算法设计report

Name: 崔子璇
StudentID: 12311007

March 13, 2025

Contents

1 Task1:Gale-Shapley Algorithm Implementation

1.1 sectionimplement the GS algorithms

```java
//package week02.lab;

import java.util.HashMap;
import java.util.Map;
import java.util.Scanner;

public class main {
    public static void main(String[] args) {
        long startTime = System.currentTimeMillis(); // Record start time
        Scanner in = new Scanner(System.in);
        int n = in.nextInt();
        HashMap<String, Node> boysMap = new HashMap<>();
        HashMap<String, Node> girlsMap = new HashMap<>();
        Node[] boys = new Node[n];
        for (int i = 0; i < n; i++) {
            String name = in.next();
            boys[i] = new Node(name, n);
            boysMap.put(name, boys[i]);
        }
        Node[] girls = new Node[n];
        for (int i = 0; i < n; i++) {
            String name = in.next();
            girls[i] = new Node(name, n);
            girlsMap.put(name, girls[i]);
        }
        for (int i = 0; i < n; i++) {
            for (int j = 0; j < n; j++) {
                String name = in.next();
                boys[i].likes[j] = girlsMap.get(name);
            }
        }
        for (int i = 0; i < n; i++) {
            for (int j = 0; j < n; j++) {
                String name = in.next();
                girls[i].likes[j] = boysMap.get(name);
            }
        }
        HashMap<String, String> ans = new HashMap<>();
        // for (Node node : boys) {
        // System.out.print(node.name + " " + Arrays.toString(node.likes));
        // }
        // System.out.println();
        // for (Node node : girls) {
        // System.out.print(node.name + " " + Arrays.toString(node.likes));
        // }
        while (!allDated(boys)) {
            for (Node boy : boys) {
                if (boy.ifFree) {
                    for (Node girl : boy.likes) {
                        if (girl.ifFree) {
                            ans.remove(boy.name);
                            boy.date(girl);
                            ans.put(boy.name, girl.name);
                            break;
                        } else {
                            if (prefer(girl, boy)) {
                                continue;
                            } else {
                                ans.remove(girl.date.name);
                                girl.date(boy);
                                ans.put(boy.name, girl.name);
                                break;
                            }
                        }
                    }
                }
            }
        }
        for (Node boy : boys) {
            if (ans.containsKey(boy.name)) { // make sure the boy has a girl in ans
                System.out.print(boy.name + "␣" + ans.get(boy.name) + "␣");
            }
        }
        long endTime = System.currentTimeMillis(); // Record end time
        long elapsedTime = endTime - startTime;
        System.out.println(); // Add a newline for better formatting
        System.out.println("Time␣taken:␣" + elapsedTime + "␣milliseconds");
        in.close();
    }

    static boolean prefer(Node girl, Node comBoy) {
        Node now = girl.date;
        int nowIndex = 0;
        int comIndex = 0;
        for (int i = 0; i < girl.likes.length; i++) {
            if (now.name == girl.likes[i].name)
                nowIndex = i;
            if (comBoy.name == girl.likes[i].name)
                comIndex = i;
        }
        if (nowIndex < comIndex) {
            // the origin wins
            return true;
        } else
            return false; // the new boy wins
    }

    static boolean allDated(Node[] nodes) {
        for (Node node : nodes) {
            if (node.ifFree) {
                return false;
            }
        }
        return true;
    }
}

class Node {
    String name;
    boolean ifFree;
    Node date;
    Node[] likes;

    public Node(String name, int n) {
        this.name = name;
        this.ifFree = true;
        this.likes = new Node[n];
        this.date = null;
    }

    public void date(Node lover) {
        // the ex was deleted
        if (this.date != null) {
            Node ex = this.date;
            ex.ifFree = true;
            ex.date = null;
        }
        // the new comes up
        this.date = lover;
        this.ifFree = false;
        lover.date = this;
        lover.ifFree = false;
    }
}
```

1.2 Stability Verification

- 测试结果经过手测是正确的

1.3 Efficiency Constraints

- Java for N = 1000
- 276 ms

1.4 Input Parsing

- 没有遇到过这个问题。不知道为什么

2 Task 2: Construct Test Cases

- 具体测试用例见dataTest.txt

3 Task 3: Test Your Algorithm

3.1 Manual Verification

3.2 Diff-Based Testing

- Procedure
  - Brute-Force Program
  - Comparison
- Think about and attempt to solve the following Questions:
  - Does diff-based testing remain feasible for a little larger inputs (e.g., (N 10 or N 20))? 不合适。因为从我的暴力算法来看，N = 13的时候就需要跑很长时间了
  - How to test when N exceeds a threshold (e.g., (N 10 or N 20))? 不清楚
- Performance Testing
  - Baseline Benchmark

    - 现场跑吧。很快的
  - Scalability Analysis
    - Measure execution time for (N = 100, 500, 1000, 2000) to observe growth trends.
    - Plot time vs. to confirm complexity.
  - Optimization

4 Fairness Discussion

Does the result differ if boys propose first vs. girls propose first? Explain why?

- 不会differ。侧重男生偏好和侧重女生偏好在算法上是相同的，只不过需要把输入的男生和女生的行交换一下。只不过最后得出的pair是更按照女生偏好来得出的，所以我觉得如果只侧重于是否能实现稳定匹配，是没有区别的。性能上也是没有区别的。但是考虑结果是否侧重男生女生的话，的确有所不同