1. 第一章引言：一些有代表性的问题

1.1 稳定匹配

定义:

1. 给定两组

$$\{X\},\ \{Y\},\ where\ len(\{X\}) = m,\ len(\{Y\}) = n$$

2. 给定两个偏好表

$$P_1,\ size = (m,n),\ represent\ X\ to\ Y$$
$$P_2,\ size = (n,m),\ represent\ Y\ to\ X$$

3. “匹配它们”：每个x映射到y，y也是，映射可能不是一对一的

$$T_1 : X \rightarrow Y,\ T_2 : Y \rightarrow X,$$
$$where\ maps\ T_1,\ T_2\ may\ neither\ be\ 1-to-1(\text{单射})\ nor\ subjection(\text{满射})$$

4. 完美搭配定义

$$Both\ T_1\ and\ T_2\ are\ 1-to-1\ and\ \forall x_i \rightarrow y_j\ in\ T_1,\ y_j \rightarrow x_i\ in\ T_2$$
$$(matched\ pair\ by\ pair)$$

5. 不稳定对定义

$$\exists\ an\ {\color{red}unmatched}\ pair\ x_i - y_j$$
$$(where\ matched\ pair\ are\ \{x_i, y_k\},\ \{x_l, y_j\})$$
$$s.t.\ x_i\ prefer\ y_j\ to\ y_k,\ y_j\ prefer\ x_i\ to\ x_l$$

   通俗解释：两对配偶中的各自一人相爱，容易私奔，故称这一对为不稳定配对

6. 稳定匹配

$$perfect\ matching\ without\ unstable\ pairs$$

7. 稳定匹配问题

$$given\ (1.)\ and\ (2.),\ solve\ (3.)\ that\ satisify\ (6.)$$

1.2 求婚-拒绝算法（Gale-Shapley）

一个实际问题

给定 n 个女人，n 个男人，一张男性偏好表，一张女性偏好表

解决一个可能的稳定匹配

描述:

```
Initialize each person to be free
while(some man is free and hasn’t propose to every woman){
    Choose a free man {m}
    {w} = 1st woman on m’s list to whom m has not yet proposed
    if (w is free)
        assign m and w to be engage
    //w has been engaged
    else (w prefer m to original m’)
        assign m and w to be engage, m’ to be free
    else
        w rejects m
    set w being proposed
}
```

观察

1. 男人向女人求婚的顺序从高到低
2. 女人可以是**未被求婚**或**“交易”**
3. 稳定的匹配必须存在，但不一定是唯一的

声明

1.时间复杂度为O(n^2)

```
证明：在每个循环中，一个男人提议一个女人，并且至少有 n^2 个提议。
```

2.男女都匹配

```
证明：

情况1，如果一个女人被求婚，那么她最终会匹配到一个确切的男人

情况2，如果一个女人还没有被求婚，当循环结束时，至少有一个男人向偏好列表中的所有女人求婚，但没有订婚。 不可能。
```

3.没有不稳定的对

```
证明：（反证法）
```

$$Suppose\ x_i\ and\ y_j\ are\ unstable\ pair,\ x_i\ matches\ y_n,\ y_j\ matches\ x_m$$

情况1:

$$\begin{gathered} x_i\ has\ not\ proposed\ to\ y_j \\ \rightarrow x_i\ propose\ y_n\ to\ y_j \\ \rightarrow x_i - y_n\ stable\ pair \end{gathered}$$

情况2:

$$x_i\ has\ proposed\ to\ y_j$$
$$\to y_j\ reject\ x_i\ sometime$$
$$\to y_j\ trades\ up$$
$$\to y_j\ prefer\ x_m\ than\ x_i\ at\ least$$

1. 必须存在稳定匹配

   证明：（反证法）

$$Suppose\ \exists x_i,\ y_j\ are\ not\ matched\ finally.$$
$$When\ x_i\ iterate\ all\ women\ in\ preference\ table,\ x_i\ will\ find\ y_j\ and\ they\ match.$$
$$Finally,\ a\ perfect\ matching\ will\ be\ realized,\ with\ (3),\ it's\ stable.$$

