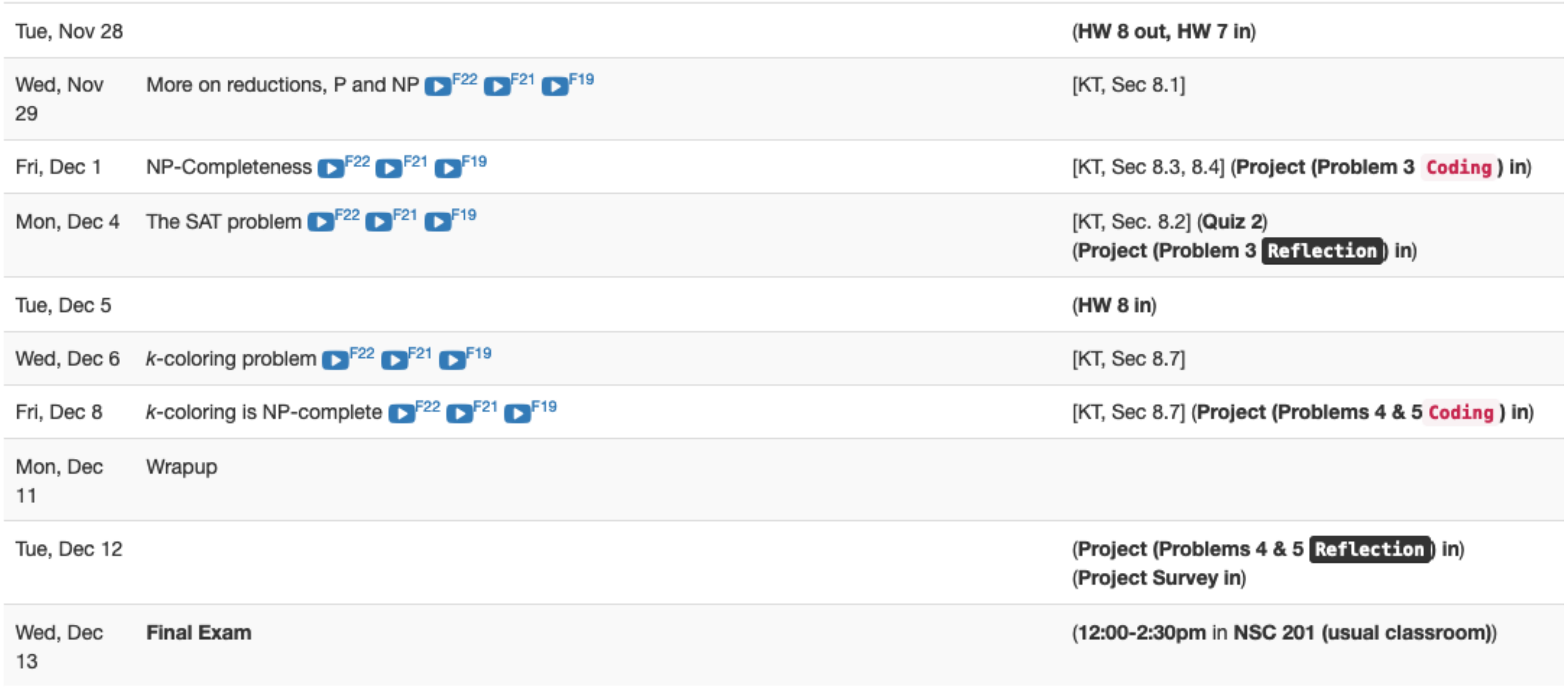
Lecture 35

CSE 331

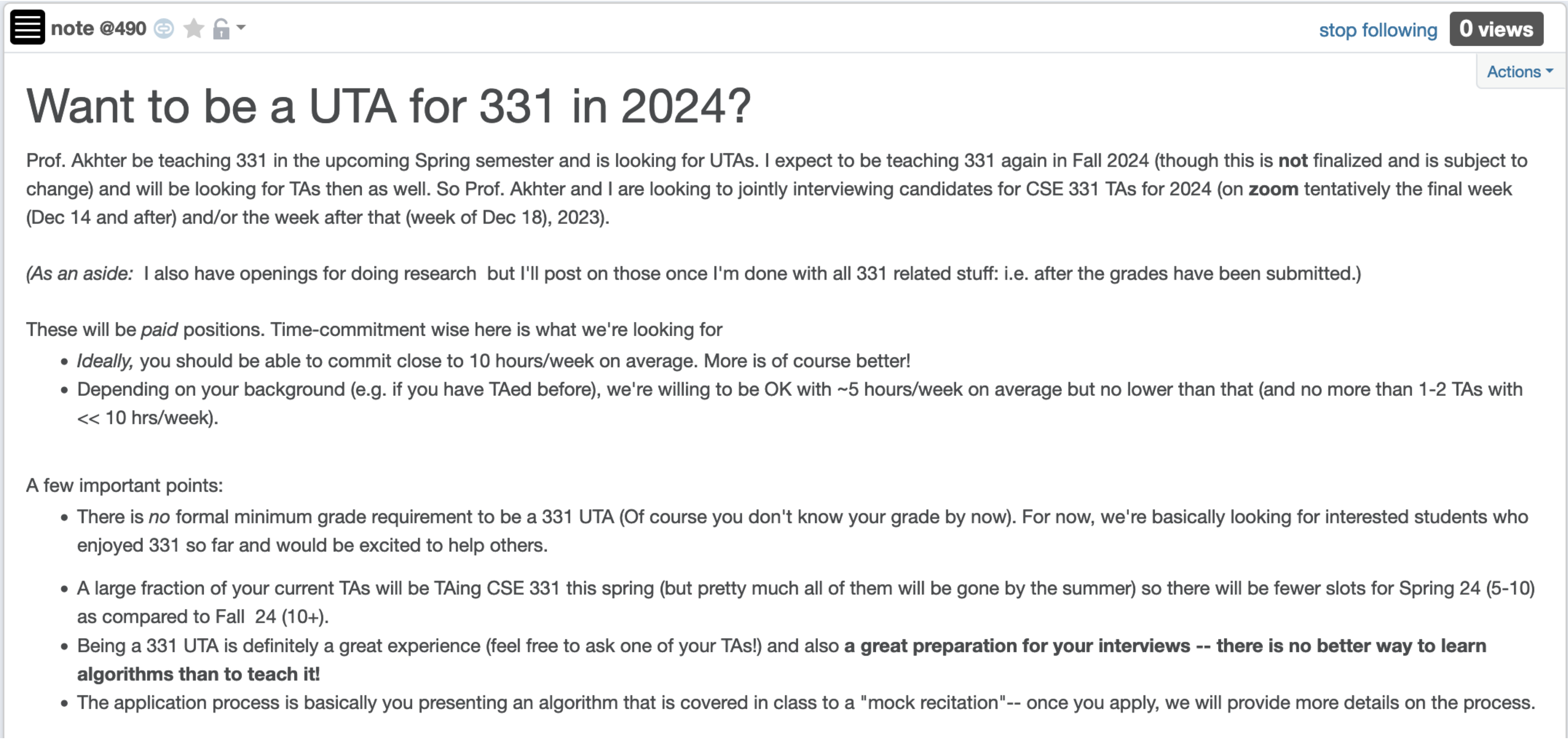
Nov 29, 2023

Next 2 weeks are brutal



Tue, Nov 28		(**HW 8 out, HW 7 in**)
Wed, Nov 29	More on reductions, P and NP ▶F22 ▶F21 ▶F19	[KT, Sec 8.1]
Fri, Dec 1	NP-Completeness ▶F22 ▶F21 ▶F19	[KT, Sec 8.3, 8.4] (**Project (Problem 3 `Coding`) in**)
Mon, Dec 4	The SAT problem ▶F22 ▶F21 ▶F19	[KT, Sec. 8.2] (**Quiz 2**) (**Project (Problem 3 `Reflection`) in**)
Tue, Dec 5		(**HW 8 in**)
Wed, Dec 6	*k*-coloring problem ▶F22 ▶F21 ▶F19	[KT, Sec 8.7]
