Lecture 13: Stable Matching
Overview of Greedy Algorithms

The Stable Marriage Problem (input)

Goal. Given n men, n women, and their preference lists, find a "suitable" matching.

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

favorite ↓ least favorite ↓

	1st	2nd	3rd
X	A	B	C
Y	B	A	C
Z	A	B	C

Men's Preference lists

favorite ↓ least favorite ↓

	1st	2nd	3rd
A	Y	X	Z
B	X	Y	Z
C	X	Y	Z

Women's Preference lists

The Stable Marriage Problem (input)

Goal. Given n men, n women, and their preference lists, find a "suitable" matching.

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

- Problem historically stated in terms of men/women but actually has broad applications
  - Matching Medical students and Hospital residency places in US
  - Auction mechanisms for sponsored internet search
  - JUPAS
  - ……

The Stable Marriage Problem (output)

A set of n m-w pairs that constitute a perfect and stable matching

Perfect matching: everyone is matched *monogamously*.

- Each man is matched to exactly one woman.
- Each woman is matched to exactly one man.

Stability: no incentive for any unmatched pair to undermine assignment by joint action.

- In matching M, an unmatched pair m-w is unstable if man m and woman w prefer each other to their current partners.
- Unstable pair m-w could each improve by divorcing their current partners and getting together.

Stable matching: perfect matching with no unstable pairs.

Stable marriage/matching problem.
Given the preference lists of n men and n women, find a stable matching (if one exists).

Stable Matching Problem

Q. Is assignment X-C, Y-B, Z-A stable?

favorite ↓ least favorite ↓

	1st	2nd	3rd
X	A	B	C
Y	B	A	C
Z	A	B	C

Men's Preference Lists

favorite ↓ least favorite ↓

	1st	2nd	3rd
A	Y	X	Z
B	X	Y	Z
C	X	Y	Z

Women's Preference Lists

Stable Matching Problem

Q. Is assignment X-C, Y-B, Z-A stable?

No. X-B is an unstable pair.

X and B prefer each other to their current matches (C and Y, respectively).

favorite ↓ least favorite ↓

	1st	2nd	3rd
X	A	B	C
Y	B	A	C
Z	A	B	C

Men's Preference Lists

favorite ↓ least favorite ↓

	1st	2nd	3rd
A	Y	X	Z
B	X	Y	Z
C	X	Y	Z

Women's Preference Lists

Stable Matching Problem

Q. Is assignment X-A, Y-B, Z-C stable?

A. Yes.

favorite ↓ least favorite ↓

	1st	2nd	3rd
X	A	B	C
Y	B	A	C
Z	A	B	C

Men's Preference Lists

favorite ↓ least favorite ↓

	1st	2nd	3rd
A	Y	X	Z
B	X	Y	Z
C	X	Y	Z

Women's Preference Lists

Stable Matching Problem

Q. Do stable matchings always exist?

A. Yes. (This is not obvious; we will prove later)

Q. Is the stable matching unique?

A. No. It is possible that there are several stable matchings

favorite ↓ least favorite ↓

	1st	2nd	3rd
X	A	B	C
Y	B	A	C
Z	A	B	C

Men's Preference Lists

favorite ↓ least favorite ↓

	1st	2nd	3rd
A	Y	X	Z
B	X	Y	Z
C	X	Y	Z

Women's Preference Lists

Shaded boxes are stable matching X-B, Y-A, Z-C

Previous page had stable matching X-A, Y-B, Z-C for same lists!

Propose-And-Reject Algorithm

Propose-and-reject algorithm. [Gale-Shapley 1962]
Intuitive algorithm that guarantees to find a stable matching.

```
Initialize each person to be free.
while (some man is free and hasn't proposed to every woman) {
    Choose such 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 engaged
    else if (w prefers m to her fiancé m')
        assign m and w to be engaged, and m' to be free
    else
        w rejects m
}
```

Shapley won Nobel Prize in Economics (partially) for this in 2012.
Gale was not eligible because he had died in 2008.