实现: O(n^2)

1. 数据结构准备

```
//准备1：O(n)
//使用HashMap将一个字符串映射到一个精确的索引
Map<String, Interger> menMap = new HashMap<>();
Map<String, Interger> weomenMap = new HashMap<>();
Insert all name-index pair;

//准备2：O(n)
//对男性和女性进行编号
men: 0, ..., n-1;
women: 0, ..., n-1;
man [] men = new man[n];
woman [] women = new woman[n];
store it as a field "index";

//准备2：O(n)
    //维护队列中的自由人
    //最大容量为n
Queue<man> freeQ = new ArrayQueue(n);
freeQ.addAll(mans); //add all mans to queue

//准备4：O(n^2)
    //对于一个人m，偏好表
m.preList = new int [n];
//m.preList[i]表示第i个女人的排名
    //同样，对于一个女性w，维护一个字段preList
w.preList = new int [n];

//准备5：O(n^2)
    //对于一个人，偏好列表按等级排序
m.preListSorted = new int[n];
//m.preListSorted[i]表示第i位女性指数
    //a pos 表示他求婚的女性
int pos = 0;

//维护2个数组来表示匹配对
int [] wifesOfMen = new int[n];
```

```java
int [] husbandsOfWomen = new int[n];
```

2. 算法

```
Initial wifesOfMen, husbandsOfWomen to -1;
while(!freeQ.isEmpty()){
    man m = freeQ.poll();
    w = m.[pos++];
    if(husbandsOfWomen[w.index] == -1){
        //直接结婚
        husbandsOfWomen[w.index] = m.index;
        wifesOfMen[m.num] = w.index;
    }
    else if(w.preList[m.index] > w.preList[husbandsOfWomen[w.index]])
    {
        //离婚
        freeQ.add(men[husbandsOfWomen[w.index]]);
        wifesOfMen[husbandsOfWomen[w.index]] = -1;
        //记录丈夫和妻子的表格改变
        husbandsOfWomen[w.index] = m.index;
        wifesOfMen[m.index] = w.index;
    }
    else
    {
        freeQ.add(m); //把m放回去
    }
}
```

深入了解

1. 有效合作伙伴

$$definition: Man\ m\ is\ a\ valid\ partner\ of\ woman\ w,$$
$$if\ there\ exists\ some\ stable\ matching\ in\ which\ they\ are\ matched$$

2. 男人最优分配

$$definition: Each\ man\ recieves\ best\ valid\ partner$$

3. GS 算法产生男人最优分配，且这是一个稳定的匹配。

   因为一个男人向一个女人求婚的顺序是从高到低，他们向另一个求婚，直到不能维持稳定的匹配

   严格证明：

   > Refer to [经典算法问题——稳定匹配（Stable Matching） - 知乎 (zhihu.com)]

