目录

用图解决问题

一、题目及任务

任务 1

建立为实现该游戏的图的抽象描述结构，包括图中顶点的意义以及存储的信息、边的意义以及存储的信息。并给出该图的逻辑示意图。

任务 2

在任务 1 的基础上，并结合教材中图的抽象数据类型的定义，设计并实现一个为该游戏而使用的具体的 Graph Class。

任务 3

通过给定的数据文件 simple.txt 构建图。Simple.txt 中的格式为：每一行代表一个电影，每行中的信息都用‘/‘进行分割，其中第一个信息为电影名称，其后所有的信息都是出现在该电影的演员名。

利用图和相应的算法，你可以根据用户输入的演员名，给出该演员的 Bacon Number，并且列出该数计算的依据，也就是通过哪些电影建立了和 Kevin Bacon 的联系。运行的样例模式如下：

```
Welcome to the Six Degrees of Kevin Bacon.
If you tell me an actor's name, I'll connect them to Kevin Bacon through
the movies they've appeared in.  I bet your actor has a Kevin Bacon number
of less than six!

Actor's name (or ALL for everyone)? Brad Pitt

Path from Brad Pitt to Kevin Bacon:
Brad Pitt was in Ocean's Eleven (2001) with Julia Roberts
Julia Roberts was in Flatliners (1990) with Kevin Bacon
Brad Pitt's Bacon number is 2
```

任务 4

Complex.txt 是一个数据规模远超 Simple.txt 的文件，两者的格式是一样的。尝试使用任务 3 的解决方案执行 Complex.txt。如果运行效果不理想，请分析问题的原因，并尝试进行改变。

任务 5

在你的日常学习生活中，寻找一个可以用图解决的问题原型，描述该问题原型，并陈述如何将该问题原型抽象成图的表示。

二、问题分析

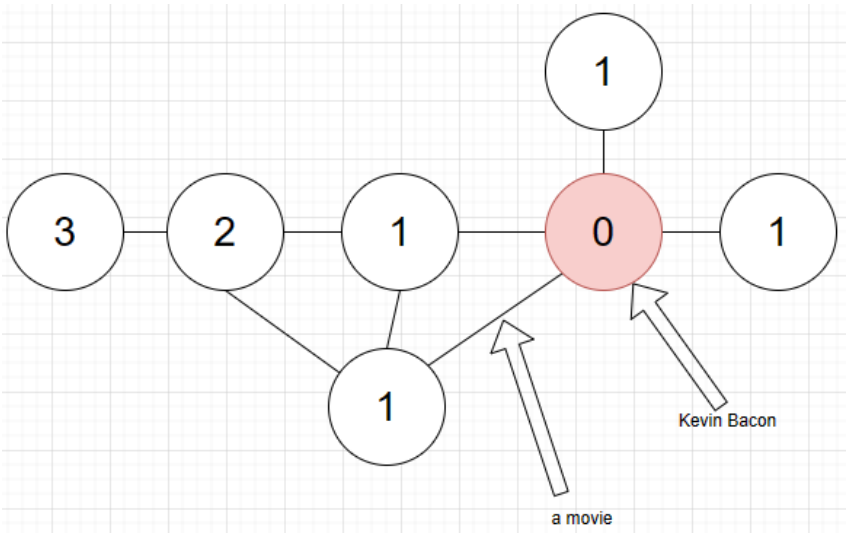
任务 1

2.1.1 图的抽象描述结构

图的每个顶点都代表一个演员以字符串存储；每条边代表一部电影，以字符串的形式存储。

两个顶点之间的边即代表两个演员曾都在该电影中出演

2.1.2 图的逻辑示意图



任务 2

在数据整理中发现输入数据属于稀疏图，尽管数据很小，但为了适用于任务四的大量数据，采用邻接表链式存储。

- 考虑到直接构建 String 的内存很大,用两个 Vector<String> starNames 和 movieNames 分别作为演员姓名和电影名的映射
- linkEdges - 用两个 Vector 模拟链表的行为从而构建邻接表，结点定义为一个新的类 LinkNode，包含当前节点对应的演员姓名和其与该链条演员的连接电影（均用整数表示，整数为其在 Vector 中的索引）

