数据结构与算法
实验报告

第 4 次

姓名	丁昌灏
班级	软件 72 班
学号	2174213743
电话	15327962577
Email	dch0031@stu.xjtu.edu.cn
日期	2020-12-12

目录

实验 1

1、题目

建立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。

2、数据结构设计

本题描述一个单词转换，用图描述这个过程，则顶点代表每个单词，存储单词的值；边则用有向边表示，边表示替换字母的位置以及替换后的字母，弧头的单词由弧尾的单词转换一个字母得到。

3、运行结果展示

要描述的问题	顶点表示的意义	顶点存储的信息	边表示的意义	边存储的信息
单词顺序转换	某个单词	单词的值	字母转换过程	字母索引位置以及替换后的字母

逻辑示意图如下

	COLD	CORD	CARD	WARD	WARM	CORM	WORM	WORD	WOLD	WALD
COLD		3:R							1:W	
CORD	3:L		2:A			4:M		1:W		
CARD		2:O		1:W						
WARD			1:C		4:M			2:O		3:L
WARM				4:D			2:O			
CORM		4:D					1:W			
WORM					2:A	1:C		4:D		
WORD		1:C		2:A			4:M		3:L	
WOLD	1:C							3:R		2:A
WALD				3:R					2:O	

实验 2

1、题目

在任务 1 的基础上，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。

2、数据结构设计

该题目中的图以相邻矩阵的形式保存，为了保存方便，建立一个标号与结点对应的哈希表。结点的定义如下

```java
public class Vertexm implements Vertex {

    private String value;

    public Vertexm(String value) {
        this.value = value;
    }

    @Override
    public String value() {
        return value;
    }

}
```

边的定义如下：

```java
public class Edgem implements Edge {

    private int vert1, vert2;

    public Edgem(int vert1, int vert2) {
        this.vert1 = vert1;
        this.vert2 = vert2;
    }

    @Override
    public int v1() {
        return vert1;
    }

    @Override
    public int v2() {
        return vert2;
    }

}
```

图的定义如下：

其中成员主要为：相邻矩阵、边数、标号与结点的映射表以及一个标记数组

函数除了课本中定义的之外，多加入判断两个结点是否联通，寻找一个结点到另一个结点的所有通路。在算法设计以及运行时详细讲解

3、算法设计

判断两个结点是否联通：这个利用递归的思想，如果 A 与 B 联通，则与 A 联通的点中至少有一个点与 B 联通，采用深度优先思想递归查找，在查找过程中为了避免重复查找造成死循环，查过的点都要进行标记。

找到两个结点所有的通路：也是采用递归的思想，与上面类似，为了避免重复查找造成的死循环，对于已经加入路径的点需要进行标记，当重新查找时删除标记。如果发现有些结点一定无法到达最后的结点，则进行标记且不删除。

4、主干代码说明

图的属性已经构造器：

```
private int[][] matrix;
private int numEdge;
private Map<Integer, Vertex> map; // 序号与结点的映射
public int[] Mark;

public Graphm(int n) {
    Mark = new int[n];
    matrix = new int[n][n];
    numEdge = 0;
    map = new HashMap<Integer, Vertex>();
}

public Graphm(Map<Integer, Vertex> map) {
    Mark = new int[map.size()];
    matrix = new int[map.size()][map.size()];
    numEdge = 0;
    this.map = map;
}
```

其余方法与教材相同就不再一一放出，下面列举加入的判断联通和查找所有路径的方法

判断两个结点是否联通：

```
private boolean isLinkedHelp(int v1, int v2) {
    if (getMark(v1) != 0) {
        return false;
    }
    if (isAlone(v1) || isAlone(v2)) {
        return false;
    }

    if (matrix[v1][v2] != 0) {
        setMark(v1, 1);
        setMark(v2, 1);
        return true;
    }

    setMark(v1, 1);

    for (int i = 0; i < n(); i++) {
        if (getMark(i) != 0) {
            continue;
        }

        if (matrix[v1][i] != 0) {
            if (isLinkedHelp(i, v2)) {
                return true;
            }
        }
    }
    return false;
}
```

