Lecture 8

CSE 331

Feb 12, 2020

The Lemmas

Lemma 1: The GS algorithm has at most n^2 iterations

Lemma 2: S is a perfect matching

Lemma 3: S has no instability

GS outputs a stable matching

Feb 10 THEOREM: For any input (M, W, 2n preference lists) the GS algorithm outputs a stable matching.

⟹ every input has a stable matching.

LEMMA 1: For every input, the GS algo. terminates in $\leq n^2$ iterations

LEMMA 2: The output of GS algo (S) is a perfect matching

LEMMA 3: S has no instability.

Lemmas 1+2+3 ⟹ Theorem

Pf idea Lemma 1: In each iteration, a new proposal is made (from w to m)

$\Rightarrow$ # iterations = # proposals $\leq$ # pairs $(w,m) = |W \times M| = |W| \cdot |M| = n \cdot n = n^2$

(Pf details are on pg 7 in book)

Obs 0: S is a matching.

Obs 1: Once a man gets engaged, he keeps getting engaged to better women

Obs 2: If w proposes to m after $m' \Rightarrow m' > m$ in L_w

LEMMA 4: If at the end an iteration, w is free ⟹ w has NOT proposed to all men.

Proof Details of Lemma 1

CSE 331 Support Pages

Using a Progress Measure

This is another trick that you might not have studied formally but have used (implicitly) before. This trick is generally used to bound the number of times a loop is executed in an algorithm.

Background

In this note, we will consider another trick that you might not have studied formally but have used (implicitly) before. This trick is generally used to bound the number of times a loop is executed in an algorithm. Since most non-trivial algorithms have loops in them, this is a useful trick to remember when trying to bound the run time of an algorithm (which you will have to do frequently in this course). Most of the time you will need to use the trivial version of this trick.

A simple example

Let us begin with a prototypical example that you have already seen. Consider the following simple problem:

Search Problem

Given $n+1$ numbers $a_1, \ldots, a_n; v$, we should output $1 \le i \le n$ if $a_i = v$ (if there are multiple such i's then output any one of them) else output -1.

Below is a simple algorithm to solve this problem.

Linear Search Algorithm

```
// Input: A[i] for 0<= i<n
// Input: v

// Search
for(i=0; i< n; i++)
    if(A[i] == v)
        Return i;

Return -1;
```

Proof technique de jour

Proof by contradiction

Assume the negation of what you want to prove

After some reasoning

Source: 4simpsons.wordpress.com

Two observations

Obs 1: Once m is engaged he keeps getting engaged to “better” women

Obs 2: If w proposes to m’ first and then to m (or never proposes to m) then she prefers m’ to m

Proof of Lemma 2

Obs 0: S is a matching.

Obs 1: Once a man gets engaged, he keeps getting engaged to better women

Obs 2: If w proposes to m after m' ⟹ m' > m in L_w

LEMMA 4: If at the end an iteration, w is free ⟹ w has NOT proposed to all men.

Pf of Lemma 2: (Pf idea) Proof by contradiction (use Obs 0, Lemmas 1+4, algo. def)

(Pf details): Assume S is not a perfect matching.

⟹ (Obs 0, Algo def) ∃ a free woman w ⟹ (Lemma 4) ∃ a man m that w has not proposed to. (*)

By Lemma 1, algo has terminated ⟹ (Algo defn) All free woman have proposed to all men ⟹ contradicts (*)

Proof of Lemma 2

Lemma 4:

