Overview

Use arrays to solve interesting problem.

Understand mathematics of marriage proposals.

- Who benefits more, the men or women?
- Who should misrepresent their feelings?

Enjoy yourself today!

Standard disclaimer.

Stable Marriage Problem

Problem: Given N men and N women, find a “suitable” matching between men and women.

- Participants have ordered preference list of members of opposite sex.
- Each man lists women in order of preference from best to worst.
- Each woman lists men in order of preference.

**Men’s Preference List**

<table>
<thead>
<tr>
<th>Man</th>
<th>0th</th>
<th>1st</th>
<th>2nd</th>
<th>3rd</th>
<th>4th</th>
</tr>
</thead>
<tbody>
<tr>
<td>Victor</td>
<td>Bertha</td>
<td>Amy</td>
<td>Diane</td>
<td>Erika</td>
<td>Clare</td>
</tr>
<tr>
<td>Wayne</td>
<td>Diane</td>
<td>Bertha</td>
<td>Amy</td>
<td>Clare</td>
<td>Erika</td>
</tr>
<tr>
<td>Xavier</td>
<td>Bertha</td>
<td>Erika</td>
<td>Clare</td>
<td>Diane</td>
<td>Amy</td>
</tr>
<tr>
<td>Yancey</td>
<td>Amy</td>
<td>Diane</td>
<td>Clare</td>
<td>Bertha</td>
<td>Erika</td>
</tr>
<tr>
<td>Zeus</td>
<td>Bertha</td>
<td>Diane</td>
<td>Amy</td>
<td>Erika</td>
<td>Clare</td>
</tr>
</tbody>
</table>

**Women’s Preference List**

<table>
<thead>
<tr>
<th>Woman</th>
<th>0th</th>
<th>1st</th>
<th>2nd</th>
<th>3rd</th>
<th>4th</th>
</tr>
</thead>
<tbody>
<tr>
<td>Amy</td>
<td>Zeus</td>
<td>Victor</td>
<td>Wayne</td>
<td>Yancey</td>
<td>Xavier</td>
</tr>
<tr>
<td>Bertha</td>
<td>Xavier</td>
<td>Wayne</td>
<td>Yancey</td>
<td>Victor</td>
<td>Zeus</td>
</tr>
<tr>
<td>Clare</td>
<td>Wayne</td>
<td>Xavier</td>
<td>Yancey</td>
<td>Zeus</td>
<td>Victor</td>
</tr>
<tr>
<td>Diane</td>
<td>Victor</td>
<td>Zeus</td>
<td>Yancey</td>
<td>Xavier</td>
<td>Wayne</td>
</tr>
<tr>
<td>Erika</td>
<td>Yancey</td>
<td>Wayne</td>
<td>Zeus</td>
<td>Xavier</td>
<td>Victor</td>
</tr>
</tbody>
</table>
Stable Marriage Problem

Problem: Given $N$ men and $N$ women, find a “suitable” matching between men and women.

- Everyone is matched monogamously (perfect matching).
  - each man gets exactly one woman
  - each woman gets exactly one man

- Stable: no incentive for some pair of participants (or coalition) to undermine assignment by joint action.
  - an unmatched pair $(m, w)$ is UNSTABLE if man $m$ would prefer woman $w$ to his wife, and $w$ would prefer $m$ to her husband
  - unstable pair could each improve by dumping spouses and eloping

STABLE MARRIAGE = perfect matching with no unstable pairs.
(Gale and Shapley, 1962)

Example

<table>
<thead>
<tr>
<th>Men’s Preference List</th>
<th>Women’s Preference List</th>
</tr>
</thead>
<tbody>
<tr>
<td>Woman</td>
<td>0th</td>
</tr>
<tr>
<td>Xavier</td>
<td>A</td>
</tr>
<tr>
<td>Yancey</td>
<td>B</td>
</tr>
<tr>
<td>Zeus</td>
<td>A</td>
</tr>
</tbody>
</table>

Lavender assignment is a perfect matching.
Is it stable?