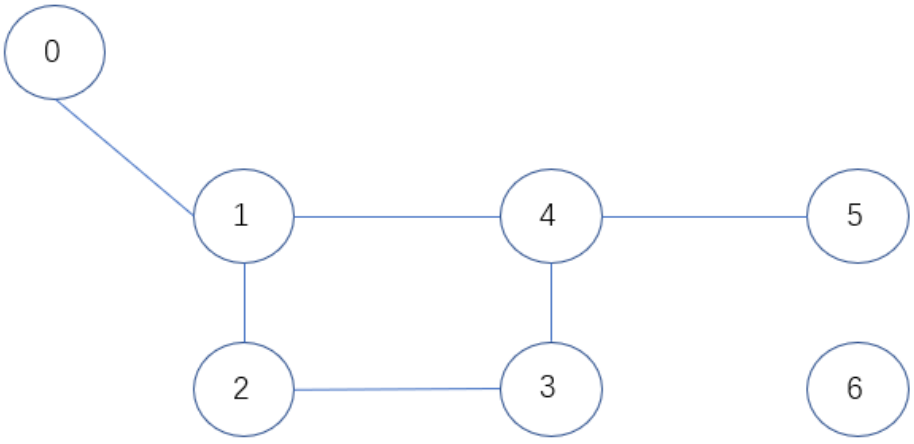
查找两个点之前所有的通路：

```
private void findAllHelp(int v1, int v2, List<Integer> list,
List<List<Integer>> listSet) {
        if (isAlone(v1)) {
            setMark(v1, 1);
            return;
        }
        if (isAlone(v2)) {
            setMark(v2, 1);
            return;
        }
        if (matrix[v1][v2] != 0) {
            if (!list.contains(v1)) {
                list.add(v1);
                setMark(v1, 1);
            }
            if (!list.contains(v2)) {
                list.add(v2);
                setMark(v2, 1);
            }
            listSet.add(new ArrayList<>(list));
            list.remove(list.size() - 1);
            setMark(v2, 0);
            return;
        }

        if (!list.contains(v1)) {
            list.add(v1);
            setMark(v1, 1);
        }

        for (int i = 0; i < n(); i++) {
            if (getMark(i) != 0) {
                continue;
            }
            if (matrix[v1][i] != 0) {
                int[] temp = new int[Mark.length];
                for (int m = 0; m < Mark.length; m++) {
                    temp[m] = Mark[m];
                }
                if (isLinked(i, v2)) {
                    for (int m = 0; m < Mark.length; m++) {
                        Mark[m] = temp[m];
                    }
```

```
                findAllHelp(i, v2, list, listSet);
                list.remove(list.size() - 1);
                setMark(i, 0);
            } else {
                setMark(i, 1);
            }
        }
    }
    return;
}
```

5、运行结果展示

首先验证两个算法的正确性，我们采用如下图



测试代码如下：找 0 和 5 之间的通路

```
    public static void main(String[] args) {
        Map<Integer, Vertex> map = new HashMap<>();
        for (int i = 0; i < 7; i++) {
            map.put(i, new Vertexm(""));
        }
        Graphm graph = new Graphm(map);
        graph.setEdge(0, 1, 1);

        graph.setEdge(1, 0, 1);
        graph.setEdge(1, 2, 1);
        graph.setEdge(1, 4, 1);

        graph.setEdge(2, 1, 1);
        graph.setEdge(2, 3, 1);

        graph.setEdge(3, 2, 1);
        graph.setEdge(3, 4, 1);

        graph.setEdge(4, 1, 1);
        graph.setEdge(4, 3, 1);
        graph.setEdge(4, 5, 1);

        graph.setEdge(5, 4, 1);

        List<Integer> list = new ArrayList<>();
        List<List<Integer>> set = new ArrayList<>();
        graph.findAll(0, 5, list, set);
        System.out.println(list);
        System.out.println(set);
    }
```

运行结果：

```
<terminated> Test (6) [Java Application] D:\Java\jdk-13\bin\
[0]
[[0, 1, 2, 3, 4, 5], [0, 1, 4, 5]]
```

