Climb an infinite ladder?

Quick Poll.

Sad Islanders...

Common Knowledge.

They know induction.

Strong Induction and Recursion.
Tidying up induction.

The induction principle works on the natural numbers.

Proves statements of form: \( \forall n \in \mathbb{N}, P(n) \).

Yes.

What if the statement is only for \( n \geq 3 \)?

Restate as: \( \forall n \in \mathbb{N}, Q(n) \) where \( Q(n) = (n \geq 3) \implies P(n) \).

Base Case: typically start at 3.

Since \( \forall n \in \mathbb{N}, Q(n) \implies Q(n+1) \) is trivially true before 3.

Can you do induction over other things? Yes.

Any set where any subset of the set has a smallest element.

In some sense, the natural numbers.

Stable Matching Problem

- \( n \) candidates and \( n \) jobs.
- Each job has a ranked preference list of candidates.
- Each candidate has a ranked preference list of jobs.

How should they be matched?

- Maximize total satisfaction.
- Maximize number of first choices.
- Maximize worse off.
- Minimize difference between preference ranks.

A stable matching??

Given a set of preferences.

Is there a stable matching?

How does one find it?

Consider a single type version: stable roommates.

Example: A matching

\[
S = \{ \{ \text{CalBears, Pac-12}\}, \{ \text{WakeForest, ACC}\} \}.
\]

Definition: A rogue couple \( b, g^* \) for a pairing \( S \):

\( b \) and \( g^* \) prefer each other to their partners in \( S \).

Example: Cal Bears and the ACC are a rogue couple in \( S \).

Not a great example of stable matching, but interesting exercise in “selfish” incentives.

The best laid plans..

Consider the pairs...

- Cal Bears and the Pac-12
- Wake Forest and the ACC

Cal Bears prefers the ACC

The ACC prefers Cal Bears.

Uh..oh. Sad Pac-12, (and Wake Forest.)

The Propose and Reject Algorithm.

Each Day:

1. Each job proposes to its favorite candidate on its list.
2. Each candidate rejects all but their favorite proposer (whom they put on a string.)
3. Rejected job crosses rejecting candidate off its list.

Stop when each job gets exactly one proposal.
The Propose and Reject Algorithm.

Each Day:
1. Each job proposes to its favorite candidate on its list.
2. Each candidate rejects all but their favorite proposer (whom they put on a string.)
3. Rejected job crosses rejecting candidate off its list.

Stop when each job gets exactly one proposal.

Does this terminate?
...produce a matching?
....a stable matching?
Do jobs or candidates do “better”?

Improvement Lemma
Improvement Lemma: It just gets better for candidates.
If on day t a candidate g has a job b on a string, any job, b′, on g's string for any day t′ > t is at least as good as b.

Proof:
P(0)– true. Candidate has b on string.
Assume P(k). Let b′ be job on string on day t + k.
On day t + k + 1, job b′ comes back.
Candidate g can choose b′, or do better with another job, b′′.
That is, b′ ≥ b by induction hypothesis.
And b′ is better than b′ by algorithm.
⇒ Candidate does at least as well as with b.
P(k) ⇒ P(k + 1).
And by principle of induction, lemma holds for every day after t.

Termination.
Every non-terminated day a job crossed an item off the list.
Total size of lists? n jobs, n length list. n^2
Terminates in ≤ n^2 steps!
Matching when done.

**Lemma:** Every job is matched at end.

**Proof:**
If not, a job $b$ must have been rejected $n$ times.
Every candidate has been proposed to by $b$, and Improvement lemma.
$\implies$ each candidate has a job on a string.
and each job is on at most one string.
$n$ candidates and $n$ jobs. Same number of each.
$\implies b$ must be on some candidate's string!
Contradiction.

---

Poll: The argument for termination ...

(A) Implies: no unmatched job at end.
(B) Uses Improvement Lemma: every candidate matched.
(C) From Algorithm: unmatched job would ask everyone.
(D) Implies: every one gets their favorite job.

---

Matching is Stable.

**Lemma:** There is no rogue couple for the matching formed by traditional marriage algorithm.

**Proof:**
Assume there is a rogue couple; $(b, g^*$).
Job $b$ proposes to $g^*$ before proposing to $g$.
So $g^*$ rejected $b$ (since he moved on)
By improvement lemma, $g^*$ prefers $b^*$ to $b$.
Contradiction!