Proof of Correctness: Termination

Claim. Algorithm terminates after at most n^2 iterations of while loop.

Pf. A man starts from the first woman in his list and then continues in decreasing order of preference, without proposing to the same woman again. There are only n^2 possible proposals. ▪

Proof of Correctness: Perfection

Perfection means that in the end of the algorithm each man and woman gets matched to exactly one partner.

Observation 1. Men propose to women in decreasing order of preference.

Observation 2. Once a woman is matched, she never becomes unmatched; she only "trades up."

Claim. All men and women get matched.

Pf. (by contradiction)

- Suppose, for sake of contradiction, that man Z is not matched upon termination of algorithm.
- Then some woman, say A, is not matched upon termination.
- By Observation 2, A was never proposed to.
- But, Z must have proposed to everyone, since he ends up single.
- Contradiction! ▪

Proof of Correctness: Stability

Claim. No unstable pairs.

Pf. (by contradiction)

- Suppose A-Z is an unstable pair: each prefers each other to their partner in Gale-Shapley matching S*.

- Case 1: Z never proposed to A.
  - ⇒ Z prefers his GS partner to A.
  - ⇒ A-Z is stable.

men propose in decreasing order of preference

- Case 2: Z proposed to A.
  - ⇒ A rejected Z (right away or later)
  - ⇒ A prefers her GS partner to Z.
  - ⇒ A-Z is stable.

women only trade up

- In either case A-Z is stable, a contradiction. ▪

S*
A-Y
B-Z
. . .

Efficient Implementation

Efficient implementation. We describe $O(n^2)$ time implementation.

Representing men and women.

- Assume men are named 1, …, n.
- Assume women are named 1', …, n'.

Engagements.

- Maintain a list of free men, e.g., in a queue or stack.
- Maintain two arrays `wife[m]`, and `husband[w]`.
  - if m matched to w then `wife[m]=w` and `husband[w]=m`
  - Otherwise, set entry to `0` if unmatched

Men proposing.

- For each man, maintain a list (linked list or array) of women, ordered by preference.

Efficient Implementation

Women rejecting/accepting.

- Does woman `w` prefer man `m` to man `m'`?
- Naïve implementation requires O(n) time for this comparison.
- For each woman, create inverse of preference list of men.
- Constant time access for each query after O(n) preprocessing.

Amy	1st	2nd	3rd	4th	5th	6th	7th	8th
Pref	8	3	7	1	4	5	6	2

Amy	1	2	3	4	5	6	7	8
Inverse	4th	8th	2nd	5th	6th	7th	3rd	1st

```
for i = 1 to n
   inverse[pref[i]] = i
```

Amy prefers man 3 to man 6
since `inverse[3]` < `inverse[6]`

2 7

Understanding the Solution

Q. For a given problem instance, several stable matchings might exist.
Recall that Gale-Shapely gives us freedom to decide *which* man proposes.
Do all executions of Gale-Shapley (for different orders in which men propose) yield the same stable matching?
If so, which one?
A. Yes, it always returns the (unique) matching that is optimal for the men (proof omitted).

> Optimal for the men means: each man gets his best possible partner in any possible stable matching.

Observation. Man and Women are not equivalent in the algorithm.
Men propose, women accept/reject.

> To get a woman optimal algorithm, have women propose and men accept/reject.

Several variants of the problem exist with multiple applications.

Stable Matching: Questions

Consider the group m1, m2, w1, and w2 with preference lists:

m1: w1, w2	m2: w2, w1
w1: m2, m1	w2:m1, m2

1. What is a man-optimal stable matching in the above example:
(m1,w1), (m2,w2)

2. What is a woman-optimal stable matching in the above example:
(m1,w2), (m2,w1)

3.In every instance of the Stable Matching Problem, there is a stable matching containing a pair (m, w) such that m is ranked first on the preference list of w and w is ranked first on the preference list of m.
True or False?
False. See above example

4. Consider that there exists a man m and a woman w such that m is ranked first on the preference list of w and w is ranked first on the preference list of m. Then, the pair (m,w) belongs to every stable matching. True or False?
True. Otherwise the pair (m,w) would be unstable.