<table>
<thead>
<tr>
<th>Men’s Preference List</th>
<th>Women’s Preference List</th>
</tr>
</thead>
<tbody>
<tr>
<td>Woman</td>
<td>0th</td>
</tr>
<tr>
<td>Amy</td>
<td>Y</td>
</tr>
<tr>
<td>Bertha</td>
<td>X</td>
</tr>
<tr>
<td>Clare</td>
<td>X</td>
</tr>
</tbody>
</table>

Green assignment is a stable matching.

<table>
<thead>
<tr>
<th>Men’s Preference List</th>
<th>Women’s Preference List</th>
</tr>
</thead>
<tbody>
<tr>
<td>Woman</td>
<td>0th</td>
</tr>
<tr>
<td>Xavier</td>
<td>A</td>
</tr>
<tr>
<td>Yancey</td>
<td>B</td>
</tr>
<tr>
<td>Zeus</td>
<td>A</td>
</tr>
</tbody>
</table>

Orange assignment is also a stable matching.
Stable Roommate Problem

Not obvious that stable marriage exists.

Consider related “stable roommate problem.”
- 2N people.
- Each person ranks others from 0 to 2N-2.
- Assign roommate pairs so that no unstable pairs.

Preference List

<table>
<thead>
<tr>
<th>0th</th>
<th>1st</th>
<th>2nd</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adam</td>
<td>B</td>
<td>C</td>
</tr>
<tr>
<td>Bob</td>
<td>C</td>
<td>A</td>
</tr>
<tr>
<td>Chris</td>
<td>A</td>
<td>B</td>
</tr>
<tr>
<td>Doofus</td>
<td>A</td>
<td>B</td>
</tr>
</tbody>
</table>

No perfect matching is stable.

For all 3 possible perfect marriage, can always find unstable pair.
E.g., A-C forms unstable pair in lavender marriage.

Existence

Surprising Fact:
- Unlike for stable roommate problem, one (or more) stable marriages exist for any input to problem.

How do we find one?
- Are there others?
- Which one is best for Zeus?
- Is there one that is best for all the men collectively? All the women?

Propose-And-Reject Algorithm

Formal (and intuitive) method that guarantees to find a stable marriage.

Repeat until no unmatched men
- An unmatched man m proposed to his favorite woman w to whom he has not already proposed.
- If w is unmatched, she accepts proposal from m (but can later dump him).
- Otherwise, if w prefers her current fiancé to m, she reject m outright.
- Otherwise, if she prefers m to her current fiancé, she dumps her fiancé and accepts the proposal from m.

Why Does Algorithm Work?

Observation 1. Men propose to their favorite women first.

Observation 2. Once a woman is matched, she never becomes unmatched. She only “trades up.”

Fact 1. All men and women get matched.
- Suppose upon termination Zeus is not matched.
- Then some woman, say Amy, is not matched upon termination.
- By Observation 2, Amy was never proposed to.
- But, Zeus proposes to everyone, since he ends up unmatched. (contradiction)
Why Does Algorithm Work?

Observation 1. Men propose to their favorite women first.

Observation 2. Once a woman is matched, she never becomes unmatched. She only “trades up.”

Fact 2. No unstable pairs.

- Suppose Zeus-Amy is an unstable pair, i.e., each prefers each other to spouse. (Zeus-Bertha, Yancy-Amy)

- Case 1. Zeus never proposed to Amy.
  - Zeus must prefer Bertha to Amy (Observation 1)
  - Zeus-Amy is stable. (contradiction)

- Case 2. Zeus proposed to Amy.
  - Amy rejected Zeus (right away or later)
  - Amy prefers Yancy to Zeus (women only trade up)
  - Zeus-Amy is stable (contradiction)

Pseudocode

```c
int marriages = 0;
while (marriages < N)
    find unmatched man m
    while (m unmatched)
        let w be man m’s favorite women to whom he has not yet proposed
        if (w unmatched)
            m and w get engaged
            marriages++;break;
        if (w prefers m to current fiancé f)
            f now unmatched
            m and w get engaged
            break;
        else w rejects m
```

How to Represent Men and Women

Represent men and women as integers between 0 and N-1.

