Student Name: Duc Tu Luong  
Last 3-digit ID: 122

# Homework #1 Source Code

## 2.2

package gsalgorithm;

import java.util.ArrayList;

import java.util.Arrays;

import java.util.Collections;

import java.util.LinkedList;

import java.util.List;

import java.util.Map;

import java.util.TreeMap;

/\*\*

\*

\* @author LuongDucTu

\*/

public class GSAlgorithm {

public Map<Integer, Integer> finalMatches = new TreeMap<>();

private List<Integer> men = new ArrayList<>();

private List<Integer> women = new ArrayList<>();

Map<Integer, List<Integer>> menPref = new TreeMap<>();

Map<Integer, List<Integer>> womenPref = new TreeMap<>();

/\*\*

\* @param args the command line arguments

\*/

/\*\*

\* Constructor

\* @param m

\* @param w

\* @param mp

\* @param wp

\*/

public GSAlgorithm(List<Integer> m, List<Integer> w, Map<Integer, List<Integer>> mp, Map<Integer, List<Integer>> wp) {

this.men = m;

this.women = w;

this.menPref = mp;

this.womenPref = wp;

// Collections.reverse(this.men);

this.finalMatches = getMatches();

}

/\*\*

\* function to calculate matches

\*/

private Map<Integer, Integer> getMatches() {

Map<Integer, Integer> matches = new TreeMap<>();

List<Integer> freeMen = new LinkedList<>();

freeMen.addAll(men);

while(!freeMen.isEmpty()) {

/\* read from 1 to 5 \*/

int currMan = freeMen.remove(0);

List<Integer> currManPref = menPref.get(currMan);

for (int i = 0; i < currManPref.size(); i++) {

/\* get the woman from the highest-ranked position \*/

int woman = currManPref.get(i);

/\* check if this woman in the current man's list is still free \*/

if (matches.get(woman) == null) {

/\* then match these two \*/

matches.put(woman, currMan);

break;

} else {

/\* this woman is matches with other man \*/

int otherMan = matches.get(woman);

/\* get current woman preference list \*/

List<Integer> currWomanPref = womenPref.get(woman);

/\* check if this woman prefers current man to other man \*/

if (currWomanPref.indexOf(currMan) < currWomanPref.indexOf(otherMan)) {

matches.put(woman, currMan);

/\* other man now becomes free \*/

freeMen.add(otherMan);

break;

}

}

}

}

return matches;

}

/\*\*

\* main function \*

\* @param args

\*/

public static void main(String[] args) {

List<Integer> m = new ArrayList<>(Arrays.asList(1, 2, 3, 4, 5));

List<Integer> w = new ArrayList<>(Arrays.asList(1, 2, 3, 4, 5));

Map<Integer, List<Integer>> mp = new TreeMap<>();

Map<Integer, List<Integer>> wp = new TreeMap<>();

/\* init data for men preference list \*/

mp.put(1, Arrays.asList(2, 1, 4, 5, 3));

mp.put(2, Arrays.asList(4, 2, 1, 3, 5));

mp.put(3, Arrays.asList(2, 5, 3, 4, 1));

mp.put(4, Arrays.asList(1, 4, 3, 2, 5));

mp.put(5, Arrays.asList(2, 4, 1, 5, 3));

/\* init data for women preference list \*/

wp.put(1, Arrays.asList(5, 1, 2, 4, 3));

wp.put(2, Arrays.asList(3, 2, 4, 1, 5));

wp.put(3, Arrays.asList(2, 3, 4, 5, 1));

wp.put(4, Arrays.asList(1, 5, 4, 3, 2));

wp.put(5, Arrays.asList(4, 2, 5, 3, 1));

GSAlgorithm gs = new GSAlgorithm(m, w, mp, wp);

System.out.println(gs.finalMatches);

}

}

## 2.3

package gsalgorithm;

import java.util.ArrayList;

import java.util.Arrays;

import java.util.Collections;

import java.util.LinkedList;

import java.util.List;

import java.util.Map;

import java.util.TreeMap;

/\*\*

\*

\* @author LuongDucTu

\*/

public class GSAlgorithm {

public Map<Integer, Integer> finalMatches = new TreeMap<>();

private List<Integer> men = new ArrayList<>();

private List<Integer> women = new ArrayList<>();

Map<Integer, List<Integer>> menPref = new TreeMap<>();

Map<Integer, List<Integer>> womenPref = new TreeMap<>();

/\*\*

\* @param args the command line arguments

\*/

/\*\*

\* Constructor

\* @param m

\* @param w

\* @param mp

\* @param wp

\*/

public GSAlgorithm(List<Integer> m, List<Integer> w, Map<Integer, List<Integer>> mp, Map<Integer, List<Integer>> wp) {

this.men = m;

this.women = w;

this.menPref = mp;

this.womenPref = wp;

// Collections.reverse(this.men);

this.finalMatches = getMatches();

}

/\*\*

\* function to calculate matches

\*/

private Map<Integer, Integer> getMatches() {

Map<Integer, Integer> matches = new TreeMap<>();

List<Integer> freeMen = new LinkedList<>();

freeMen.addAll(men);

while(!freeMen.isEmpty()) {

/\* read from 5 to 1 \*/

int currMan = freeMen.remove(freeMen.size() - 1);

List<Integer> currManPref = menPref.get(currMan);

for (int i = 0; i < currManPref.size(); i++) {

/\* get the woman from the highest-ranked position \*/

int woman = currManPref.get(i);

/\* check if this woman in the current man's list is still free \*/

if (matches.get(woman) == null) {

/\* then match these two \*/

matches.put(woman, currMan);

break;