- dis[] - 表示每个节点离 start 节点的距离（本任务中一直为 Kevin Bacon）

任务 3

本次任务代码实现在 movieStarGraph.java
该任务可以分为三部分：输入数据并存储，BFS 确定距离，回溯得到路径

- 构造方法 - 构造方法从给定文件中逐行读取，用/分割开并添加到对应的数据结构，更改邻接表
- BFS - 用递归即可很简单地实现 BFS，函数接收两个参数分别为深度和结点的索引，则令该结点访问并赋深度作为距离，然后对所有未被访问过的子节点递归调用深度加一后的 BFS
- path - 输入一个结点，按要求输出路径信息和 Kevin Bacon 数字。对每个结点，遍历其邻接表上的每个结点直到找到一个结点其 dis 恰好比输入结点小一，然后对该结点递归调用 path
- 还需要一个函数用来模拟交互界面，较简单，此处不再赘述

任务 4

本次任务代码实现在 hashGraph.java
虽然最终解决方法很简单，但是从我的视角经历了很多波折，在我看来这些提出问题解决问题的过程比直接给出我认为这道题的最优解更宝贵，故依时间先后和对应的解决方法一同叙述如下：

1. 第一次构建 graph 类时，我采用邻接矩阵，只 Vector<String> 索引电影和演员名。但在第一次输入 Complex.txt 时直接报错堆溢出，把堆改大到 3-4 个 GB 也不够，于是我才改成了邻接表表示并通过。
2. 但在第一次输入数据时我发现运行很慢，debug 后诊断问题出现构造函数明显太慢了，数据读入每一百行要一两秒且会越来越慢，我一开始把问题归为两个原因
    - 堆内存还是太小 - 但在网上查阅一些资料后发现大多数人对读入数据的速度分析时，即便是很大的文件内存也在 2.5G 内，而从计算机原理的角度考虑，程序运行时装入内存，而读入文件常见的方式是把文件直接从磁盘调入内存。数据源文件的大小为 10MB，远小于我调给 IDEA 的堆内存。
    - 输入方式的问题 - 查阅网络并参考改变几个不同的读取文件形式后，发现效果有，但不大。这时我想起在第三次实验中，读入 article.txt 文件（30000+

行）似乎并未遇到这种问题，于是我把焦点放在构造函数中每次数据读入之后的数据处理上。

Listing 1: 数据输入后的处理代码

```
while (sc.hasNextLine()) {
    tmp = sc.nextLine();
    lines++;
    split = tmp.split("/");
    for (int i = 1; i < split.length; i++) {
        if (!starNames.contains(split[i])) {
            starNames.add(split[i]);
            linkEdges.add(new Vector<>());
        }
    }
    movieNames.add(split[0]);
    int edgIndex = movieNames.size() - 1;
    for (int i = 1; i < split.length; i++) {
        int startStarIndex = starNames.indexOf(split[i]);
        for (int j = 1; j < split.length; j++) {
            if (j == i) continue;
            linkEdges.get(startStarIndex).add(new LinkNode(edgIndex,
                starNames.indexOf(split[j])));
        }
    }
}
```

3. 经过分析不难发现，主要应该在 Vector 的性能问题上

   - 开始时我认为问题出在 Vector 的 capacity 每次扩容都需要搬运原来的 Vector，于是初始化设了一个 capacity，但收效甚微。
   - 虽然并没有解决问题，但却引发我思考：为什么一开始决定用 Vector 索引，如果 capacity 是固定的，这和 ArrayList 又有什么区别？（从某种角度说，Vector 为了线程安全的牺牲在当前的应用场景下是否还不如 ArrayList？）
   - 一开始使用 Vector 是觉得，底层为数组，从数字映射字符串会和哈希表一样快————到这时我才恍然大悟，数字映射到字符串并不是经过分析得到的需求