- 0 through N-1 since C array indices start at 0.
- Could use struct if we want to carry around more information, e.g., name, age, astrological sign.

How to Represent Marriages

Use array to keep track of marriages.

```c
wife[m] = \begin{cases} 
  w & \text{if man } m \text{ matched to woman } w \\
  -1 & \text{if man } m \text{ unmatched}
\end{cases}

husb[w] = \begin{cases} 
  m & \text{if man } m \text{ matched to woman } w \\
  -1 & \text{if woman } w \text{ unmatched}
\end{cases}
```

```c
int wife[N];
int husb[N];

for (m = 0; m < N; m++)
    wife[m] = -1;
for (w = 0; w < N; w++)
    husb[w] = -1;
```
Filling in Some of the Code

```
while (marriages < N)
    for (m = 0; wife[m] != -1; m++)
        while (wife[m] == -1)
            let w be man m’s favorite women to whom he has not yet proposed
            if (husb[w] == -1)
                husb[m] = w; wife[m] = w;
                marriages++;
                break;
            if (w prefers m to current fiancé f)
                f = husb[w];
                wife[f] = -1;
                husb[m] = w; wife[w] = m;
                break;
```

Find unmatched man

```
while (m unmatched)
    if (w unmatched)
        m and w get engaged
```

Look for current fiancé f

```
f = current fiancé of w
f now unmatched
m and w get engaged
```

Representing the Preference Lists

Use 2D-array to represent preference lists.
- 2D-array is array of arrays.
- mp[m][i] = w if man m’s i th favorite woman is w.
- wp[w][i] = m if woman w’s i th favorite man is m.

### Men’s Preference List

<table>
<thead>
<tr>
<th>Man</th>
<th>0th</th>
<th>1st</th>
<th>2nd</th>
<th>3rd</th>
<th>4th</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>3</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>1</td>
<td>0</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>4</td>
<td>2</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>3</td>
<td>0</td>
<td>4</td>
<td>2</td>
</tr>
</tbody>
</table>

### Women’s Preference List

<table>
<thead>
<tr>
<th>Woman</th>
<th>0th</th>
<th>1st</th>
<th>2nd</th>
<th>3rd</th>
<th>4th</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>4</td>
<td>0</td>
<td>1</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>1</td>
<td>4</td>
<td>2</td>
<td>0</td>
</tr>
</tbody>
</table>

### Dumping

```
int mp[N][N];
int wp[N][N];
```

Search preference list sequentially until m1 or m2 found

```
for (i = 0; i < N; i++) {
    if (wp[w][i] == m1) YES
    if (wp[w][i] == m2) NO
}
```

TOO SLOW if N is large, since need to repeat many times.

### Initializing the Preference Lists

Could read from stdin.
We’ll assign random lists for each man and woman.
- Use randomPermutation function from last time.
- Need N random permutations for men, and N for women.

```
int mp[N][N];
int wp[N][N];
for (m = 0; m < N; m++)
    randomPermutation(mp[m]);
for (w = 0; w < N; w++)
    randomPermutation(wp[w]);
```

### Men’s Preference List

- mp[1][0] = 3
  man 1 likes woman 3 the best

### Women’s Preference List

- mp[1][0] = 3
  woman 1 prefers man 3 to man 4.
Keeping Track of Men’s Proposals

Unmatched man proposes to most favorable woman to whom he hasn’t already proposed.

How do we keep track of which woman a man has proposed to?