Fri, Dec 8	*k*-coloring is NP-complete ▶F22 ▶F21 ▶F19	[KT, Sec 8.7] (**Project (Problems 4 & 5 `Coding`) in**)
Mon, Dec 11	Wrapup	
Tue, Dec 12		(**Project (Problems 4 & 5 `Reflection`) in**) (**Project Survey in**)
Wed, Dec 13	**Final Exam**	(**12:00-2:30pm** in **NSC 201 (usual classroom)**)

Apply to be a CSE 331 TA in 2024!



Want to be a UTA for 331 in 2024?

Prof. Akhter be teaching 331 in the upcoming Spring semester and is looking for UTAs. I expect to be teaching 331 again in Fall 2024 (though this is **not** finalized and is subject to change) and will be looking for TAs then as well. So Prof. Akhter and I are looking to jointly interviewing candidates for CSE 331 TAs for 2024 (on **zoom** tentatively the final week (Dec 14 and after) and/or the week after that (week of Dec 18), 2023).

(As an aside: I also have openings for doing research but I'll post on those once I'm done with all 331 related stuff: i.e. after the grades have been submitted.)

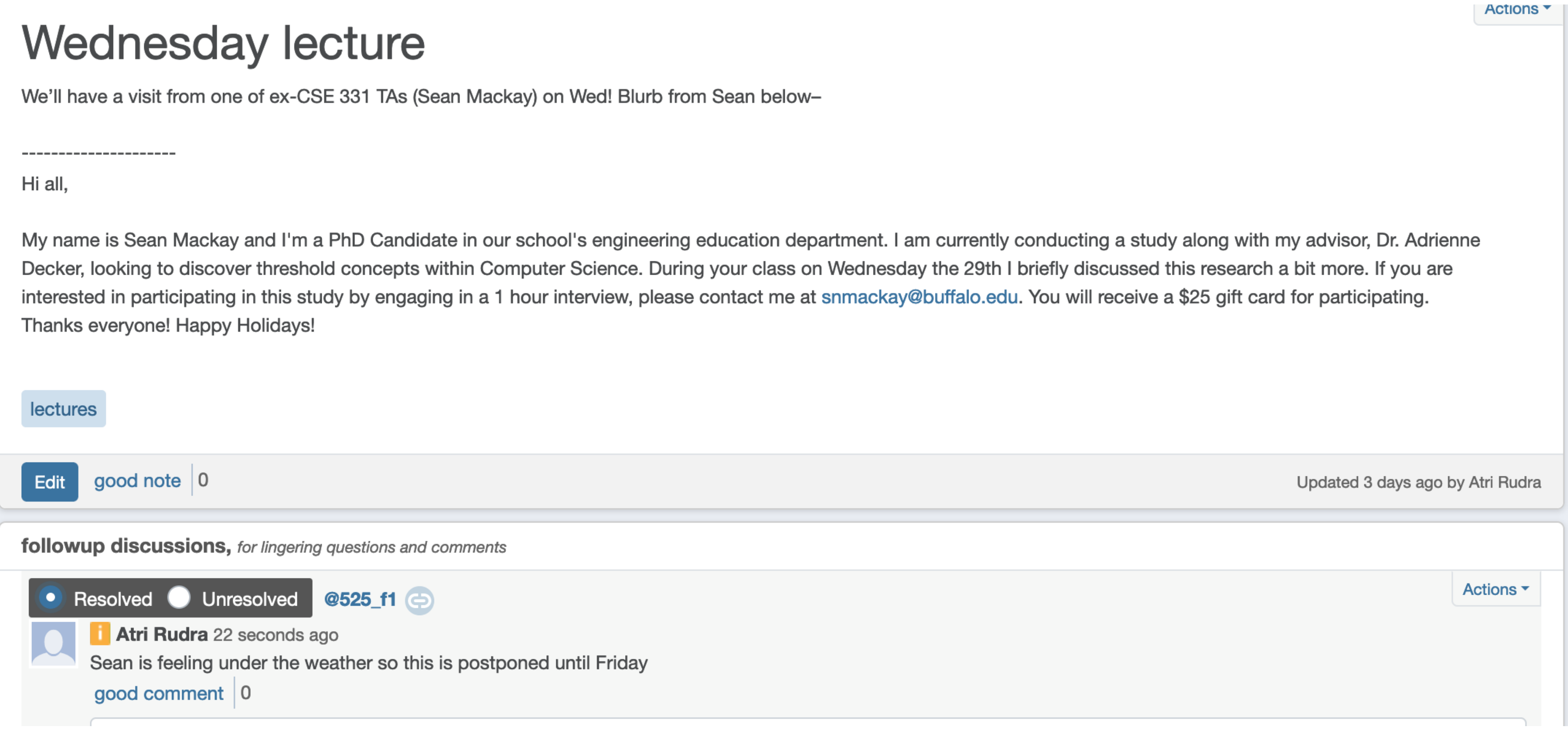
These will be *paid* positions. Time-commitment wise here is what we're looking for