Stable Matching: Questions 2

1. Suppose that each man has a different most favorite woman. How many steps does it take for the algorithm to converge?

A single round. Each man proposes to a different woman who accepts the invitation. The woman will not be proposed again.

2. Suppose that the men have identical preferences. How many rounds does it take for the algorithm to converge?

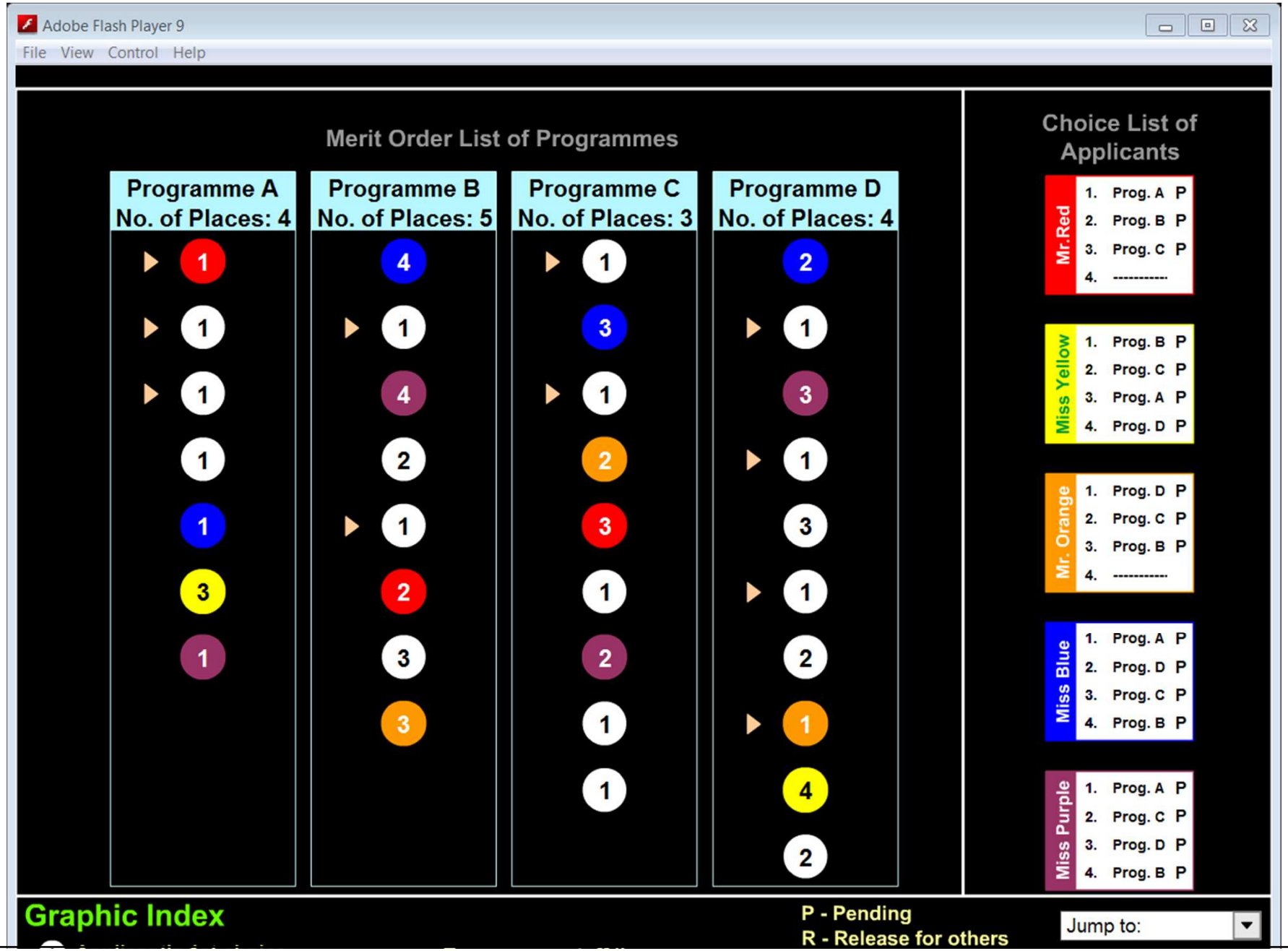
At round 1, all men propose to the same woman. She dates her favorite. At the second round, all but that favorite man proposes to the second-best woman. She dates her favorite. The process continues for N rounds until one man is left to propose to the least desirable woman.