- Men propose in decreasing order of preference.
- Suffices to keep track of number of proposals in array.

propose[m] = i if man m has proposed to i woman already.

```c
int props[N];
for (i = 0; i < N; i++)
    props[i] = 0;
```

```c
for (;;) {
    w = mp[m][props[m]];
    props[m]++;
    ...;
}
```

props[m] is next woman on preference list
make next proposal to woman mp[m][props[m]]

Try Out The Code

```c
#include <stdio.h>
#include <stdlib.h>
#include <assert.h>
#define N 500

int main(void) {
    int mp[N][N]; /* mp[m][i] = w if man m’s ith favorite woman is w */
    int wp[N][N]; /* wp[w][i] = m if woman w’s ith favorite man is m */
    int wife[N];  /* wife[m] = w if m married to w */
    int husb[N];  /* husb[w] = m if m married to w */
    int props[N]; /* props[m] = i if man m has proposed to i women */
    int marriages = 0; /* number of couples matched so far */
    int m, w;

    /* initialize men */
    for (m = 0; m < N; m++) {
        wife[m] = -1;
        randomPermutation(mp[m], N);
    }

    /* initialize women */
    for (w = 0; w < N; w++) {
        husb[w] = -1;
        randomPermutation(wp[w], N);
    }

    marriage.c
    while (marriages < N) {
        /* find first unmatched man */
        for (m = 0; m < N; m++)
            if (wife[m] == -1) break;
        printf("man %d proposing:
", m);
        /* propose to next women on list until successful */
        for (;;) {
            w = mp[m][props[m]];
            props[m]++;
            /* woman w unmatched */
            if (husb[w] == -1) {
                print("accepted\n", w, wife[w]);
                husb[w] = m;
                wife[m] = w;
                marriages++;
                break;
            }
            /* woman w prefers m to current mate */
            if (wr[w][m] < wr[w][husb[w]]) {
                print("rejected\n");
                husb[w] = -1;
                husb[w] = m;
            }
            /* otherwise m rejected by w */
            if (wr[m][w] < wr[w][husb[w]]) {
                print("rejected\n");
                husb[w] = -1;
            }
        }
    }

    marriage.c
    printf("Stable matching\n");
    for (m = 0; m < N; m++)
        printf("%5d %5d\n", m, wife[m]);
    return 0;
}
```

Unix

```c
% gcc marriage.c
% a.out
```

```
man 0 proposing: to woman 4 accepted (woman 4 previously unmatched)
man 1 proposing: to woman 0 accepted (woman 0 previously unmatched)
man 2 proposing: to woman 2 accepted (woman 2 previously unmatched)
man 3 proposing: to woman 3 accepted (woman 3 previously unmatched)
man 4 proposing: to woman 2 accepted (woman 2 previously unmatched)
man 2 proposing: to woman 3 accepted (woman 3 previously unmatched)
man 3 proposing: to woman 0 rejected (woman 2 prefers 0)
man 4 proposing: to woman 1 rejected (woman 4 prefers 0)
Stable matching
0 4
1 0
2 3
3 1
4 2
```

Try Out The Code

Observation: code is REALLY slow for large N.
**An Auxiliary Data Structure**

Create a 2D array that stores men's ranking of women.

- \( mr[m][w] = i \) if man \( m \)'s ranking of woman \( w \) is \( i \).
- \( wr[w][m] = i \) if woman \( w \)'s ranking of man \( m \) is \( i \).

<table>
<thead>
<tr>
<th>Men's Preference List</th>
<th>Men's Rankings</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Man</strong></td>
<td><strong>Men</strong></td>
</tr>
<tr>
<td>0th</td>
<td>0</td>
</tr>
<tr>
<td>1st</td>
<td>1</td>
</tr>
<tr>
<td>2nd</td>
<td>2</td>
</tr>
<tr>
<td>3rd</td>
<td>3</td>
</tr>
<tr>
<td>4th</td>
<td>4</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Man</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>3</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>1</td>
<td>0</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>4</td>
<td>2</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>3</td>
<td>0</td>
<td>4</td>
<td>2</td>
</tr>
</tbody>
</table>

- \( mp[1][0] = 3 \)
  - man 1 likes woman 3 best
- \( mr[1][3] = 0 \)
  - man 1 likes woman 3 best

**Check if Marriage is Stable**

Check if \( \text{husb}[N] \) and \( \text{wife}[N] \) correspond to a stable marriage.