$$Proof:\ Suppose\ it's\ not\ man-optimal \Rightarrow some\ man\ is\ rejected\ by\ some\ valid\ partner.$$
$$1. \exists\ another\ stable\ matching\ called\ S^*,\ let\ Y\ be\ the\ first\ man\ rejected\ by\ a\ valid\ partner\ and$$
$$A\ be\ the\ first\ women\ who\ reject\ A(rejection\ can\ be\ during\ proposal\ or\ after\ being\ pairs)$$
$$2. (In\ S^*)Wlog.\ A\ matches\ Z\ finally,\ which\ means\ for\ A:\ Z > Y.$$
$$3. Wlog.\ In\ S,\ A-Y,\ B-Z\ are\ two\ pairs. (A-Y\ and\ B-Z\ are\ both\ valid\ pairs)$$
$$4. (In\ S^*)According\ the\ first\ mentioned\ above,\ when\ Y\ is\ rejected\ by\ A,$$
$$Z\ never\ recieve\ rejections\ from\ any\ valid\ partner$$
$$5. (In\ S^*)From\ 2.\ and\ 3.,\ both\ A\ and\ B\ are\ valid\ partner\ for\ Z.$$
$$6. (In\ S^*)Z\ proposing\ to\ A\ must\ happen\ before\ Y\ rejected\ by\ A$$
$$7. (In\ S^*)When\ Z\ proposed\ to\ A, if\ Z\ had\ proposed\ to\ B,$$
$$Z\ must\ be\ rejected\ sometime\ before\ his\ proposal\ to\ B,$$
$$which\ contradicts\ that\ the\ first\ rejection\ is\ A\ rejected\ Y$$
$$8. (In\ S^*)So\ Z\ hadn't\ propose\ to\ B,\ which\ indicates\ for\ Z: A > B$$
$$9.\ A\ prefer\ Z,\ Z\ prefer\ A\ in\ S^* \Rightarrow\ S^*\ is\ unstable$$

4. GS 算法产生女人最差匹配，且这是一个稳定的匹配。

   因为当且仅当不能满足稳定匹配时，一个女人换一个新男人（获得高分）。

   严格证明:

5.

$$Proof:\ Suppose\ it's\ not\ woman-pessimal \Rightarrow some\ woman\ don't\ match\ her\ worst\ choice$$
$$1. (In\ S^*\ deduced\ from\ GS)Wlog.\ A-Z\ is\ a\ pair,\ \exists Y,\ for\ A,\ Y > Z$$
$$2. (In\ S)Wlog.\ \exists\ a\ stable\ matching\ called\ S(not\ deduced\ from\ GS),$$
$$in\ which\ A-Y\ is\ a\ pair,\ while\ Z\ matches\ B.$$
$$3. Because\ Z-B\ is\ a\ valid\ pair(from\ 3.).\ Z-A\ is\ a\ valid\ pair(from\ 2.)\ and\ in\ the\ GS.$$
$$Therefore,\ for\ Z,\ A > B \Rightarrow\ A-Z\ is\ unstable.$$

Lab1：稳定匹配

稳定匹配中的细节

1. 如何使用独特的名字来识别男人或女人
2. 如何将**男女偏好列表**转换为**男性偏好列表**

   (如何将横纵坐标为男人女人编号，内容为女人的排名的二维数组转换成横纵坐标为男人编号和女人排名，内容为女人编号的数组)
3. 要求输出配对结果的时候输出名字，如何实现。
4. 如何找到未匹配的男人

测试你的程序

1. 目标：构建一些样本来测试你的程序，那么如何创建这样的样本呢?
2. 构建特殊测试用例:

通过与同行和朋友**交流**，思考问题的**被忽视的细节**，思考**特殊的io格式**

3. 构建通用/任意测试用例:

对于这个问题，我们弄清楚我们应该构建的主要部分：

```
1.男女姓名
我们可以使用没有相同字符的字符串，例如“ABCD”，那么我们可以生成 4! 个名字按顺序排列，男的加“m”，女的加“w”。
2.男女优先列表
我们知道有 n 个女人和 n 个男人，所以通过生成 [1, ..., n] 的组合，我们可以将任意一个分配给男人/女人的偏好列表。
```

4. 检查结果

   划分程序，将分为4个部分来仔细检查：

```
1. hashmaps是否将名字映射到具有准确索引的男人或女人？
2. 是否每个人在自由人 Q 中出现的次数不超过一个？
3. 是否有人在最终结果中出现了不止一次？
4、最后的结果有没有不稳定的对？
```

   与可信赖的朋友一起测试

```
1.生成多个测试样例，比如10个。
2. 和你的朋友一起测试。 将控制台输出保存到命名文件。
3. 使用一个程序（tellDifference.java）检查它们是否不同。
```

   可以在存储库中下载程序 **tellDifference.java**。