Lecture 3 (Jan 27, 2019)

Announcements

→ HW

→ TA office hours.

→ Partner finding

Recall the notion of Stable matching

$E = \{e_1, \dots, e_n\}$ $A = \{a_1, \dots, a_n\}$

$M \subset E \times A$ is a stable matching if

(a) perfect matching

(b) no unstable pair

(e, a) is unstable if:

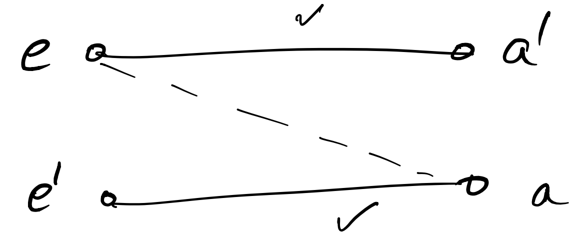
(i) $(e, a) \notin M$

(ii) $\exists\ a', e'$ s.t.

$(e, a'), (e', a) \in E$

$e : a > a'$

$a : e > e'$



Question. → Existence of a stable matching?
→ Uniqueness? (No)

G-S algorithm (Gale, Shapley 1962)

Input: $E = \{e_1, \ldots, e_n\}$ $A = \{a_1, \ldots, a_n\}$

their rankings.

Algorithm (i) Initialize all $e \in E$, $a \in A$ to be 'free'.

(*) while $\exists\ a \in A$ who is free 'and a has not proposed to all $e \in E$'

a proposes to highest-ranked employer she hasn't yet proposed to.

Let e be this employer.

(a proposes to e)

✓ If e is currently matched
to a', s.t $e: a' > a$
then a remains 'free'.
Else (e, a) are paired.

endwhile.

Run time ?

(a) May not terminate

(b) $2^{O(n)}$

(c) $O(n^3)$

(d) $O(n^2)$

Run-time analysis:

(i) While loop: atmost n^2 iterations

Since each a proposes to
any e at most once.

(ii) With right data-structures,
each iteration is $O(1)$ time.

Which data-structures?

(i) rankings as linked lists.

(ii) maintain a state for each a, e for current pairings.

(iii) 'reverse index array' for each $e \in E$.

e.g. e

a_2	a_1	a_3

$(a_2 > a_1 > a_3)$

a_1	a_2	a_3
2	1	3

(*) Preprocessing : $O(n^2)$

$$n^2 \cdot O(1) + O(n^2) = \underline{\underline{O(n^2)}}.$$

<u>Proof of correctness</u>

Let M be the matching returned.

Obs 1 : M is a perfect matching.

Pf: Suppose not.

$\exists$ $a \in A$ which is unmatched.
$\Rightarrow$ $\exists$ $e \in E$ " .

e : starts out as 'free'.

What happens when a proposed to e?.... (When did a get unpaired from e?)

$\exists$ $a' \in A$ / (e, a') were paired and $e: a' > a$.

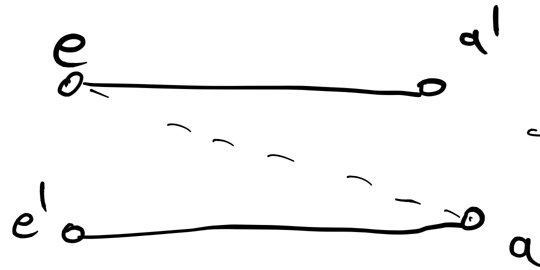
But then e cannot be free!
Contradiction.
Hence M is a perfect matching.

(ii) No unstable pairs (w.r.t M)

Pf: Suppose not.
Let (e, a) be unstable.

(i) $(e,a) \notin M$.

(ii) $\exists\ a', e'$ s.t



$e : a > a'$

$a : e > e'$

$(e,a') \in M, \quad (e',a) \in M$.

$\to$ a must have proposed to e.

So, some $a'' \in A$ must have 'displaced' a from e.

(At some pt in the algoritm) (e, a'') were matched.

$e : a'' > a$

$\to$ $a' = a''$

or $e : a' > a''$.

$\Rightarrow$ $e : a' > a$ which is a contradiction.

No unstable pair :)

M is a stable matching!

Fairness of the algorithm.

The algorithm finds the 'applicant-optimal' stable matching.

Def^n: For $a \in A$,

$$\text{valid}(a) = \{ e \in E : \exists \text{ a stable matching } M, (e,a) \in M \}.$$

$$\text{Valid}(a) \subseteq E$$

$\text{best}(a)$ = best ranked e in $\text{Valid}(a)$.

Lemma: The output of G-S algorithm

$$M = \{ (\text{best}(a), a) : a \in A \}.$$

(Is M even a matching?)

Proof:

Consider the 1^{st} moment (in the algorithm) that some a was rejected by a valid partner, i.e, $e \in valid(a)$.

(or a' displaced a from (e,a))

$\exists\ a' \in A,\quad e: a' > a$

Since (e,a) is a valid pair,

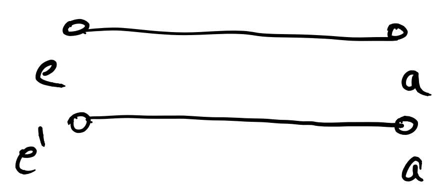
$\exists$ a stable matching M' s.t

$(e,a) \in M'$.

In M', what is a' matched to?

Let $(e',a') \in M'$.

Since M' is stable,



a' : $e' > e$ (since $e : a' > a$)

What happened when a' proposed to e' in the G-S algorithm.

$e' \in$ valid(a') but e' rejected a' contradicting that the first time some valid $a \in A$ was rejected was a by e.

$\Rightarrow$ G-S in fact outputs $\{(\text{best}(a), a) : a \in A\}$.