If at the end of an iteration, w is free

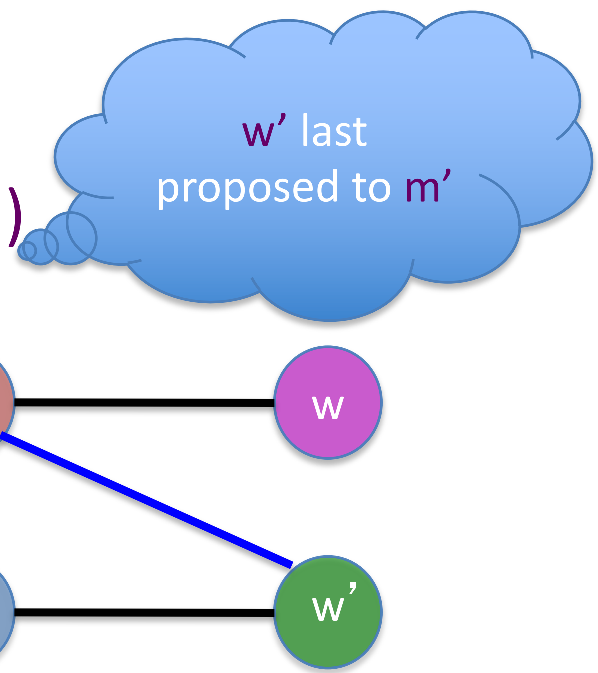
then w has **not** proposed to all men

Pigeon-hole principle!

Proof of Lemma 3

By contradiction

Assume there is an instability (m,w’)



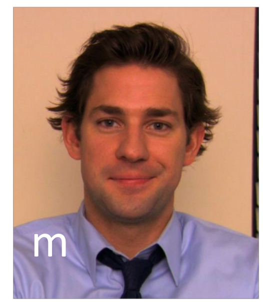
m prefers w’ to w

w’ prefers m to m’

Contradiction by Case Analysis

Depending on whether w’ had proposed to m or not

Case 1: w’ never proposed to m

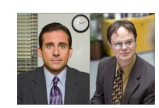


w’ prefers m’ to m

By Obs 2

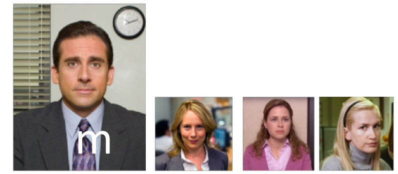
Assumed w’ prefers m to m’

Source: 4simpsons.wordpress.com



Case 2: w’ had proposed to m

Case 2.1: m had accepted w’ proposal



m is finally engaged to w

By Obs 1

Thus, m prefers w to w’

4simpsons.wordpress.com

Case 2.2: m had rejected w’ proposal

m was engaged to w’’ (prefers w’’ to w’)

By Obs 1

m is finally engaged to w (prefers w to w’’)