} else {

/\* this woman is matches with other man \*/

int otherMan = matches.get(woman);

/\* get current woman preference list \*/

List<Integer> currWomanPref = womenPref.get(woman);

/\* check if this woman prefers current man to other man \*/

if (currWomanPref.indexOf(currMan) < currWomanPref.indexOf(otherMan)) {

matches.put(woman, currMan);

/\* other man now becomes free \*/

freeMen.add(otherMan);

break;

}

}

}

}

return matches;

}

/\*\*

\* main function \*

\* @param args

\*/

public static void main(String[] args) {

List<Integer> m = new ArrayList<>(Arrays.asList(1, 2, 3, 4, 5));

List<Integer> w = new ArrayList<>(Arrays.asList(1, 2, 3, 4, 5));

Map<Integer, List<Integer>> mp = new TreeMap<>();

Map<Integer, List<Integer>> wp = new TreeMap<>();

/\* init data for men preference list \*/

mp.put(1, Arrays.asList(2, 1, 4, 5, 3));

mp.put(2, Arrays.asList(4, 2, 1, 3, 5));

mp.put(3, Arrays.asList(2, 5, 3, 4, 1));

mp.put(4, Arrays.asList(1, 4, 3, 2, 5));

mp.put(5, Arrays.asList(2, 4, 1, 5, 3));

/\* init data for women preference list \*/

wp.put(1, Arrays.asList(5, 1, 2, 4, 3));

wp.put(2, Arrays.asList(3, 2, 4, 1, 5));

wp.put(3, Arrays.asList(2, 3, 4, 5, 1));

wp.put(4, Arrays.asList(1, 5, 4, 3, 2));

wp.put(5, Arrays.asList(4, 2, 5, 3, 1));

GSAlgorithm gs = new GSAlgorithm(m, w, mp, wp);

System.out.println(gs.finalMatches);

}

}

## 2.4

package gsalgorithm;

import java.util.ArrayList;

import java.util.Arrays;

import java.util.LinkedList;

import java.util.List;

import java.util.Map;

import java.util.TreeMap;

/\*\*

\*

\* @author LuongDucTu

\*/

public class GSAlgorithm {

public Map<Integer, Integer> finalMatches = new TreeMap<>();

private List<Integer> men = new ArrayList<>();

private List<Integer> women = new ArrayList<>();

Map<Integer, List<Integer>> menPref = new TreeMap<>();

Map<Integer, List<Integer>> womenPref = new TreeMap<>();

/\*\*

\* @param args the command line arguments

\*/

/\*\*

\* Constructor

\* @param m

\* @param w

\* @param mp

\* @param wp

\*/

public GSAlgorithm(List<Integer> m, List<Integer> w, Map<Integer, List<Integer>> mp, Map<Integer, List<Integer>> wp) {

this.men = m;

this.women = w;

this.menPref = mp;

this.womenPref = wp;

this.finalMatches = getMatches();

}

/\*\*

\* function to calculate matches

\*/

private Map<Integer, Integer> getMatches() {

Map<Integer, Integer> matches = new TreeMap<>();

List<Integer> freeMen = new LinkedList<>();

freeMen.addAll(men);

while(!freeMen.isEmpty()) {

/\* read from 1 to 5 \*/

int currMan = freeMen.remove(0);

List<Integer> currManPref = menPref.get(currMan);

for (int i = 0; i < currManPref.size(); i++) {

/\* get the woman from the highest-ranked position \*/

int woman = currManPref.get(i);

/\* check if this woman in the current man's list is still free \*/

if (matches.get(woman) == null) {

/\* then match these two \*/

matches.put(woman, currMan);

break;

} else {

/\* this woman is matches with other man \*/

int otherMan = matches.get(woman);

/\* get current woman preference list \*/

List<Integer> currWomanPref = womenPref.get(woman);

/\* check if this woman prefers current man to other man \*/

if (currWomanPref.indexOf(currMan) < currWomanPref.indexOf(otherMan)) {

matches.put(woman, currMan);

/\* other man now becomes free \*/

freeMen.add(otherMan);

break;

}

}

}

}

return matches;

}

/\*\*

\* main function \*

\* @param args

\*/

public static void main(String[] args) {

List<Integer> m = new ArrayList<>(Arrays.asList(1, 2, 3, 4, 5));

List<Integer> w = new ArrayList<>(Arrays.asList(1, 2, 3, 4, 5));

Map<Integer, List<Integer>> mp = new TreeMap<>();

Map<Integer, List<Integer>> wp = new TreeMap<>();

/\* init data for men preference list \*/

mp.put(1, Arrays.asList(2, 1, 4, 5, 3));

mp.put(2, Arrays.asList(4, 2, 1, 3, 5));

mp.put(3, Arrays.asList(2, 5, 3, 4, 1));

mp.put(4, Arrays.asList(1, 4, 3, 2, 5));

mp.put(5, Arrays.asList(2, 4, 1, 5, 3));

/\* init data for women preference list \*/

wp.put(1, Arrays.asList(5, 1, 2, 4, 3));

wp.put(2, Arrays.asList(3, 2, 4, 1, 5));

wp.put(3, Arrays.asList(2, 3, 4, 5, 1));

wp.put(4, Arrays.asList(1, 5, 4, 3, 2));

wp.put(5, Arrays.asList(4, 2, 5, 3, 1));

/\* women propose \*/

GSAlgorithm gs = new GSAlgorithm(w, m, wp, mp);

System.out.println(gs.finalMatches);

}

}

## 