- 经过分析上面的代码不难发现，在输入过程中从字符串映射到数字占绝大部分，而基于数字映射字符串逻辑而构建的仿 hash 作用的 Vector，在处理字符串映射到数字时却需要 O(n)，这和哈希表其实是一样的。但我们无法基于 Vector 创建从字符串映射到数字的关系，这就要用到哈希表，我选择了 Java 自带的 HashMap

4. 但这还没有完，在任务三就解释过，程序主要分三部分，刚刚堆第一部分的优化是否会影响后两部分？经过分析不难发现：BFS 完全基于整数数组和邻接表处理，不会用到索引；而 path 确实需要索引，但正如题目所说的，每次 path 调用最多只需要反映射六七组（实际上在这次数据中有更多但不会超过 10）数字到字符串，这远小于构造函数中的需求。所以不会影响。

5. 在使用 HashMap 后，我依然遇到问题：比如原来定义的 100000 作为 MAXNUM 显然不够，演员有 34w；BST 在大量的结点下，原来的递归方法很容易导致栈溢出，于是需要重写一个非递归的 BFS。

任务 5

图是一种很精妙的构思，而图的解构更是这种思想很好的训练。现实生活中，尽管学习了图，大多数同学还是把眼光局限在那些似乎明显能构成图的，比如人与人之间的关系，比如城市和城市这类点到点的模型，事实上，一些似乎和图无关的场景也能借用图论的思想抽象并辅助解决。

图染色问题就是其中的杰出代表：总的来说，图染色适用于安排发生冲突的事件或资源以避免重叠和帮助调度。

一个比较明显的例子是地图中为彼此共享边界的不同区域，我们可以认为每个区域都是一个节点，相邻的区域对应到结点上会有一条边，从而用图论解决地图着色问题，如著名的四色问题。

稍微抽象的例子有活动/时间的安排. 以大学排课为例，可以用一个二部图来分别表示课程和教室，教室可以看作资源不同的课程不能在相同的时间占据相同的个教室，把时间抽象为边，这就又回到了图染色问题。

由上一个例子不难展现出图着色问题对于资源调度问题抽象后的巨大潜力，而计算机作为人类智慧的结晶之作，以操作系统为首的计算机底层软件需要对硬件做出合理的调度，也会常用到这种思想，其中最出名的就是广泛应用于编译器的图染色寄存器分配算法。这种计算机中最抽象的思想和最底层的现实的结合常常让我神往，这也是我我选择系统领域作为未来几年主要科研兴趣的主要原因。

三、主干代码说明

Listing 2: hashGraph 构造方法

```
hashGraph(String filename) throws IOException {
    // BufferedReader bf = new BufferedReader(new FileReader(filename));
    Arrays.fill(dis, Integer.MAX_VALUE);
    String tmp;
    String[] split;
    Scanner sc = new Scanner(new FileReader(filename));
    while (sc.hasNextLine()) {
        tmp = sc.nextLine();
        split = tmp.split("/");
        for (int i = 1; i < split.length; i++) {
            if (!actors.containsKey(split[i])) {
                actors.put(split[i],actors.size());
                linkEdges.add(new Vector<>());
            }
        }
        movies.put(split[0],movies.size());
        int edgIndex = movies.get(split[0]);
        // int edgIndex = movies.size() - 1;
        for (int i = 1; i < split.length; i++) {
            for (int j = 1; j < split.length; j++) {
                if (j == i) continue;
                linkEdges.get(actors.get(split[i])).add(new LinkNode(
                    edgIndex, actors.get(split[j])));
            }
        }
    }
}
```

Listing 3: 非递归 BFS

```
public void bfsNoRecursion(int start) {
    boolean[] visited = new boolean[actors.size()];
    ArrayDeque<depthInt> queue = new ArrayDeque<>();
    int dist;
    queue.add(new depthInt(0,start));
    visited[start] = true;
```

```
7       dis[start] = 0;
8       while (!queue.isEmpty()) {
9           depthInt k = queue.poll();
10          Vector<LinkNode> vc = this.linkEdges.get(k.actorIndex);
11          dist = k.depth + 1;
12          for (LinkNode ln : vc) {
13              if (!visited[ln.starIndex]) {
14                  queue.add(new depthInt(dist, ln.starIndex));
15                  visited[ln.starIndex] = true;
16                  dis[ln.starIndex] = dist;
17              }
18          }
19      }
20  }
```