- *Ideally,* you should be able to commit close to 10 hours/week on average. More is of course better!
- Depending on your background (e.g. if you have TAed before), we're willing to be OK with ~5 hours/week on average but no lower than that (and no more than 1-2 TAs with << 10 hrs/week).

A few important points:

- There is *no* formal minimum grade requirement to be a 331 UTA (Of course you don't know your grade by now). For now, we're basically looking for interested students who enjoyed 331 so far and would be excited to help others.
- A large fraction of your current TAs will be TAing CSE 331 this spring (but pretty much all of them will be gone by the summer) so there will be fewer slots for Spring 24 (5-10) as compared to Fall 24 (10+).
- Being a 331 UTA is definitely a great experience (feel free to ask one of your TAs!) and also **a great preparation for your interviews -- there is no better way to learn algorithms than to teach it!**
- The application process is basically you presenting an algorithm that is covered in class to a "mock recitation"-- once you apply, we will provide more details on the process.

Sean will here Friday



Actions

Wednesday lecture

We'll have a visit from one of ex-CSE 331 TAs (Sean Mackay) on Wed! Blurb from Sean below–

Hi all,

My name is Sean Mackay and I'm a PhD Candidate in our school's engineering education department. I am currently conducting a study along with my advisor, Dr. Adrienne Decker, looking to discover threshold concepts within Computer Science. During your class on Wednesday the 29th I briefly discussed this research a bit more. If you are interested in participating in this study by engaging in a 1 hour interview, please contact me at snmackay@buffalo.edu. You will receive a $25 gift card for participating. Thanks everyone! Happy Holidays!

lectures

Edit | good note | 0 — Updated 3 days ago by Atri Rudra

followup discussions, *for lingering questions and comments*

[✓ Resolved] Unresolved @525_f1 — Actions

Atri Rudra 22 seconds ago

Sean is feeling under the weather so this is postponed until Friday

good comment | 0

The course so far…

https://www.teepublic.com/sticker/1100935-obama-yes-we-can

The rest of the course…

https://www.madduckposters.com/products/megamind-no-you-cant?variant=13565168320556

