strogatz; their framework provided compelling evidence that the small-world phenomenon is pervasive in a range of networks arising in nature and technology, and a fundamental ingredient in the evolution of the World Wide Web.

In this assignment you will investigate the *six degrees of Kevin Bacon*. Two actors or actresses are linked if they appeared in a movie together. The Kevin Bacon number of an actor is the shortest chain of links that leads to Kevin Bacon. For example, Robert De Niro has a Kevin Bacon number of 1 because he appeared in *Sleepers* with Kevin Bacon. Elvis Presley's number is 2: although Elvis did not appear in any movie with Kevin Bacon, he was in *Change of Habit* with Edward Asner, and Asner appeared in *JFK* with Kevin Bacon. Your task is to read in a file containing a list of movies and the actors that appeared in them and compute the Kevin Bacon numbers for each actor. You will then read in a list of actors from standard input and print out a shortest chain of movies that leads each actor back to Kevin Bacon.

Input

In addition to the massive list of movies and casts from the Internet Movie Database, we include some smaller data files that include only a specific subset of movies, e.g., all movies released in 2000. Each line in the data file consists of a movie title, followed by a list of actors and actresses that appeared in that movie, delimited by the character `'/'`. Here is an abbreviated example:

```
Picture Perfect (1997)/Aniston, Jennifer/Bacon, Kevin/Dukakis, Olympia/Mohr, Jay
Planes, Trains & Automobiles (1987)/Bacon, Kevin/Candy, John/Martin, Steve/Robins, Laila
Beach, The (2000)/DiCaprio, Leonardo/York, Daniel/Patarakijjanon, Patcharawan
```

Use a command-line parameter to enter the name of the movie database file. You will also read in a list of actors from standard input, one per line.

Output

First, print out a table of the distribution of Kevin Bacon numbers.

```
Bacon number    Frequency
-------------------------
           0            1
           1         1494
           2       127791
           3       239671
           4        36475
           5         2965
           6          275
           7           39
           8           47
           9           99
          10           15
          11            2
    infinity         9703
```

Then, for each actor and actress read from standard input, print out their Kevin Bacon number and a shortest chain of movies that connect them to Kevin Bacon. Here's an example.

```
Dane, Cynthia has a Bacon number of 3
Dane, Cynthia was in "Solstice (1994)" with Scott, Dennis
Scott, Dennis was in "What About Bob? (1991)" with Murray, Bill
Murray, Bill was in "Wild Things (1998)" with Bacon, Kevin
```

Solution

题目意思大致就是找最短路径，设每个演员为点，演过同一部电影，就在这两个点上连边，

查找各个点到演员**Bacon, Kevin**的最短路的值**Bacon number**，并且输出每个最短路值的**频率**，

即有**n**个演员的最短路值为**k**，则**Bacon number[k] = n**

另外如果没有到**Bacon, Kevin**的通路，则设其**Bacon number**值为 infinity(无穷)

dataStruct

```
// 记录演员的一些属性
struct actorsFilm{
    vector<wstring> films; // 演员演过的电影
    int bacon; // 该演员的bacon值
    vector<pair<wstring, wstring>> path; // 记录了一条从Bacon, Kevin到该演员的路径
};
map<wstring, actorsFilm> actorToFilm; // （演员名，演员的一些属性）

map<wstring, bool> haveIn; // 主要用于后面bfs时，某个演员入队过后不会再入队

// 记录电影的一些属性
struct filmsActor{
    vector<wstring> actors; // 这部电影由哪些演员参演
};
```

```
map<wstring, filmsActor> filmToActor; // （电影名，电影的一些属性）

int baconTofre[20]; // 记录bacon number频率的数组
```

function(bfs)

```
// 用bfs的方式遍历每一个演员
void bfs()
{
    wstring st= L"Bacon, Kevin";
    actorToFilm[st].bacon = 0;
    actorToFilm[st].path.clear();
    baconTofre[0 + 1] = 1;
    haveIn[st] = true;


    queue<wstring> que;
    que.push(st);
    while (!que.empty())
    {
        st = que.front();
        que.pop();

        for (auto iter = actorToFilm[st].films.begin(); iter !=
actorToFilm[st].films.end(); iter++)
        {// 遍历当前演员st演过的电影
            for (auto iter2 = filmToActor[*iter].actors.begin(); iter2 !=
filmToActor[*iter].actors.end(); iter2++)
            {// 遍历该电影中的演员指针iter2
                if (haveIn[*iter2] != true) // 如果该演员iter2不在队列中
                {
                    que.push(*iter2); // 入队
                    actorToFilm[*iter2].bacon = actorToFilm[st].bacon + 1; // 设
定bacon值

                    // 设定最短路路径
                    for (auto n: actorToFilm[st].path)
                        actorToFilm[*iter2].path.push_back(n);
                    actorToFilm[*iter2].path.push_back(make_pair(*iter, st));

                    // baconnumber频率值相应加一
                    baconTofre[actorToFilm[*iter2].bacon + 1]++;

                    // 修改haveIn表示演员iter2已经入队
                    haveIn[*iter2] = true;
                }
            }
        }
    }
}
```

