1 RESIDENCY MATCHING

1.1 The Medical Track

Timetable for physicians:

- Childhood (13 years).
- High School (4 years).
- College (4 years).
- Medical School (4 years).
- Residency (3-8 years).
- Fellowship (optional 2-8 years).

Each phase change has entry requirements and an application procedure. The medical school to residency transition is the one we’re focusing on.

1.2 Residency is Special

In college and grad school, programs are typically large enough that it is possible to accept more students than are actually expected to attend and it is satisfactory if slightly fewer or slightly more than the expected number accept. Residency programs can have as few as one or two slots per year and rarely more than 20. Can’t leave things to chance! Hospitals wise to recruit students early (perhaps even years before they graduate), to ensure a strong and full program.

1.3 The Match

About 40 years ago, “The Match” was developed to rationally assign prospective residents (students) to residency programs (hospitals). Students submit a rank-order list (ROL) to the National Residency Match Program (NRMP). Hospitals rank-order the students, too.
NRMP finds a match and publishes its results in mid March.
Legally binding!
Used since 1952 based on a 1962 (!) Gale and Shapley paper.

2 THE STABLE MARRIAGE PROBLEM

2.1 Simplified Problem

It turns out that there are some special complications in the Match (couples match, program parity restrictions) that make it difficult to analyze. We’ll look at an idealization of this problem.

We’ve got $n$ boys and $n$ girls. Each boy submits a ROL of all $n$ girls. Ditto for the girls. (How much data do we have?)

A marriage (or matching) is a bijection between girls and boys.

2.2 Example

Boys preferences:

<table>
<thead>
<tr>
<th>Albert</th>
<th>Laura</th>
<th>Nancy</th>
<th>Kira</th>
<th>Marcy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Brad</td>
<td>Marcy</td>
<td>Kira</td>
<td>Nancy</td>
<td>Laura</td>
</tr>
<tr>
<td>Chuck</td>
<td>Laura</td>
<td>Marcy</td>
<td>Kira</td>
<td>Nancy</td>
</tr>
<tr>
<td>Daniel</td>
<td>Nancy</td>
<td>Kira</td>
<td>Marcy</td>
<td>Laura</td>
</tr>
</tbody>
</table>

Girls preferences:

<table>
<thead>
<tr>
<th>Kira</th>
<th>Brad</th>
<th>Albert</th>
<th>Daniel</th>
<th>Chuck</th>
</tr>
</thead>
<tbody>
<tr>
<td>Laura</td>
<td>Daniel</td>
<td>Chuck</td>
<td>Albert</td>
<td>Brad</td>
</tr>
<tr>
<td>Marcy</td>
<td>Albert</td>
<td>Daniel</td>
<td>Chuck</td>
<td>Brad</td>
</tr>
<tr>
<td>Nancy</td>
<td>Brad</td>
<td>Albert</td>
<td>Daniel</td>
<td>Chuck</td>
</tr>
</tbody>
</table>

A marriage: (A, L), (B, K), (C, M), (D, N).

2.3 Stability

Not a great coupling. Why?

A stable marriage is a marriage in which no boy-girl pair would want to swap spouses.

Such a pair is unstable because they’d rather dump their partners and marry each other than stay with who they’ve got.

**Theorem 1**: If there is no unstable pair, then there is also no unstable coalition. Why? A coalition is really just a set of pairs.

Can you find a stable marriage for this group?

2.4 Stability Checking

Here’s an obvious algorithm for checking if a given marriage is stable.

**CHECK-MARRIAGE**($R$, $M$)
for each \( b \in \{1, n\} \)
1. do for each \( g \in \{1, n\} \)
2. \( bm \leftarrow \text{RANK}(R_B[b], M_B[b]) \)
3. \( bg \leftarrow \text{RANK}(R_B[b], g) \)
4. \( gm \leftarrow \text{RANK}(R_G[g], M_G[g]) \)
5. \( gb \leftarrow \text{RANK}(R_G[g], b) \)
6. if \((bg < bm)\) and \((gb < gm)\)
7. then return \((b, g)\)
8. return True

Here, \( R_B[b] \) is the ROL for boy \( b \), \( R_G[g] \) is the ROL for girl \( g \), \( M_B[b] \) is the mate of boy \( b \) in matching \( M \), and \( M_G[g] \) is the mate of girl \( g \) in matching \( M \).

\( M_G[M_B[b]] = ? \).

### 2.5 Rank Computation

\( \text{RANK}(L, p) \) is the rank position of person \( p \) in ROL \( L \).

\( \text{RANK}(L, p) \)
1. for each \( j \in \{1, n\} \)
2. do if \( L[j] = p \)
3. then return \( j \)
4. return Bug!

What’s the total algorithm running time?

\( \Theta(n^3) \).

Not hard to do better than this (HW).

### 3 THE MATCHING ALGORITHM

#### 3.1 Fundamental Result

**Theorem 2:** There is always a stable marriage, for any well-formed set of ROLs. To prove this, we describe an algorithm for finding such a marriage.