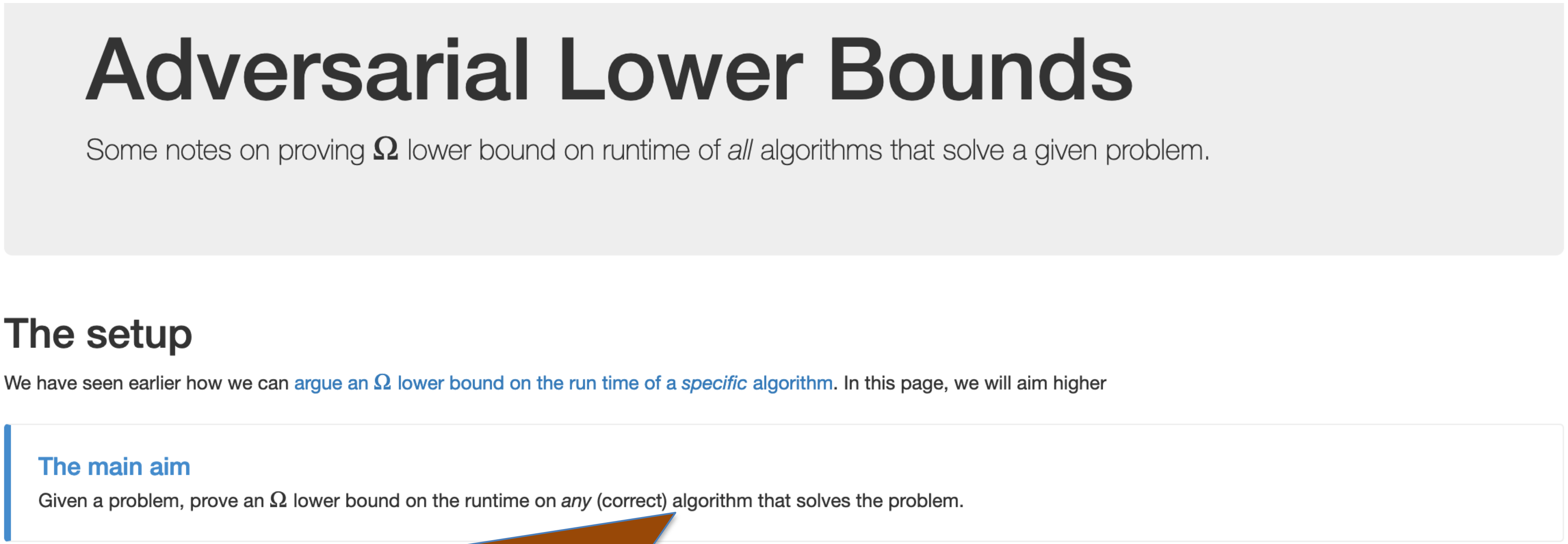
Questions?

No, you can't– what does it mean?

NO algorithm will be able to solve a problem in polynomial time

Still for worst-case runtime

No, you can't take- 1



Adversarial Lower Bounds

Some notes on proving Ω lower bound on runtime of *all* algorithms that solve a given problem.

The setup

We have seen earlier how we can argue an Ω lower bound on the run time of a *specific* algorithm. In this page, we will aim higher

The main aim

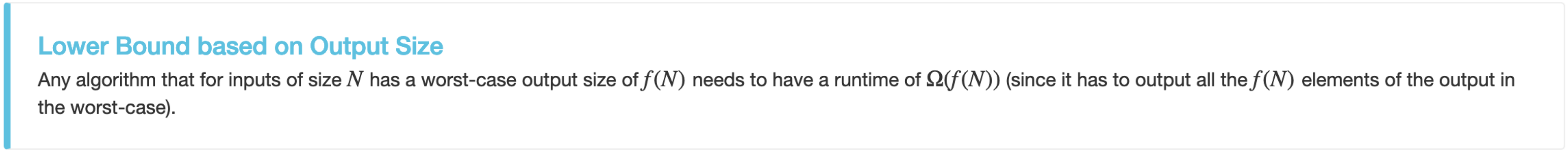
Given a problem, prove an Ω lower bound on the runtime on *any* (correct) algorithm that solves the problem.

What is the best lower bound you can prove?

