5112

Algorithms

course website ↓

Alex Conway
Tom Ristenpart

Intro

Algorithms (general)

Stable Matching

Course Details (Tour)

Intro

Algorithms

Stable Matching Problem

Candidates	Universities
Alex	Cornell
Jennifer	Princeton
Raj	Rutgers

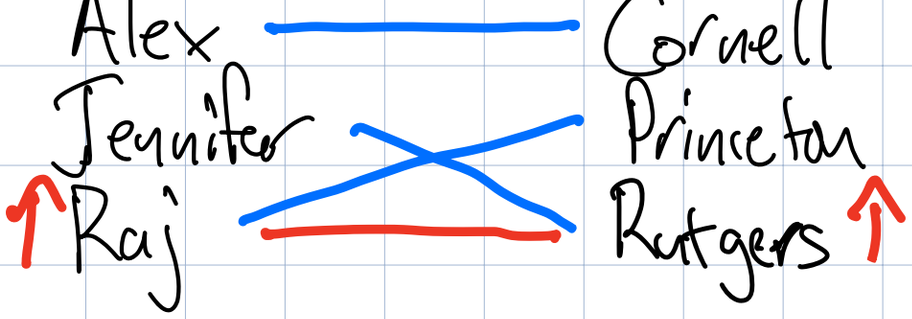
Stable Matching Problem

Candidates

Alex
Jennifer
Raj

Universities

Cornell
Princeton
Rutgers



Raj - Rutgers is called an unstable pair, both prefer each other to their match.

Preferences

Alex	Jennifer	Raj
Cornell	Princeton	Rutgers
Rutgers	Rutgers	Cornell
Princeton	Cornell	Princeton

Cornell	Princeton	Rutgers
Jennifer	Alex	Raj
Alex	Jennifer	Alex
Raj	Raj	Jennifer

Stable Matching Problem

Candidates

Alex
Jennifer
Raj

Universities

Cornell
Princeton
Rutgers

Raj - Rutgers is called an unstable pair, both prefer each other to their match.

Preferences

Alex	Jennifer	Raj
Cornell	Princeton	Rutgers
Rutgers	Rutgers	Cornell
Princeton	Cornell	Princeton

Cornell	Princeton	Rutgers
Jennifer	Alex	Raj
Alex	Jennifer	Alex
Raj	Raj	Jennifer

A stable matching is a matching with no unstable pairs

Does a stable matching exist?
How do you find one?
Are they unique?

Gale-Shapley Algorithm (1962)

while (∃ an unmatched university U):
    C ← next candidate
    if (C is unmatched or prefers U):
        match C to U (and unmatch C if necessary)

For each university, maintain a list of candidates in order of preference. By next, mean next in this list.

