(J, ~~C~~) → (J', C)

1 Stable Matching

Consider the set of candidates $C = \{1, 2, 3\}$ and the set of jobs $J = \{A, B, C\}$ with the following preferences.

C	J		
1	A	B	C
2	B	A	C
3	A	B	C

J	C		
A	2	1	3
B	1	2	3
C	1	2	3

Run the applicant propose-and-reject algorithm on this example. How many days does it take and what is the resulting pairing? (Show your work)

Jobs propose

	Day 1	Day 2	Day 3
1	B C	B	B
2	A	A C	A
3			C

A $S_1 > S_2$
B $S_1 > S_2$
C $S_1 = S_1$

1 (1, B)
(2, A)
(3, C)

Candidates propose

	Day 1	Day 2	Day 3
A	1 ~~3~~	1	1
B	2	2 ~~3~~	2
C			3

2 (A, 1)
(B, 2)
(C, 3)

2 Good, Better, Best

In a particular instance of the stable marriage problem with n applicants and n jobs, it turns out that there are exactly three distinct stable matchings, S_1, S_2, and S_3. Also, each applicant m has a different partner in the three matchings. Therefore each applicant has a clear preference ordering of the three matchings (according to the ranking of his partners in his preference list). Now, suppose for applicant m_1, this order is $S_1 > S_2 > S_3$.

App-opt

Prove that every applicant has the same preference ordering $S_1 > S_2 > S_3$. — Emp. opt / App-pess

Hint: Recall that a applicant-optimal matching always exists and can be generated using applicant proposes matching algorithm. By reversing the roles of stable matching algorithm, what other matching can we generate?

1 J_1 J_2 . . . J_n

2

3

⋮

n

$S_1 > S_3$

For everyone!

$(m_1, j) \in S_1$

$S_1 > S_2$

$(m_1, j') \in S_3$

$S_2 > S_3$

$S_1 > S_2 > S_3$