Ω(N)

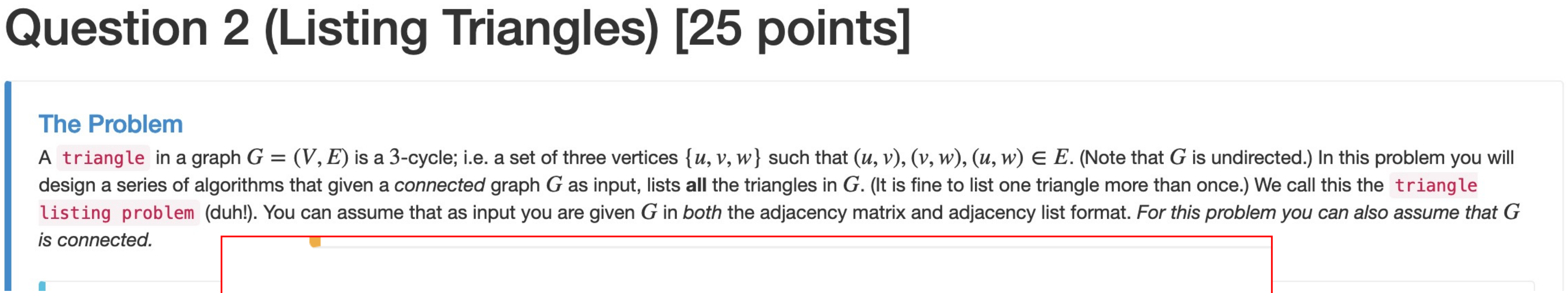
No, you can't take- 2

Lower bounds based on output size



Lower Bound based on Output Size

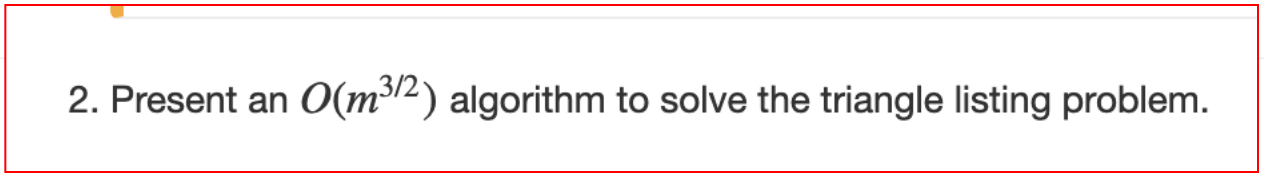
Any algorithm that for inputs of size N has a worst-case output size of $f(N)$ needs to have a runtime of $\Omega(f(N))$ (since it has to output all the $f(N)$ elements of the output in the worst-case).



Question 2 (Listing Triangles) [25 points]

The Problem

A `triangle` in a graph $G = (V, E)$ is a 3-cycle; i.e. a set of three vertices $\{u, v, w\}$ such that $(u, v), (v, w), (u, w) \in E$. (Note that G is undirected.) In this problem you will design a series of algorithms that given a *connected* graph G as input, lists **all** the triangles in G. (It is fine to list one triangle more than once.) We call this the `triangle listing problem` (duh!). You can assume that as input you are given G in *both* the adjacency matrix and adjacency list format. *For this problem you can also assume that G is connected.*



2. Present an $O(m^{3/2})$ algorithm to solve the triangle listing problem.

Exists graphs with $m^{3/2}$ triangles

No, you can't take- 2

Lower bounds based on output size

On input n, output 2^n many ones

Every algo takes (doubly) exponential time

But at heart problem is "trivial"

From now on, output size is always O(N) and could even be binary.