Listing 4: path

```
1   public void path(int index) {
2       if (index == -1) return;
3       if (dis[index] == 0) return;
4       int next = -1;
5       Vector<LinkNode> stLink = linkEdges.get(index);
6       for (LinkNode linkNode : stLink) {
7           int st = linkNode.starIndex;
8           if (dis[st] == dis[index] - 1) {
9               next = st;
10              System.out.println("\t" + getkey(index,actors) + " was in " +
                    getkey(linkNode.edge,movies) + " with " + getkey(st,actors)
                    );
11              break;
12          }
13      }
14      path(next);
15  }
```

Listing 5: 递归 BFS

```
1   public void BFS(int depth, int start) {
2       dis[start] = depth;
```

```
    visited[start] = true;
    Vector<Integer> tmp = new Vector<>();
    Vector<LinkNode> stLink = linkEdges.get(start);
    for (LinkNode linkNode : stLink)
        if (!visited[linkNode.starIndex])
            tmp.add(linkNode.starIndex);
    for (int i : tmp) BFS(depth + 1, i);
}
```

四、运行结果展示

Simplex.txt

```
Welcome to the Six Degrees of Kevin Bacon.
If you tell me an actor's name, I'll connect them to Kevin Bacon through the movies they've appeared in.
I bet your actor has a Kevin Bacon Number of less than 6!

Please input the actor's name(or ALL for everyone)? Brad Pitt
Path from Brad Pitt to Kevin Bacon:
    Brad Pitt was in Ocean's Eleven (2001) with Julia Roberts
    Julia Roberts was in Flatliners (1990) with Kevin Bacon
Brad Pitt's Bacon number is 2
```

Complex.txt

由于 Complex.txt 文件中并未包含 Kevin Bacon 或 Kevin, Bacon. 所以优化了接口，改为可以计算以任何人为中心的度数，并选择了一个在 Complex.txt 中出现过四次的演员 Sheahan, Norma 作为中心，随机选取了另一个演员 Lerchenberg, Michael 作测试

```
Welcome to the Six Degrees of Sheahan, Norma.
If you tell me an actor's name, I'll connect them to + Sheahan, Norma through the movies they've appeared in.
I bet your actor has a Sheahan, Norma Number of less than 6!
PS: Actually, I'm not so sure because the data set is much bigger and the actress may not be the famous Kevin Bacon. :)

Please input the actor's name(or ALL for everyone | or test for check out if there is a degree more than 6)? Lerchenberg, Michael
Path from Lerchenberg, Michael to Kevin Bacon:
    Lerchenberg, Michael was in NaPolA (2004) with Svarc, Bohumil
    Svarc, Bohumil was in Hart's War (2002) with Farrell, Colin (I)
    Farrell, Colin (I) was in Intermission (2003) with Sheahan, Norma
Lerchenberg, Michael's Bacon number is 3

Please input the actor's name(or ALL for everyone | or test for check out if there is a degree more than 6)?
```

基于对这个有趣游戏的兴趣，我认为接口实现上在课程要求之外还有些不足，实现如下：

- test ALL - 用来验证六度空间理论，找到一共有多少演员满足该理论可以作为中心点，也可以改为打印出所有符合要求的演员名
- test - 在当前中心下检查有多少演员不符合六度空间理论

五、总结和收获

本次实验的前几部分都很简单，不需要花太大功夫。

第四部分的各种问题让我体验到久违的探索问题发现问题解决问题的快感，也让我更加掌握了“种”程序（即在程序分析理论中一层层分析验证程序正确性的方法），尤其是本次出现的问题会让我以后在下意识使用 Vector 作为映射前思考映射是什么，我需要的是什么，再确定怎样的才是高效的。