实验 3

1、题目

该任务中会提供一个所有长度为 5 的单词列表文件 words5.txt，需要针对提供的这个单词列表解决如下问题：

① 针对 words5.txt 文件中的单词列表，生成一个 noladder.txt 文件，该文件中记录的单词是无法和其他单词形成字梯的所有单词。

② 编写一个具有交互功能的程序，给用户随机抽两个单词（注：这两个单词必须要保证能够有字梯链），接受用户的输入，判断用户的每次输入是否是正确的，直到用户失败或者成功。（如果可能，还可以增加判断用户的成功输入是否是最短的变化链路）

2、数据结构设计

不能与其他单词形成字梯，也就是相邻矩阵中该单词所在的行均为 0，如果存在 1 说明至少可以与一个单词形成字梯。

3、算法设计

每个单词依次与其他单词进行比较，如果相差在一个字母，则这两个单词之间能够形成字梯，若某个单词一个相邻都没有，则他不能与其他单词形成字梯。

第二题做了一个简单的 GUI，用户输入后会在语料库(words5.txt)中查找，来判断是否满足条件，具体在代码说明中进行解释。

4、主干代码说明

1）判断两个单词是否能形成字梯：

```
public static boolean isRight(String word1, String word2) {
    if (word1.length() != word2.length()) {
        return false;
    }
    int count = 0;
    for (int i = 0; i < word1.length(); i++) {
        if (word1.charAt(i) != word2.charAt(i)) {
            count++;
        }
    }
    return count == 1;
}
```

将不能形成字梯的写入到文件：

```
public static void main(String[] args) {
      File file = new
File("src//com//pickupppp//task4//words5.txt");
      BufferedReader br = null;
      Map<Integer, Vertex> map = new HashMap<>();
      BufferedWriter bw = null;
      BufferedWriter bw2 = null;
      try {
         br = new BufferedReader(new InputStreamReader(new
FileInputStream(file)));
         String word = "";
         int index = 0;
         while ((word = br.readLine()) != null) {
            map.put(index, new Vertexm(word));
            index++;
         }
         Graphm graph = new Graphm(map);
         for (int i = 0; i < graph.n(); i++) {
            for (int j = 0; j < graph.n(); j++) {
               if (isRight(map.get(i).value(),
map.get(j).value())) {
```

```
                    graph.setEdge(i, j, 1);
                }
            }
        }
        File file1 = new
File("src//com//pickupppp//task4//noladder.txt");
        File file2 = new
File("src//com//pickupppp//task4//others.txt");
        bw = new BufferedWriter(new OutputStreamWriter(new
FileOutputStream(file1)));
        bw2 = new BufferedWriter(new OutputStreamWriter(new
FileOutputStream(file2)));
        for (int i = 0; i < graph.n(); i++) {
            if (graph.isAlone(i)) {
                bw.write(map.get(i).value());
                bw.write("\n");
            } else {
                bw2.write(map.get(i).value());
                bw2.write("\n");
            }
        }
        bw.flush();
        bw2.flush();
    } catch (FileNotFoundException e) {
        e.printStackTrace();
    } catch (IOException e) {
        e.printStackTrace();
    } finally {
        if (null != bw2) {
            try {
                bw2.close();
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
        if (null != bw) {
            try {
                bw.close();
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
        if (null != br) {
            try {
```

```
                br.close();
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
    }
}
```