---

Question: Proof of Job Propose and Reject a stable pairing uses?

(A) Contradiction.
(B) Uses the improvement lemma.
(C) Induction.
(D) Direct.
(E) The algorithm description.

---

Good for jobs? candidates?

Is the Job-Proposes better for jobs? for candidates?

**Definition:** A matching is $x$-optimal if $x$’s partner
is its best partner in any stable pairing.

**Definition:** A matching is $x$-pessimal if $x$’s partner
is its worst partner in any stable pairing.

**Definition:** A matching is job optimal if it is $x$-optimal for all jobs $x$.
and so on for job pessimal, candidate optimal, candidate pessimal.

**Claim:** The optimal partner for a job must be first in its preference list.

As well as you can be in a globally stable solution!

Question: Is there a job or candidate optimal matching?

Is it possible:
- $b$-optimal pairing different from the $b^*$-optimal matching!

Yes? No?

---

Understanding Optimality: by example.

A: 1,2 1: A,B
B: 1,2 2: B,A
Consider pairing: (A,1), (B,2).
Stable? Yes.
Optimal for $B$?
Notice: only one stable pairing. If (A,2) are pair, (A,1) is rogue couple.
So this is the best $B$ can do in a stable pairing.
So optimal for $B$.
Also optimal for $A$, $1$ and $2$. Also pessimal for $A$, $B$, $1$ and $2$.
A: 1,2 1: A,B
B: 2,1 2: A,B
Pairing S: (A,1), (B,2) Stable? Yes.
Pairing T: (A,2), (B,1) Also Stable.
Which is optimal for $A$? $S$ Which is optimal for $B$? $S$
Which is optimal for $1$? $T$ Which is optimal for $2$? $T$
Pessimality?
Job Propose and Candidate Reject is optimal!

For jobs? For candidates?

**Theorem:** Job Propose and Reject produces a job-optimal pairing.

**Proof:**
Assume not: a job \( b \) is not paired with optimal candidate, \( g \).
There is a stable pairing \( S \) where \( b \) and \( g \) are paired.
Let \( t \) be first day job \( b \) gets rejected
by its optimal candidate \( g \) who it is paired with
in stable pairing \( S \).
\( b^- \) - knocks \( b \) off of \( g \)'s string on day \( t \) \( \implies g \) prefers \( b^- \) to \( b \)
By choice of \( t \), \( b^- \) likes \( g \) at least as much as optimal candidate.
\( \implies b^- \) prefers \( g \) to its partner \( g^- \) in \( S \).
Rogue couple for \( S \).
So \( S \) is not a stable pairing. Contradiction.

Notes:
- \( S \) - stable. \((b^*, g^*) \) \( \in S \).
- But \((b^*, g)\) is rogue couple!
- Used Well-Ordering principle...Induction.

How about for candidates?

**Theorem:** Job Propose and Reject produces candidate-pessimal pairing.

**T** – pairing produced by JPR.
**S** – worse stable pairing for candidate \( g \).
In **T**, \((g, b)\) is pair.
In **S**, \((g, b^*)\) is pair.
\( g \) prefers \( b \) to \( b^- \).
**T** is job optimal, \( \implies b \) prefers \( g \) to its partner in \( S \).
\((g, b)\) is Rogue couple for \( S \)
\( S \) is not stable.
Contradiction.

Notes:
- Not really induction.
- Structural statement: Job optimality \( \implies \) Candidate pessimality.

Quick Questions.

How does one make it better for candidates?
Propose and Reject - stable matching algorithm. One side proposes.
Jobs Propose \( \implies \) job optimal.
Candidates propose. \( \implies \) optimal for candidates.

Takeaways.

Analysis of cool algorithm with interesting goal: stability.
“Economic”: different utilities.
Definition of optimality: best utility in stable world.
Action gives better results for individuals but gives instability.
Induction over steps of algorithm.
Proofs carefully use definition:
- Stability: Improvement Lemma plus every day the job gets to choose.
- Optimality proof:
- Job Optimality:
  contradiction of the existence of a better stable pairing.
  that is, no rogue couple by improvement, job choice, and well ordering principle. Candidate Pessimality:
  contradiction plus cuz job optimality implies better pairing.
- Life Lesson: ask, you will do better even if rejection is hard.