第五部分的很多东西在一年前学习算法理论时并没有很深地认识到，但在刚刚学习完编译原理和编译器构造后，其中一些思想才领悟得更深刻，这愈发体现出学习代码和感悟比起一味灌知识点和上层思想的重要性。

这些都对我以后的学习方法很有影响。

附录：源代码汇总

movieStarGraph.java

Listing 6: movieStarGraph.java

```
import java.io.BufferedReader;
import java.io.FileInputStream;
import java.io.FileReader;
import java.io.IOException;
import java.time.LocalTime;
import java.util.Arrays;
import java.util.HashMap;
import java.util.Scanner;
import java.util.Vector;

public class movieStarGraph {

    private class LinkNode {
        int edge;
        int starIndex;

        // if edge = -1, it means this is the first one of a link;
        LinkNode(int e, int s) {
            edge = e;
            starIndex = s;
        }
    }

    Vector<Vector<LinkNode>> linkEdges = new Vector<>();

    static final int MAXNUM = 50000; // simple.txt has or so 61 lines
    int[] dis = new int[MAXNUM]; // distance from bst
    boolean[] visited = new boolean[MAXNUM];
    Vector<String> starNames = new Vector<>(MAXNUM);
```

```
30      Vector<String> movieNames = new Vector<>(MAXNUM);
31      HashMap<Integer, String> has = new HashMap<>(MAXNUM);
32
33      movieStarGraph(String filename) throws IOException {
34          // BufferedReader bf = new BufferedReader(new FileReader(filename))
                ;
35          Arrays.fill(dis, -1);
36
37          String tmp;
38          String[] split;
39          int lines = 0;
40          Scanner sc = new Scanner(new FileReader(filename));
41          while (sc.hasNextLine()) {
42              tmp = sc.nextLine();
43              lines++;
44              split = tmp.split("/");
45              for (int i = 1; i < split.length; i++) {
46                  if (!starNames.contains(split[i])) {
47                      starNames.add(split[i]);
48                      linkEdges.add(new Vector<>());
49                  }
50              }
51              movieNames.add(split[0]);
52              int edgIndex = movieNames.size() - 1;
53              for (int i = 1; i < split.length; i++) {
54                  int startStarIndex = starNames.indexOf(split[i]);
55                  for (int j = 1; j < split.length; j++) {
56                      if (j == i) continue;
57                      linkEdges.get(startStarIndex).add(new LinkNode(edgIndex
                            , starNames.indexOf(split[j])));
58                  }
59              }
60              if (lines % 500 == 0) {
61                  System.out.println("has read " + lines);
62              }
63          }
```

```
64      }
65
66      public static void main(String[] args) throws IOException {
67          movieStarGraph msg = new movieStarGraph("../Simple.txt");
68          msg.BFS(0, msg.starNames.indexOf("Kevin Bacon"));
69          msg.sixDegreeShell();
70      }
71
72      /**
73       * BFS started with given index, after BFS, dis[] values should be set.
74       *
75       * @param depth the depth of BFS, this is used for initialize
76       * @param start start index
77       */
78      public void BFS(int depth, int start) {
79          dis[start] = depth;
80          visited[start] = true;
81          Vector<Integer> tmp = new Vector<>();
82          Vector<LinkNode> stLink = linkEdges.get(start);
83          for (LinkNode linkNode : stLink)
84              if (!visited[linkNode.starIndex])
85                  tmp.add(linkNode.starIndex);
86          for (int i : tmp) BFS(depth + 1, i);
87      }
88
89      /**
90       * based on given dst[] (best after BFS), get the path to start vertex
             and print as requested
91       * @param index - index of the star name.
92       */
93      public void path(int index) {
94          if (index == -1) return;
95          if (dis[index] == 0) return;
96          int next = -1;
97          Vector<LinkNode> stLink = linkEdges.get(index);
98          for (LinkNode linkNode : stLink) {
```

```
99              int st = linkNode.starIndex;
100             if (dis[st] == dis[index] - 1) {
101                 next = st;
102                 System.out.println("\t" + starNames.get(index) + " was in "
                        + movieNames.get(linkNode.edge) + " with " + starNames
                        .get(st));
103                 break;
104             }
105         }
106         path(next);
107     }
108
109     public void sixDegreeShell(){
110         System.out.print("""
111                 Welcome to the Six Degrees of Kevin Bacon.
112                 If you tell me an actor's name, I'll connect them to Kevin
                        Bacon through the movies they've appeared in.
113                 I bet your actor has a Kevin Bacon Number of less than 6!
114                 """);
115         while(true){
116             System.out.print("\nPlease input the actor's name(or ALL for
                    everyone)? ");
117             Scanner sc = new Scanner(System.in);
118             String str = sc.nextLine();
119             if (str.equals("ALL")) {
120                 for (int i = 0; i < this.starNames.size(); i++) {
121                     System.out.println("\nPath from " + this.starNames.get(
                            i) + " to Kevin Bacon:");
122                     this.path(i);
123                     System.out.println(this.starNames.get(i) + "'s Bacon
                            number is " + this.dis[i]);
124                 }
125             }else if (str.equals("exit")) return;
126             else {
127                 int index = this.starNames.indexOf(str);
128                 System.out.println("Path from " + this.starNames.get(index)
```

```
                    + " to Kevin Bacon:");
129                 this.path(index);
130                 System.out.println(this.starNames.get(index) + "'s Bacon
                        number is " + this.dis[index]);
131             }
132         }
133
134     }
135
136 }
```