2）GUI 界面

布局部分：这部分主要是 GUI 界面的一些元素

```
private JTextField fie1, fie2, fie3, fie4;
    private JTextArea are1;
    private Graphm graph;
    private JButton btn;
    /**
     *
     */
    private static final long serialVersionUID = 1L;

    public GameWindow() {
        setDefaultCloseOperation(EXIT_ON_CLOSE);
        setBounds(100, 100, 450, 300);
        setLayout(new FlowLayout());

        JLabel lab1 = new JLabel("start word:");
        fie1 = new JTextField(8);
        fie1.setEditable(false);
        add(lab1);
        add(fie1);

        JLabel lab2 = new JLabel("end word:");
        fie2 = new JTextField(8);
        fie2.setEditable(false);
        add(lab2);
        add(fie2);

        JLabel lab3 = new JLabel("your answer:");
        are1 = new JTextArea(8, 30);
        are1.setEditable(false);
        add(lab3);
        add(are1);

        JLabel lab4 = new JLabel("please enter your answer:");
        fie3 = new JTextField(12);
```

```
        fie3.setEditable(false);
        fie3.addActionListener(new computeWord());
        add(lab4);
        add(fie3);

        btn = new JButton("start game");
        btn.addActionListener(new StartGame());
        add(btn);

        JLabel lab5 = new JLabel("game status:");
        fie4 = new JTextField(20);
        fie4.setText("please enter the button");
        fie4.setEditable(false);
        add(lab5);
        add(fie4);
    }
```

游戏开始，按钮的事件监听部分：

点击开始，会从单词中找到两个可以形成字梯的单词，然后初始化整个 GUI

```
class StartGame implements ActionListener {
        @Override
        public void actionPerformed(ActionEvent e) {
            File file = new
File("src//com//pickupppp//task4//words5.txt");
            BufferedReader br = null;
            Map<Integer, Vertex> map = new HashMap<>();
            try {
                br = new BufferedReader(new
InputStreamReader(new FileInputStream(file)));
                String word = "";
                int index = 0;
                while ((word = br.readLine()) != null) {
                    map.put(index, new Vertexm(word));
                    index++;
                }
                graph = new Graphm(map);
                for (int i = 0; i < graph.n(); i++) {
                    for (int j = 0; j < graph.n(); j++) {
                        if (Utils.isRight(map.get(i).value(),
map.get(j).value())) {
                            graph.setEdge(i, j, 1);
                        }
                    }
                }
                int v1 = -1;
```

```
            int v2 = -1;
            List<Integer> list = new ArrayList<>();
            do {
                v1 = (int) (Math.random() * map.size());
                v2 = (int) (Math.random() * map.size());
            } while (!graph.isLinked(v1, v2));

            graph.isLinked(v1, v2, list);
            System.out.println(list);
            System.out.println(map.get(v1).value() + "--->"
+ map.get(v2).value());
            for (int temp : list) {
                System.out.print(map.get(temp).value() + "-
->");
            }

            fie1.setText(map.get(v1).value());
            fie2.setText(map.get(v2).value());
            fie3.setEditable(true);
            are1.setText(fie1.getText());
        } catch (FileNotFoundException e1) {
            e1.printStackTrace();
        } catch (IOException e1) {
            e1.printStackTrace();
        } finally {
            if (null != br) {
                try {
                    br.close();
                } catch (IOException e1) {
                    e1.printStackTrace();
                }
            }
        }
    }
```

监听用户输入已经判断游戏状态部分：

当用于输入完单词按下回车后触发，如果用户输入正确，则提示正确；如果错误则提示错误，相应的开始按钮变成重新开始；如果输入到了最后结果则会提示获胜，开始按钮也变成重新开始。

```
class computeWord implements ActionListener {
    @Override
    public void actionPerformed(ActionEvent e) {
        String word = fie3.getText();
        String[] words = are1.getText().split("-->");
```

```
            String word1 = words[words.length-1];
            fie3.setText("");
            System.out.println(word1);
            if (Utils.isRight(word1, word) &&
graph.isLinked(fie1.getText(), word)) {
                if (graph.isEdge(word, fie2.getText())) {
                    are1.setText(are1.getText() + "-->" + word +
"-->" + fie2.getText());
                    fie4.setText("you win, enter button restart
game");
                    fie3.setEditable(false);
                    btn.setText("restart game");
                } else if (graph.isLinked(word, fie2.getText()))
{
                    are1.setText(are1.getText() + "-->" + word);
                    fie4.setText("you are right, please enter
next word");
                } else {
                    are1.setText(are1.getText() + "-->" + word);
                    fie4.setText("you lose, enter button restart
game");
                    btn.setText("restart game");
                    fie3.setEditable(false);
                }
            } else {
                are1.setText(are1.getText() + "-->" + word);
                fie4.setText("you lose, enter button restart
game");
                btn.setText("restart game");
                fie3.setEditable(false);
            }
        }

    }
```