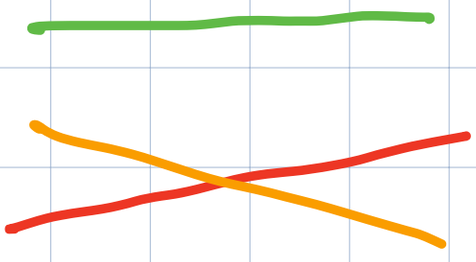
Stable Matching Problem

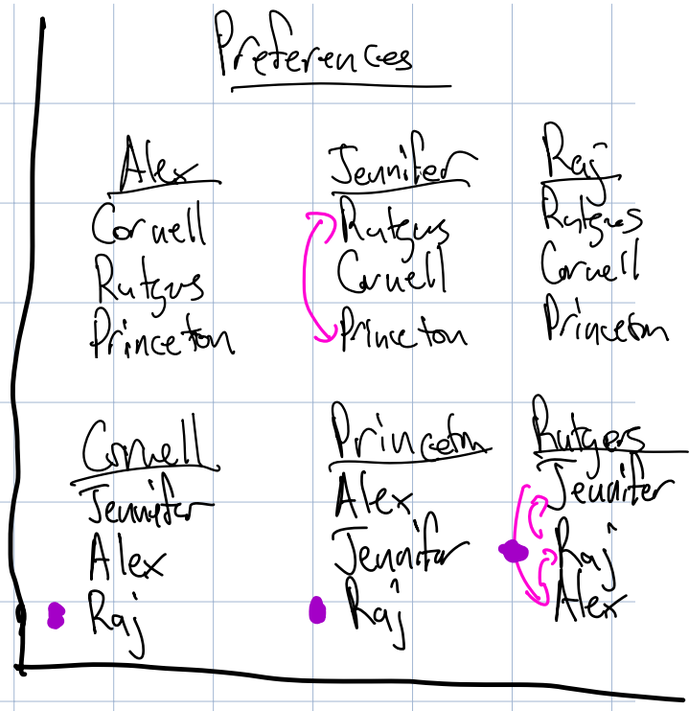
Universities

Cornell
Princeton
Rutgers

Candidates

Alex
Jennifer
Raj





1. Cornell makes an offer to Jennifer, accepts
2. Princeton makes an offer to Alex, accepts
3. Rutgers makes an offer to Jennifer, accepts and unmatches Cornell
4. Cornell makes an offer to Alex, accepts and unmatches Princeton
5. Princeton makes an offer to Jennifer, rejects 6. Princeton offers Raj, accepts

Gale-Shapley Algorithm (1962)

while (∃ an unmatched university U):
  C ← next candidate
  if (C is unmatched or prefers U):
    match C to U (and unmatch C if necessary)

For each university, maintain a list of candidates in order of preference. By next, mean next in this list.

Can we run out of candidates to make offers to?

Invariant: once a candidate receives an offer, they will have an offer for the rest of the algorithm.

If we reach the end of a candidate list, all candidates must have offers,

n candidates have offers ⟶ n universities have offers

Gale-Shapley Algorithm (1962)

while (∃ an unmatched university U):
  $C \leftarrow$ next candidate
  if (C is unmatched or prefers U):
    match C to U (and unmatch C if necessary)

For each university, maintain a list of candidates in order of preference. By next, mean next in this list.

Can we run out of candidates to make offers to?
Invariant: once a candidate receives an offer, they will have an offer for the rest of the algorithm.
If we reach the end of a candidate list, all candidates must have offers.
n candidates have offers $\longrightarrow$ n universities have offers

How many times do we go through the loop?
Each university can make an offer to each candidate at most once.
$\Rightarrow$ At most $n \cdot n = n^2$, because there are n universities and n candidates.

What if we use a hash table to store candidates' preferences?

Alex: U_1, rank

Gale-Shapley Algorithm (1962)

while (∃ an unmatched university U):
  C ← next candidate
  if (C is unmatched or prefers U):
    match C to U (and unmatch C if necessary)

For each university, maintain a list of candidates in order of preference. By next, mean next in this list.

How many times do we go through the loop?
Each university can make an offer to each candidate at most once.
⇒ At most $n \cdot n = n^2$, because there are n universities and n candidates.

Can we run out of candidates to make offers to?
Invariant: once a candidate receives an offer, they will have an offer for the rest of the algorithm.
If we reach the end of a candidate list, all candidates must have offers.
n candidates have offers ⟶ n universities have offers

Each university keeps an array of the candidates' ranks for that university.

Gale-Shapley Algorithm (1962)

while (∃ an unmatched university U):
  $C \leftarrow$ next candidate
  if (C is unmatched or prefers U):
    match C to U (and unmatch C if necessary)

For each university, maintain a list of candidates in order of preference. By next, mean next in this list.

Can we run out of candidates to make offers to?
Invariant: once a candidate receives an offer, they will have an offer for the rest of the algorithm.
If we reach the end of a candidate list, all candidates must have offers.
n candidates have offers $\longrightarrow$ n universities have offers

How many times do we go through the loop?
Each university can make an offer to each candidate at most once.
$\Rightarrow$ At most $n \cdot n = n^2$, because there are n universities and n candidates.

What is the cost of each loop iteration?
- Need to find candidate preferences efficiently