hashGraph.java

Listing 7: hashGraph.java

```
1  import java.io.FileReader;
2  import java.io.IOException;
3  import java.util.*;
4
5  public class hashGraph {
6
7
8      private class LinkNode {
9          int edge;
10         int starIndex;
11
12         // if edge = -1, it means this is the first one of a link;
13         LinkNode(int e, int s) {
14             edge = e;
15             starIndex = s;
16         }
17     }
18
19     Vector<Vector<LinkNode>> linkEdges = new Vector<>();
20
21     static final int MAXNUM = 400000; // simple.txt has or so 61 lines
```

```
22      int[] dis = new int[MAXNUM]; // distance from bst
23
24      HashMap<String, Integer> actors = new HashMap<>(MAXNUM);
25      HashMap<String, Integer> movies = new HashMap<>(MAXNUM);
26
27      hashGraph(String filename) throws IOException {
28          // BufferedReader bf = new BufferedReader(new FileReader(filename))
                ;
29          Arrays.fill(dis, Integer.MAX_VALUE);
30          String tmp;
31          String[] split;
32          Scanner sc = new Scanner(new FileReader(filename));
33          while (sc.hasNextLine()) {
34              tmp = sc.nextLine();
35              split = tmp.split("/");
36              for (int i = 1; i < split.length; i++) {
37                  if (!actors.containsKey(split[i])) {
38                      actors.put(split[i],actors.size());
39                      linkEdges.add(new Vector<>());
40                  }
41              }
42              movies.put(split[0],movies.size());
43              int edgIndex = movies.get(split[0]);
44              // int edgIndex = movies.size() - 1;
45              for (int i = 1; i < split.length; i++) {
46                  for (int j = 1; j < split.length; j++) {
47                      if (j == i) continue;
48                      linkEdges.get(actors.get(split[i])).add(new LinkNode(
                            edgIndex, actors.get(split[j])));
49                  }
50              }
51          }
52      }
53
54      public static void main(String[] args) throws IOException {
55          hashGraph hg = new hashGraph("../Complex.txt");
```

```
56          hg.bfsNoRecursion(hg.actors.get("Sheahan, Norma"));
57          hg.sixDegreeShell("Sheahan, Norma");
58      }
59
60      public void bfsNoRecursion(int start) {
61          boolean[] visited = new boolean[actors.size()];
62          ArrayDeque<depthInt> queue = new ArrayDeque<>();
63          int dist;
64          queue.add(new depthInt(0,start));
65          visited[start] = true;
66          dis[start] = 0;
67          while (!queue.isEmpty()) {
68              depthInt k = queue.poll();
69              Vector<LinkNode> vc = this.linkEdges.get(k.actorIndex);
70              dist = k.depth + 1;
71              for (LinkNode ln : vc) {
72                  if (!visited[ln.starIndex]) {
73                      queue.add(new depthInt(dist, ln.starIndex));
74                      visited[ln.starIndex] = true;
75                      dis[ln.starIndex] = dist;
76                  }
77              }
78          }
79      }
80
81      /**
82       * based on given dst[] (best after BFS), get the path to start vertex
             and print as requested
83       *
84       * @param index - index of the star name.
85       */
86      public void path(int index) {
87          if (index == -1) return;
88          if (dis[index] == 0) return;
89          int next = -1;
90          Vector<LinkNode> stLink = linkEdges.get(index);
```

```
91          for (LinkNode linkNode : stLink) {
92              int st = linkNode.starIndex;
93              if (dis[st] == dis[index] - 1) {