Code

PS: 本程序中用-1表示bacon值为infinity

main.cpp

```
#include <iostream>
#include <fstream>
#include <vector>
#include <map>
#include <queue>
#include <string>
#include <iomanip>
#include <ctime>
using namespace std;
const int MAXN = 1e8 + 10;

struct actorsFilm{
    vector<wstring> films;
    int bacon;
    vector<pair<wstring, wstring>> path;
};

map<wstring, actorsFilm> actorToFilm;
map<wstring, bool> haveIn;

struct filmsActor{
    vector<wstring> actors;
};

map<wstring, filmsActor> filmToActor;

int baconTofre[20];

void out();
void bfs();

wstring oneline;

int main(int argv, char *argc[])
{
    clock_t startTime,endTime;
    startTime = clock();

    wifstream in(argc[1]);
    // wofstream ou1(argc[2]);
    int num = 0;
    while (getline(in, oneline))
    {
        // ou1 << oneline << endl;
        if (oneline == L"")
            continue;
        unsigned long long n1 = 0;
        unsigned long long n = oneline.find(L'/');
        wstring film = oneline.substr(n1, n);
        // wcout << film << endl;
        wstring actor;
        filmsActor actors;
        n1 = n + 1;
        bool flag = false;

        while (n = oneline.find(L"/", n1))
        {
            //cout << n1 << " " << n << endl;
            actor = oneline.substr(n1, n - n1);
            if (actor == L"Bacon, Kevin")
                flag = true;
            //wcout << actor << endl;
            actors.actors.push_back(actor);

            if (actorToFilm.find(actor) != actorToFilm.end())
            {
                actorToFilm[actor].films.push_back(film);
            } else {
                actorsFilm asf;
                asf.bacon = -1;
                asf.films.push_back(film);

                actorToFilm[actor] = asf;
                haveIn[actor] = false;


            }
            if (n == wstring::npos)
                break;
            n1 = n + 1;
        }

        // if (!flag)
        //     wcout << oneline << endl;
        //if (filmToActor.find(film) == filmToActor.end())
        filmToActor[film] = actors;
        num++;
    }
    // cout << num << endl;
    // cout << filmToActor.size() << endl;
    bfs();
    endTime = clock();//计时结束
    cout << "The run time is: " <<(double)(endTime - startTime) / CLOCKS_PER_SEC
<< "s" << endl;

    out();
    return 0;
}

void out()
{
    cout << "Bacon number\t" << "Frequency" << endl;
    cout << "-------------------------" << endl;
    int s = 0;
    for (int i = 1; baconTofre[i] != 0; i++)
    {
        cout << setw(12) << i - 1 << setw(13) << baconTofre[i] << endl;
        s += baconTofre[i];
    }
    cout << setw(12) << "infinity" << setw(13) << haveIn.size() - s << endl;

    wstring actor;
    while (getline(wcin, actor))
    {
        if (actorToFilm.find(actor) == actorToFilm.end())
            cerr << "wrong actor" << endl;
        else
        {
            wcout << actor << L" has a Bacon number of " <<
actorToFilm[actor].bacon << endl;
            wstring actor2 = actor;
            for (auto iter = actorToFilm[actor].path.rbegin(); iter !=
actorToFilm[actor].path.rend(); iter++)
            {
                wcout << actor2 << " was in \"" << iter->first << "\" with " <<
iter->second << endl;
                actor2 = iter->second;
            }
        }
    }
    }
}


void bfs()
{
    wstring st= L"Bacon, Kevin";
    actorToFilm[st].bacon = 0;
    actorToFilm[st].path.clear();
    baconTofre[0 + 1] = 1;
    haveIn[st] = true;


    queue<wstring> que;
    que.push(st);
    while (!que.empty())
    {
        st = que.front();
        que.pop();

        for (auto iter = actorToFilm[st].films.begin(); iter !=
actorToFilm[st].films.end(); iter++)
        {
            for (auto iter2 = filmToActor[*iter].actors.begin(); iter2 !=
filmToActor[*iter].actors.end(); iter2++)
            {
                if (haveIn[*iter2] != true)
                {
                    que.push(*iter2);
                    actorToFilm[*iter2].bacon = actorToFilm[st].bacon + 1;

                    for (auto n: actorToFilm[st].path)
                        actorToFilm[*iter2].path.push_back(n);

                    actorToFilm[*iter2].path.push_back(make_pair(*iter, st));
                    baconTofre[actorToFilm[*iter2].bacon + 1]++;
                    haveIn[*iter2] = true;
                }
            }
        }
    }
}
```