#### 3.2 Matching Algorithm

\( \text{MATCH}(R) \)
1. for each \( i \in \{1, n\} \)
2. do \( M_B[i] \leftarrow \text{free} \)
3. rejected[i] \( \leftarrow 0 \)
4. \( M_G[i] \leftarrow \text{free} \)
5. while there exists some \( b \) who is a free man
6 do
7 g ← RB[b][rejected[b] + 1]
8 if MG[g] = free
9 then MB[b] ← g
10 MG[g] ← b
11 else t' ← MG[g]
12 if rank(RG[g], b) < rank(RG[g], t')
13 then rejected[t'] ← rejected[t'] + 1
14 MB[t'] ← free
15 MB[b] ← g
16 MG[g] ← b
17 else rejected[b] ← rejected[b] + 1
18 return M

3.3 In Words

The boys propose (to their top-ranked girl that hasn’t yet rejected them) and the girls tentatively accept. This is an engagement.

If a girl ever gets an offer from a boy that she likes better than her current fiance, she breaks her old engagement for him.

Stop when all boys are engaged.

M holds the tentative matching. rejected[b] records the rank position of b’s most recent rejection (ouch).

Once b is rejected by g, b never needs to propose to g again. Why?

She is already holding a proposal from someone better.

(It is possible to also have the girls do the proposing... we will return to this later.)

Example!

3.4 More Theorems

We will soon prove a number of important facts about this algorithm.

- **Theorem 3:** For every girl g, \text{RANK}(RG[g], MG[g]) improves (decreases) throughout the execution of MATCH.

- **Theorem 4:** The order in which the boys are chosen to propose doesn’t matter (you get the same marriage no matter what).

- **Theorem 5:** The algorithm will terminate before the ROL of any boy is exhausted.

- **Theorem 6:** The returned matching is stable.

- **Theorem 7:** The boys get to marry the best girl that they could possibly get in any stable marriage. The matching is simultaneously optimal for all of them!
3.5 Time Analysis

But first, how long does it take for MATCH to run? What happens in the while loop? Either a free girl accepts her first proposal or a boy gets rejected. What’s an upper bound for each of these? So, that tells us how many times the loop is executed. How much work do we do per iteration? We need to find a free man and we execute RANK several times. These can both be done in $O(n)$. This gives us a running time of $O(n^3)$.

3.6 Improving the Running Time

An easy change to the algorithm improves the running time to $O(n^2)$.

- How can we make the search for a free man (boy) take constant time?
- How can we implement RANK to run in constant time? It needs to, for example, take a boy $b$ and a girl $g$ and tell us where $b$ is on $g$’s ROL. With some preprocessing, can do this with a single array lookup.

Therefore, each iteration of the while loop takes constant time. Total $O(n^2)$ time.

4 UNDERSTANDING THE ALGORITHM

4.1 Termination

As simple as the algorithm is, its behavior is not immediately obvious. Let’s show that no boy can be rejected by all girls.

- A boy $b$ is only rejected by a girl who is already engaged (not to him).
- So, the last rejection for $b$ can only come if all $n$ girls are already engaged to someone better.
- But there can only be $n - 1$ boys better than $b$, so this can’t happen.

This proves Theorem 5: the algorithm terminates before the ROL of any boy is exhausted. We get a marriage!

4.2 Correctness

Furthermore, the final marriage is stable (Theorem 6). Proof by contradiction.

- Assume that there is some pair $(b, g)$ that prefer each other to whomever they are matched to.
• Since $b$ proposes strictly in order of his ROL, this means that he must have proposed to $g$ at some point.

• Since $b$ isn’t engaged to $g$, that means $g$ must have rejected $b$.

• But, $g$ would only reject $b$ for someone she ranks more highly.

• Therefore, by Theorem 3, $g$’s final fiancé is ranked more highly than $b$.

• Therefore $g$ doesn’t prefer $b$ to her mate. Contradiction!

Since the algorithm always terminates (Theorem 5), and the returned matching is stable (Theorem 6), then there is always a stable marriage, for any well-formed set of ROLs (Theorem 2). Note that this is a proof by algorithmic correctness!

4.3 Boy Optimality

We say that $(b, g)$ is a stable pair if there is some stable marriage in which $b$ and $g$ are matched. We say that $g$ is a stable partner of $b$ and vice versa. The marriage returned by MATCH assigns every boy to his favorite stable partner! We, therefore, call this the boy-optimal match.

• Claim: At no point in the execution of MATCH does a girl reject a stable partner. This means that the first time a boy proposes to one of his stable partners, he gets to stay with her (and consequently marries his favorite stable partner).

• Let’s assume that a stable pair is rejected. Let $(b_1, g_1)$ be the first such stable pair. Let $b_2$ be the boy that causes the breakup of $b_1$ and $g_1$.