- Good warmup and useful for debugging.
- Check every man-woman pair to see if they're unstable.
- Use ranking arrays.

```c
int isStable(int husb[], int wife[], int mr[N][N], int wr[N][N]) {
    int m, w;
    for (m = 0; m < N; m++)
        for (w = 0; w < N; w++)
            if (mr[m][w] < mr[m][wife[m]]) &&
               (wr[w][m] < wr[w][husb[w]])
                return 0;
    return 1;
}
```

**Check if Marriage is Stable**

Check if \( \text{husb}[N] \) and \( \text{wife}[N] \) correspond to a stable marriage.

- Good warmup and useful for debugging.
- Check every man-woman pair to see if they're unstable.
- Use ranking arrays.

Time/space tradeoff for using auxiliary ranking arrays.

- Disadvantage: requires twice as much memory (storage).
- Advantage: dramatic speedup in running time (using 400 MHz Pentium II with \( N = 10,000 \)).
Men vs. Women

Given input, there may be several stable marriages. Which one does algorithm find?

Fact 3. Propose-and-reject algorithm is MAN-OPTIMAL:
  - Simultaneously best for each and every man.
  - There is no stable marriage in which any single man individually does better.

Fact 4. Propose-and-reject algorithm is WOMAN-PESSIMAL:
  - Simultaneously worst for each and every woman.
  - There is no stable marriage in which any single woman individually does worse.

Fact 5. The man-optimal stable matching is weakly Pareto optimal.
  - In every other matching (stable or unstable), at least one man does strictly worse.

Extensions

Yeah, but in real-world every woman is not willing to marry every man, and vice versa?
  - Some participants declare others as “unacceptable” (prefer to be alone than with given partner).
  - Algorithm extends to handle partial preference lists.

Also, there may be an unequal number of men and women.
  - E.g., 150 men, 100 women.
  - Algorithm extends.

What about limited polygamy?
  - E.g., Bill wants 3 women.
  - Algorithm extends.

Application

Matching medical school residents to hospitals. (NRMP)
  - Hospitals ~ Men (limited polygamy allowed).
  - Residents ~ Women.
  - Original use just after WWII (predates computer usage).
  - Ides of March, 13,000+ residents.

Rural hospital dilemma.
  - Certain hospitals (mainly in rural areas) were unpopular and declared unacceptable by many residents.
  - Rural hospitals were under-subscribed in NRMP matching.
  - How can we find stable matching that benefits “rural hospitals”?

Rural Hospital Theorem:

Deceit: Machiavelli Meets Gale-Shapley

Is there any incentive for a participant to misrepresent his/her preferences?
  - Assume you know men’s propose-and-reject algorithm will be run.
  - Assume that you know the preference lists of all other participants.

Fact 6.
Deceit: Machiavelli Meets Gale-Shapley

Is there any incentive for a participant to misrepresent his/her preferences?

Fact 7.

<table>
<thead>
<tr>
<th>Men's Preferences</th>
<th>Women's Preferences</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Amy</strong></td>
<td>Y X Z</td>
</tr>
<tr>
<td><strong>Bertha</strong></td>
<td>X Y Z</td>
</tr>
<tr>
<td><strong>Clare</strong></td>
<td>X Y Z</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Men</th>
<th>Propose early and often.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Women</td>
<td>Ask out the guys.</td>
</tr>
</tbody>
</table>

CS can be socially enriching and fun!

Engineers get the best partners!!