

# NOTE BOOK

Rokshana Ahmed

# Algorithms

DATE : 1/3/2022

# CONTINUE of Previous Lecture

## PROOF OF THEOREM A

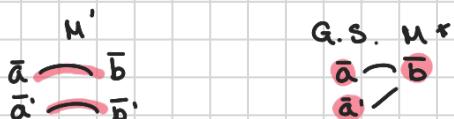
By contradiction, suppose that some " $a_i$ " ends up being matched to a partner other than  $\text{best}(a_i)$  in  $M^+$

Since the  $a$ 's propose in decreasing order of preference, there must be a time when some " $a'$ " gets rejected by one of its valid matches " $b'$ " (rejection can happen right after a proposal or when " $b$ " accepts some other proposal)

Let the pair  $\{\bar{a}, \bar{b}\}$  be the first pair (during the execution of G-S algorithm) that is valid, and such that  $b$  rejects  $\bar{a}$ .

When this rejection happens,  $\bar{b}$  will be paired up with some other  $\bar{a}'$  that she likes better than  $\bar{a}$ .

Now, since  $\{\bar{a}, \bar{b}\}$  is a valid pair there must exist a stable matching  $M'$  such that  $\{\bar{a}, \bar{b}\} \in M'$ . Since  $M'$  is a stable matching, it must match  $\bar{a}'$  to some  $\bar{b}'$ ,  $\{\bar{a}', \bar{b}'\} \in M'$



Then observe that  $\{\bar{a}', \bar{b}'\}$  is a valid pair.

Given that  $\{\bar{a}, \bar{b}\}$  was the first valid pair with a rejection, it must be that  $\bar{a}'$  was not rejected by a valid partner before  $\bar{a}'$  gets engaged with  $\bar{b}$  1st claim

2nd claim Now, since  $\bar{b}'$  is a valid partner of  $\bar{a}'$  ( $\{\bar{a}, \bar{b}'\} \in M'$ ), and since  $\bar{a}'$  proposes in decreasing order of preference, it must be that  $\bar{a}'$  prefers  $\bar{b}'$  to  $\bar{b}$  3rd claim

Since  $\bar{b}$  prefers  $\bar{a}'$  to  $\bar{a}$  (she rejected  $\bar{a}$  to be with  $\bar{a}'$ ) and since  $\{\bar{a}, \bar{b}\}, \{\bar{a}', \bar{b}'\} \in M$ . It holds that  $\{\bar{a}, \bar{b}\}, \{\bar{a}', \bar{b}'\} \in M'$  is an instability of  $M'$  thus  $M'$  is unstable. This is a contradiction.

## PROOF OF THEOREM B

Suppose that  $\exists \{\bar{a}, \bar{b}\} \in M^*$  s.t.  $\bar{a} \neq \text{worst}(\bar{b})$ .

Then, there exists a stable matching  $M'$  s.t.  $\{\bar{b}, \bar{a}\} \in M'$  and  $\bar{b}'$  likes  $\bar{a}'$  less than  $\bar{a}$ .

Suppose that in  $M'$ ,  $\bar{a}$  is matched with some  $\bar{b}' \neq \bar{b}$  ( $\{\bar{a}, \bar{b}'\} \in M'$ ).

By theorem A, we know that  $\bar{b}$  is the best valid partner of  $\bar{a}$  thus,  $\bar{a}$  likes  $\bar{b}$  more than  $\bar{b}'$  (or thm A would not hold). that is  $\bar{a}: \bar{b} > \bar{b}'$  and  $\bar{b}: \bar{a} > \bar{a}'$  but,  $\{\bar{a}, \bar{b}'\}, \{\bar{a}, \bar{b}\} \in M'$ .

Thus,  $= =$  form an instability of  $M'$ , which is then unstable. **CONTRADICTION**

G-S has given us the opportunity to discuss about a number of questions one encounters when studying algorithm problems.

- ① Formulated the problem in a mathematical precise way
- ② Ask questions about the mathematical problem
- ③ Design an efficient algorithm for your problem
- ④ Prove that your assumption is correct, and bound its runtime

## EFFICIENCY

Def (??) "An algorithm is efficient if, after being implemented, it runs quickly on real input instances."  
(not a formal definition)

"QUICKLY":

"quick" on which hardware? with which programming language? with which implementation? (each of these questions significantly changes the runtime)

We would like our definition to be independent of details

## REAL INPUT INSTANCES

: what does it mean? How to define them?

To prove something about an algorithm, we need mathematical definitions.

## WORST-CASE ANALYSIS

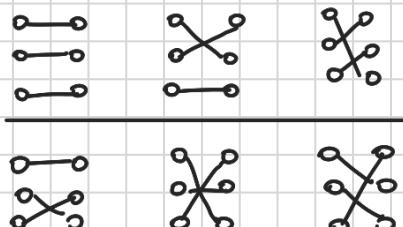
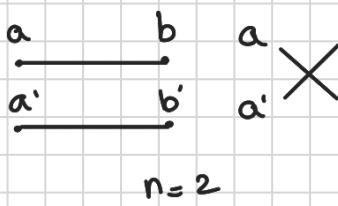
We want the runtime of an algorithm to be bounded in terms of the worst possible input. For instance, in the case of stable matching, we proved that G-S algorithm uses at most  $n^2$  iterations ( $n = |A| = |B|$ )

Suppose, for instance that we consider inputs in which  $|A| = |B| = N$ , for  $N = 2n$

$$N^2 = (2n)^2 = 4n^2$$

Doubling  $n$  changes the runtime by a factor 4

The brute-force algorithm tried each of the  $n!$  matches



What would happen to the brute-force algorithm if we doubled the number of people?

$$N! = (2n)! =$$

$$\begin{aligned} &= 2n(2n-1)(2n-2) \dots (n+1) \cdot n(n-1) \cdot 2 \cdot 1 \\ &= 2n(2n-1) \dots (n+1) n! \\ &\geq (n+1)^n \cdot n! \geq 2^n \cdot n! \end{aligned}$$

Def: An algorithm having a runtime  $\leq c \cdot n^d$ , where  $c$  and  $d$  are constants, on inputs of size  $n$  is said to be a polynomial time (POLYTIME) algorithm

If I have a polytime algorithm running in time  $c \cdot n^d$  on inputs of size  $n$ , and I run it on inputs of size  $N = b \cdot n$ ,

$$c \cdot N^d = c \cdot (nb)^d = c \cdot n^d \cdot b^d$$

then the blow-up in the runtime is going to be a constant  $b^d$