Test Result

input8.txt

```
The run time is: 0s
Bacon number    Frequency
-------------------------
           0            1
           1            1
           2            4
           3            2
           4            1
           5            3
           6            3
    infinity            0
J
J has a Bacon number of 5
J was in "Movie 5" with H
H was in "Movie 4" with F
F was in "Movie 3" with E
E was in "Movie 2" with A
A was in "Movie 0" with Bacon, Kevin
```

input-all.txt

```
The run time is: 74.358s
Bacon number    Frequency
-------------------------
           0            1
           1         1494
           2       127774
           3       239608
           4        36455
           5         2963
           6          275
           7           39
           8           47
           9           99
          10           15
          11            2
    infinity         9696
Dane, Cynthia
Dane, Cynthia has a Bacon number of 3
Dane, Cynthia was in "Solstice (1994)" with Singer, Rachel
Singer, Rachel was in "Fight Club (1999)" with Andrews, David
Andrews, David was in "Apollo 13 (1995)" with Bacon, Kevin
```

input-bacon.txt

```
The run time is: 0.046s
Bacon number    Frequency
-------------------------
           0            1
           1         1494
    infinity            3
Doherty, Shannen
Doherty, Shannen has a Bacon number of -1
```

input-hero.txt

```
The run time is: 0.048s
Bacon number    Frequency
-------------------------
           0            1
           1           61
           2           25
           3           36
           4          119
           5          181
           6          248
           7           76
           8          302
           9          237
          10           43
          11            6
    infinity         1171
Blanc, Mel
Blanc, Mel has a Bacon number of -1
```

input-mpaa.txt

```
The run time is: 22.716s
Bacon number    Frequency
-------------------------
           0            1
           1         1372
           2        93798
           3        72980
           4         1636
           5           14
    infinity          717
Moranz, Jennifer
Moranz, Jennifer has a Bacon number of 2
Moranz, Jennifer was in "Radioland Murders (1994)" with Kroeger, Gary
Kroeger, Gary was in "Big Picture, The (1989)" with Bacon, Kevin
```

input-mpaa-g.txt

```
The run time is: 0.615s
Bacon number    Frequency
-------------------------
           0           1
           1          20
           2         686
           3        5378
           4        6163
           5         573
           6         103
           7          61
           8           3
    infinity         860
Abbott, Fredric
wrong actor
Allister, Claud
Allister, Claud has a Bacon number of 4
Allister, Claud was in "Adventures of Ichabod and Mr. Toad, The (1949)" with
Campbell, Colin
Campbell, Colin was in "National Velvet (1944)" with Lansbury, Angela
Lansbury, Angela was in "Beauty and the Beast (1991)" with Angel, Jack
Angel, Jack was in "Balto (1995)" with Bacon, Kevin
```

input-top-grossing.txt

```
The run time is: 0.486s
Bacon number    Frequency
-------------------------
           0           1
           1         160
           2        3626
           3        4361
           4         106
    infinity          10
Schwarzenegger, Arnold
Schwarzenegger, Arnold has a Bacon number of 2
Schwarzenegger, Arnold was in "Terminator 2: Judgment Day (1991)" with Berkeley,
Xander
Berkeley, Xander was in "Apollo 13 (1995)" with Bacon, Kevin
```

input-year2000.txt

```
The run time is: 2.052s
Bacon number    Frequency
-------------------------
           0           1
           1          73
           2        1375
           3       12770
           4       14016
           5        7341
           6        2096
           7         774
           8          116
           9           64
          10            2
    infinity         5295
Velasquez, Jaci
Velasquez, Jaci has a Bacon number of -1
```