# Chapter 1 Intro: Some Representative Problem

## 1.1 Stable matching

### Definition:

1. given two sets

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

2. given two preference tables

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

3. "matching them": each x maps to a y and so do y, which the maps may not be one-to-one

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

4. perfect matching

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

5. unstable pairs

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

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. Stable matching

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

7. Stable matching problem

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

## 1.2 Propose-And-Reject Algorithm(Gale-Shapley)

### An actual probelm

given n women, n men, a preference table for men, a preference table for women

solve a possible stable matching

### description:

```
Initialize each person to be free
while(some man is free and hasn;t propose to every woman){
    Choose a 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
}
```

## Observations

1. Men propose to women in decending order of preference
2. A woman can be either **not proposed** or **"trades up"**
3. stable matchings must exists, but may not be unique

## Claims

1. With time complexity O(n^2)

   proof: In each loop a man propose a woman, and there are at least n^2 proposals.

2. All men and women get matched

   proof:

   case1, if a women has been proposed, then she will finally match 1 exact man

   case2, if a women has not been proposed, when the loop is end, at least one man has proposed to all women in the preference list, but not engaged. It's impossible.

3. No unstable pairs

   proof: (by contradiction)

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

   case 1:

   $$x_i\ has\ not\ proposed\ to\ y_j$$
   $$\rightarrow x_i\ propose\ y_n\ to\ y_j$$
   $$\rightarrow x_i - y_n\ stable\ pair$$

   case 2:

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

4. stable matching must exist

   proof: (by contradiction)

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

## Implementation: O(n^2)

1. data structure preparation

```
//preparation 1:  O(n)
//use HashMap to map a string to an exact index
Map<String, Interger> menMap = new HashMap<>();
Map<String, Interger> weomenMap = new HashMap<>();
Insert all name-index pair;

//preparation 2:  O(n)
//numbering men and women
men: 0, ..., n-1;
women: 0, ..., n-1;
man [] men = new man[n];
woman [] women = new woman[n];
store it as a field "index";

//preparation 2: O(n)
//maintain free mans in a queue
//max capacity is n
Queue<man> freeQ = new ArrayQueue(n);
freeQ.addAll(mans); //add all mans in queue

//prepareation 4: O(n^2)
//for a man m, preference table
m.preList = new int [n];
//m.preList[i] means the ith woman's rank
//Similarly, for a women w, maintain a field preList
w.preList = new int [n];

//prepareation 5: O(n^2)
//for a man, preference list sorted by rank
m.preListSorted = new int[n];
//m.preListSorted[i] means the ith rank women's index
//a pos denote the women he proposed to
int pos = 0;

//maintain 2 arrays to indicate a match pair
```

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

2. algorithm

```
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
    }
}
```

## Deeper Understanding

1. valid parnter

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

2. man-optimal assignment

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

3. GS algorithm yields man-optimal assignment, which is a stable matching.

   Because a man propose to a woman in descending order of preference, they proposed to another until it cannot maintain the stable matching

   Strict proof:

> 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 algorithm yields woman-pessimal assignment, which is a staable matching.

   Because a woman switch to a new man(get a high score) if and only if it cannot satisfy the stable matching.

   Strict proof:

$$Proof:\ Suppose\ it's\ not\ woman-pessimal \Rightarrow some\ woman\ don't\ match\ her\ worst\ choice$$

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: stable matching

## details in stable matching

1. how to use an unique name to identify a man or woman
2. how to convert a **men-women preference list** to a **men-rank preference list**

   (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. How to output the name when it is required to output the pairing result.要求输出配对结果的时候输出名字，如何实现。
4. how to find unmatched men.如何找到未匹配的男人

## Test your program

1. Objective: construct some sample to test your program, so how to create such sample?
2. construct the special testcases:

   By **communicating** to peers and friends, thinking about **neglected details** of the problem, thinking about **special io fomat**

3. construct the general/arbitrary testcases:

   for this problem, we figure out the main part we should construct:

```
1. man and women's name
    We can use a string without same character, for example "ABCD", then we can generate 4! names by arrangement in total, and add a prefix 'm' for man, 'w' for woman.
2. man and women's preerenceList
    We know there is n women and n men, so by generating combinations of [1, ..., n] and when we can assign an arbitrary one to a man/woman's preference list.
```

4. check result

   divide the program, they will be 4 parts to inspect carefully:

```
1. Does hashmaps map the name to man or woman with accurate index?
2. Does every man occur no more than one in the free-man Q?
3. Does someone occur more than onece in the final result?
4. Is there any unstable pair in the final result?
```

   test with a trusted friend

```
1. Generate several testcase, for example, 10.
2. Test with your friend. Save console output to a named file.
3. use a program(tellDifference.java) to inspect whether they are different.
```

   with a handy prgram **tellDifference.java** in the repository.