94                  next = st;
95                  System.out.println("\t" + getkey(index,actors) + " was in "
                        + getkey(linkNode.edge,movies) + " with " + getkey(st,
                       actors));
96                  break;
97              }
98          }
99          path(next);
100     }
101
102     public void sixDegreeShell(String name) {
103         System.out.println("THIS IS HASH_GRAPH");
104         System.out.println("Welcome to the Six Degrees of " + name + ".");
105         System.out.println("If you tell me an actor's name, I'll connect
               them to + " + name + " through the movies they've appeared in."
               );
106         System.out.println("I bet your actor has a " + name + " Number of
               less than 6!");
107         System.out.println("PS: Actually, I'm not so sure because the data
               set is much bigger and the actress may not be the famous Kevin
               Bacon. :)");
108         while (true) {
109             System.out.print("\nPlease input the actor's name(or ALL for
                   everyone)? ");
110             Scanner sc = new Scanner(System.in);
111             String str = sc.nextLine();
112             if (str.equals("ALL")) {
113                 for (int i = 0; i < this.actors.size(); i++) {
114                     String s = getkey(i, actors);
115                     System.out.println("\nPath from " + s + " to " + name +
                            " :");
116                     this.path(i);
117                     System.out.println(s + "'s Bacon number is " + this.dis
```

```
                    [i]);
118             }
119         } else if (str.equals("exit")) return;
120         else if(str.equals("test ALL")){
121             int num = 0;
122             boolean[] isMore = new boolean[actors.size()];
123             Arrays.fill(isMore,false);
124             for(int i  = 0;i<actors.size();i++) {
125                 if(isMore[i]) continue;
126                 this.bfsNoRecursion(i);
127                 int k;
128                 for(k  = 0;k<actors.size();k++){
129                     if(dis[k] > 6) {
130                         isMore[i] = true;
131                         isMore[k] = true;
132                         break;
133                     }
134                 }
135                 if(k == actors.size()) num++;
136                 if(i%50 == 0) System.out.println(i);
137             }
138             System.out.println("result " + num);
139         } else if (str.equals("test")){
140             int j = 0, k = 0;
141             for(int i  = 0;i<actors.size();i++) {
142                 if(dis[i] == Integer.MAX_VALUE)  j++;
143                 else if(dis[i] > 6) k++;
144             }
145             System.out.println(j + " "+k);
146         } else {
147             int index = this.actors.get(str);
148             System.out.println("Path from " + str + " to Kevin Bacon:")
                    ;
149             this.path(index);
150             System.out.println(str + "'s Bacon number is " + this.dis[
                    index]);
```

```
151             }
152         }
153     }
154 
155     private static String getkey(int value, HashMap<String,Integer> hash) {
156         for (HashMap.Entry<String, Integer> entry : hash.entrySet())
157             if (entry.getValue().equals(value))
158                 return entry.getKey();
159         return null;
160     }
161 
162     private class depthInt {
163         int depth;
164         int actorIndex;
165 
166         depthInt(int d,int i){
167             depth = d;
168             actorIndex = i;
169         }
170     }
171 
172 }
```