Roommates Problem

$2n$ people must be paired up to be roommates. For each person we have the list of preferences, but the graph is not bi-partite (a person can be matched with any of the remaining $2n-1$). Is it always possible to find a stable match?

No. Assume 4 people A,B,C,D with preference lists:
A: B C D
B: C A D
C: A B D
D: any
Any matching will contain an unstable pair:
If we match (A,D), (B,C), then (A,C) is unstable.
If we match (B,D), (A,C), then (A,B) is unstable.
If we match (C,D), (A,B), then (B,C) is unstable.

JUPAS Admission Scheme



JUPAS Admission Scheme

Applicants correspond to men that order their programme choices (which correspond to women).
Differences w.r.t. to the basic version of the problem:

1. Many more applicants than programmes (places)
2. A programme is matched to many (instead of one) applicants.
   # it's matched to is its *intake*

http://www.jupas.edu.hk/
The JUPAS computer system will match (the "iteration process") the order of preference you have assigned to your programme choice list with the position you have been placed in each merit order list of these programmes.

You will then be made an offer of the highest priority on your programme choice list for which you have the required rating.

That is, the matching is optimal (i.e., fair) for the applicants!

Greedy Overview

1. Greedy algorithms are often the simplest possible algorithms to design
   - They build solutions, one step at a time.
   - Sometimes the solution is optimal, sometimes not.

2. Proving that the greedy solution is optimal is usually the hard part
   - Greedy is not always optimal (often isn't)
   - There is no one way to prove optimality
     - We saw three standard approaches
     - Note that not every method can work for every problem.

Three Different Proof Techniques

1. Modifying an Optimal Solution into Greedy (cost common)
   - Did this for **Interval Scheduling** & **Fractional Knapsack** and several exercises
   - Start (conceptually) with different ***G***(reedy) and ***O***(optimal) solutions
     - Show how ***O*** can be modified to ***O'*** that is still optimal but closer to ***G***
     - Repeat, until have created optimal solution that is exactly ***G***

2. Lower Bound Technique
   - Did this for **Interval Partitioning**
   - For problems trying to find **minimum** solution (can be modified for max)
     - Define value **L** that is a lower bound on ANY feasible solution
     - Show that Greedy solution has value **L**

3. Algorithm-specific Techniques
   - Inductive Proof for **Huffman Coding**
   - Proof by Contradiction for **Stable Matching**

Exercise on Stairs Climbing

Problem: You can climb 1 or 2 stairs with one step.
How many *different* ways can you climb n stairs?

Solution: Let $\mathrm{F}(n)$ be the number of different ways to climb n stairs.