5、运行结果展示

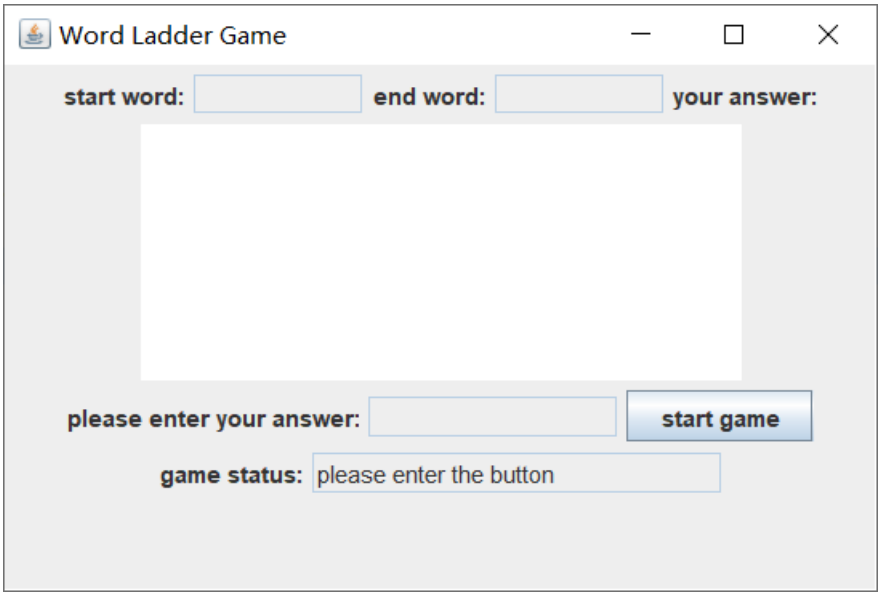
1）所有无法形成字梯的单词共 602 个，完整版见附件”noladder.txt”

```
aback
abbas
abbey
abbot
abyss
acrid
actor
acute
adage
added
addle
adieu
adult
aegis
affix
afire
afoot
afoul
again
agile
aging
agree
ahead
```

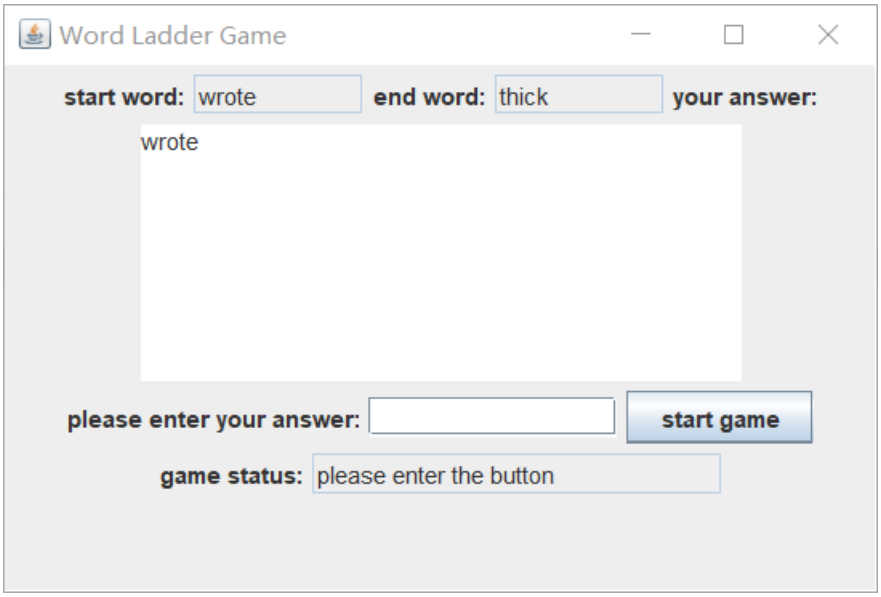
2)运行代码

```
public static void main(String[] args) {
    GameWindow game = new GameWindow();
    game.setTitle("Word Ladder Game");
    game.setVisible(true);
}
```