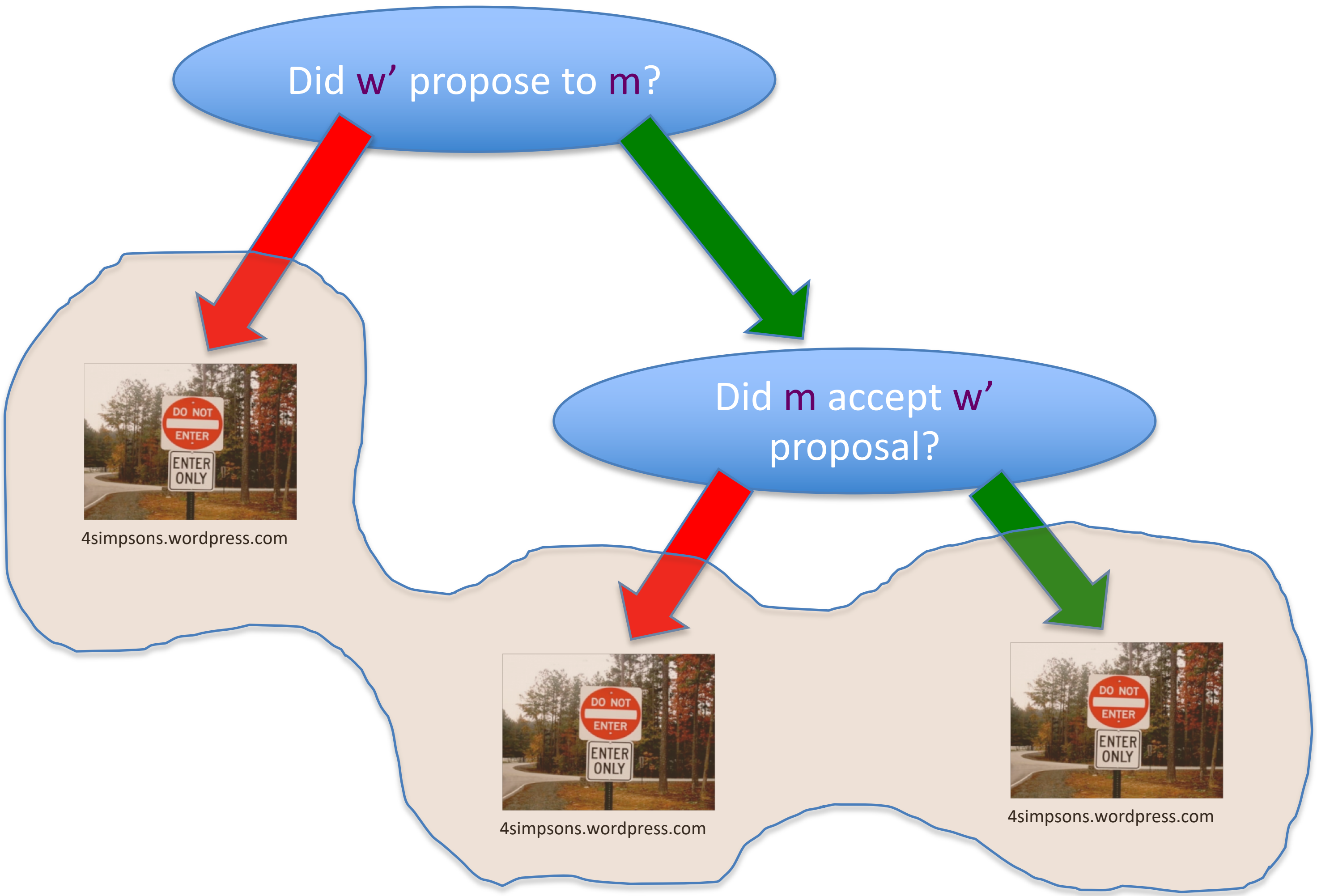
By Obs 1

m prefers w to w’

4simpsons.wordpress.com

Proof of the theorem is done!

Overall structure of case analysis



Questions?

Extensions

Fairness of the GS algorithm

Different executions of the GS algorithm

Main Steps in Algorithm Design

Problem Statement

↓

Problem Definition

↓

Algorithm

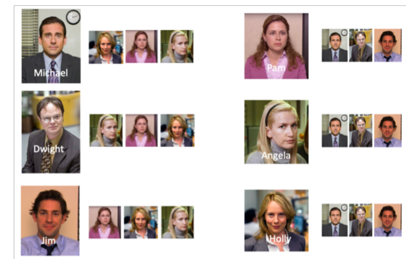
↓

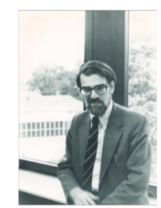
"Implementation"

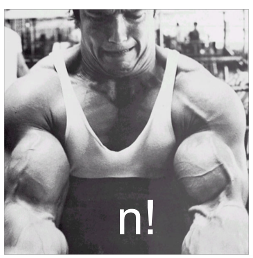
↓

Analysis

NRMP
National Resident Matching Program







Correctness Analysis

Definition of Efficiency

An algorithm is efficient if, when implemented, it runs quickly on real instances

Implemented where?



Platform independent definition

What are real instances?

Worst-case Inputs

$N = 2n^2$ for SMP

Efficient in terms of what?

Input size N