$$F(1) = 1, F(2) = 2, F(3) = 3, \ldots$$
$$F(n) = F(n-1) + F(n-2)$$

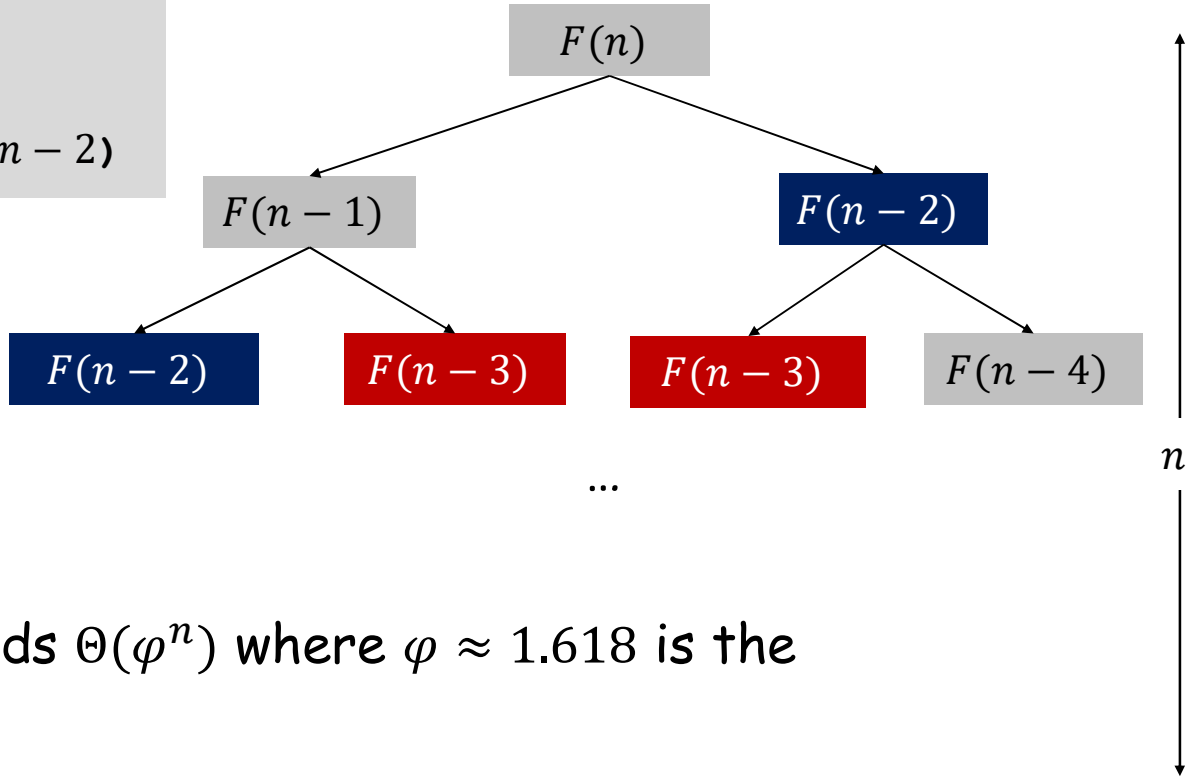
Because you can only reach n^{th} stair from the $(n\text{-}1)^{th}$ or the $(n\text{-}2)^{th}$.

Observation: F(1), F(2), …., F(n) are the Fibonacci numbers.

Solving the recurrence by recursion

$$F(1) = 1, \qquad F(2) = 2$$
$$F(n) = F(n-1) + F(n-2)$$

```
F(n):
if n = 1 return 1
if n = 2 return 2
return F(n − 1)+F(n − 2)
```



Running time?

Between $2^{n/2}$ and 2^n.

A deeper analysis yields $\Theta(\varphi^n)$ where $\varphi \approx 1.618$ is the golden ratio.

Q: Why so slow?

A: Solving the same subproblem many many times.

Solving the recurrence by dynamic programming

$$F(1) = 1, \qquad F(2) = 2$$
$$F(n) = F(n-1) + F(n-2)$$

```
F(n):
allocate an array F of size n
F[1] ← 1
F[2] ← 2
for i = 3 to n
    F[i] ← F[i − 1] + F[i − 2]
return F[n]
```

Running time: $\Theta(n)$

Space: $\Theta(n)$?

Dynamic programming:

- Used to solve recurrences
- Avoid solving a subproblem more than once by storing solutions
- Usually done "bottom-up", filling in subproblem solutions in table in order from "smallest" to largest".
  - There is also "top-down" version (memoization). Essentially equivalent
- "Programming" here means "planning", not coding!