运行界面



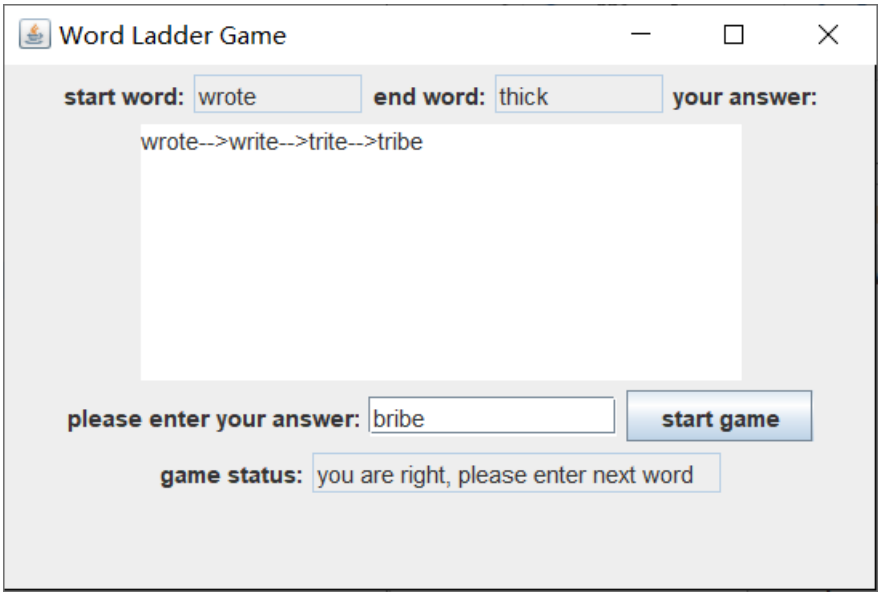
点击 start game 开始游戏



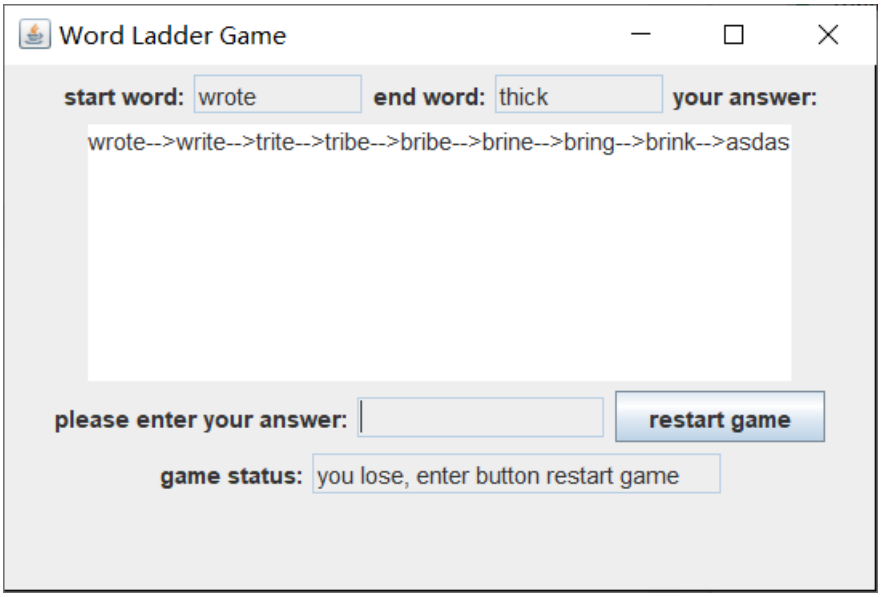
控制台输出游戏秘籍

```
wrote--->thick
wrote-->write-->trite-->tribe-->bribe-->bride-->brine-->bring-->brink-->blink-->blank-
```

对照输入



如果输入错误



正确通关(选取一条路径比较短的演示)