• Since $(b_1, g_1)$ is a stable pair, there is some stable marriage that includes it. In one such stable marriage, let $g_2$ be $b_2$’s mate.

• But, a marriage that includes $(b_1, g_1)$ and $(b_2, g_2)$ can’t be stable! Because $g_1$ rejected $b_1$ for $b_2$, we know $g_1$ would rather be with $b_2$. Because the $(b_1, g_1)$ breakup was the first, we know that $b_2$ prefers $g_1$ to any other stable partner (such as the hypothetical $g_2$).

• So, $(b_1, g_1)$ isn’t a stable pair—a contradiction—and, therefore, no stable pair can be rejected.

This proves Theorem 7 (boy optimality), but it also implies Theorem 4: you get the same marriage independent of the order in which free boys propose.
4.4 Girl Pessimality

Consider the pairing obtained by taking each girl and assigning her to her least favorite stable partner.

**Claim:** The girl-pessimal pairing is a stable marriage, and it is the same as the marriage returned by MATCH.

This can be shown by the same type of argument as above or by appealing to the boy-optimality property of MATCH in the right way. Now, if we reverse the roles of boys and girls (girls propose and boys tentatively accept), then we get a girl-optimal and boy-pessimal marriage.

5 BACK TO THE HOSPITAL

5.1 Matching Students to Hospitals

The problem of matching medical students to hospitals is a bit different from what we’ve discussed.
A single hospital usually takes on more than one student.
This can be handled by actually treating a hospital as a group of people (one per fillable slot).
The same basic mathematical results hold in this variation (the MATCH algorithm always gives a unique, stable, boy-optimal marriage).

5.2 Matching Algorithm

So, who gets to “play the boy” in this setting?

- Hospital-proposing algorithm: Hospital optimal, student pessimal. No incentive for the hospitals to cheat.
- Student-proposing algorithm: Student optimal.

From 1952 to 1997, a hospital-proposing (and therefore student pessimal) algorithm has been used.

5.3 Controversy!

The written information concerning the Match has confused some of the medical students over the years:

- “...each applicant is matched to the hospital program highest on the applicant’s rank order list that has offered the applicant a position...” (1995 NRMP Handbook)

Is this correct?

Yes, this is not the same as saying that the applicant gets the highest possible hospital.
Someone, at some point, decided that this wasn’t fair (both the description and the process). Some quotes from AMSA, October 1995.

- “Don’t let the NRMP alone plan your future. The American Medical Student Association, as your advocate and voice, is demanding that the scales be tipped in YOUR favor to give the power of choice back to the MEDICAL STUDENT.” (October, 1995).

- “The NRMP and its algorithm can be thought of as the rules to a game with the hospital and the students acting as players (a game with important consequences, as the outcome greatly affects careers and lives).”

5.4 The Study

The students got enough people worked up that NRMP had to respond.

- NRMP had economist Alan Roth do a study examining the effect of changing the algorithm from basically hospital-optimal to basically student-optimal. (Why is it not absolutely optimal?)

  Because of complications like couples matches.

- He used 5 years of actual ROLs and tested a large number of variations of the matching algorithm.

- He found that 91 out of 111,026 students would be affected by a shift from roughly hospital-optimal to student-optimal (and only 65 of them actually improved!). That’s about .06%.

5.5 The Decision

From a NRMP Press Release:

- Washington, D.C., May 8, 1997—Beginning January 1, 1998, the National Resident Matching Program (NRMP) will change the algorithm used to match students and physicians with residency programs. Although the new mathematical formula will effect less than 15 of the over 22,000 individuals who participate in the Match annually, the change refines a highly respected process that has served residency training for over 45 years.

- ...

- Now, following 15 months of intense study, the NRMP has decided to change the algorithm so that in a very small number of instances, a few applicants will be matched to a more preferred position than they would have received under the present process. “Essentially, the new algorithm improves the chances of one applicant in a thousand,”
said NRMP Executive Director Robert L. Beran, Ph.D. “After careful examination, the NRMP board supported this change in recognition of the major impact the Match can have on the career plans of every applicant.”

• ...

6 OTHER NEAT FACTS

• Can be more students than hospital slots, or vice versa. Theory carries over (smaller set gets completely matched).

• The same students are matched and the same hospital positions are filled at every stable matching. (That is, any student who is unmatched at one stable matching is unmatched at every stable matching, and the hospital positions that are unfilled are the same at every stable matching.) Only the specific assignment of which matched students are in which filled hospital positions differs between different stable matchings.

• When the student-proposing algorithm is used, but not when the hospital-proposing algorithm is used, no student can possibly improve his or her match by submitting an ROL that is different from his or her true preferences.

• There is a polynomial-time egalitarian algorithm (neither student nor hospital optimal).

• The student-optimal algorithm is, in fact, hospital pessimal (even though hospitals don’t directly give a ranking over groups of students).

• Couples match: The basic algorithm can be extended to permit constraints on where pairs of people end up. However, the nice mathematics is trashed.