No, you can't take -3

Argue that a given problem is AS HARD AS

a "known" hard problem

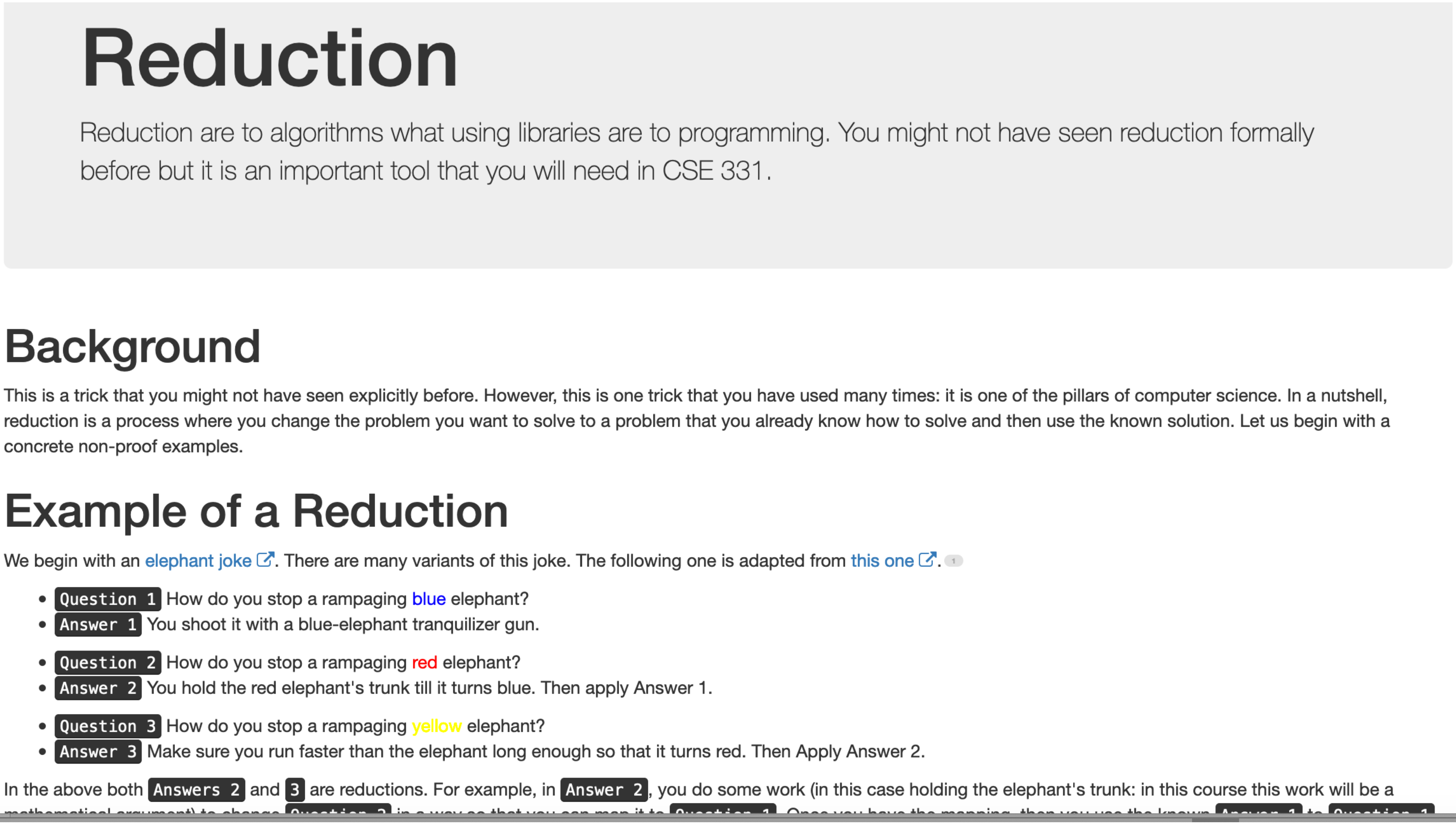
How can we argue something like this?

Reductions

So far: “Yes, we can” reductions

https://www.teepublic.com/sticker/1100935-obama-yes-we-can

Reduce Y to X where X is “easy”



Reduction

Reduction are to algorithms what using libraries are to programming. You might not have seen reduction formally before but it is an important tool that you will need in CSE 331.

Background

This is a trick that you might not have seen explicitly before. However, this is one trick that you have used many times: it is one of the pillars of computer science. In a nutshell, reduction is a process where you change the problem you want to solve to a problem that you already know how to solve and then use the known solution. Let us begin with a concrete non-proof examples.

Example of a Reduction

We begin with an elephant joke. There are many variants of this joke. The following one is adapted from this one. 1

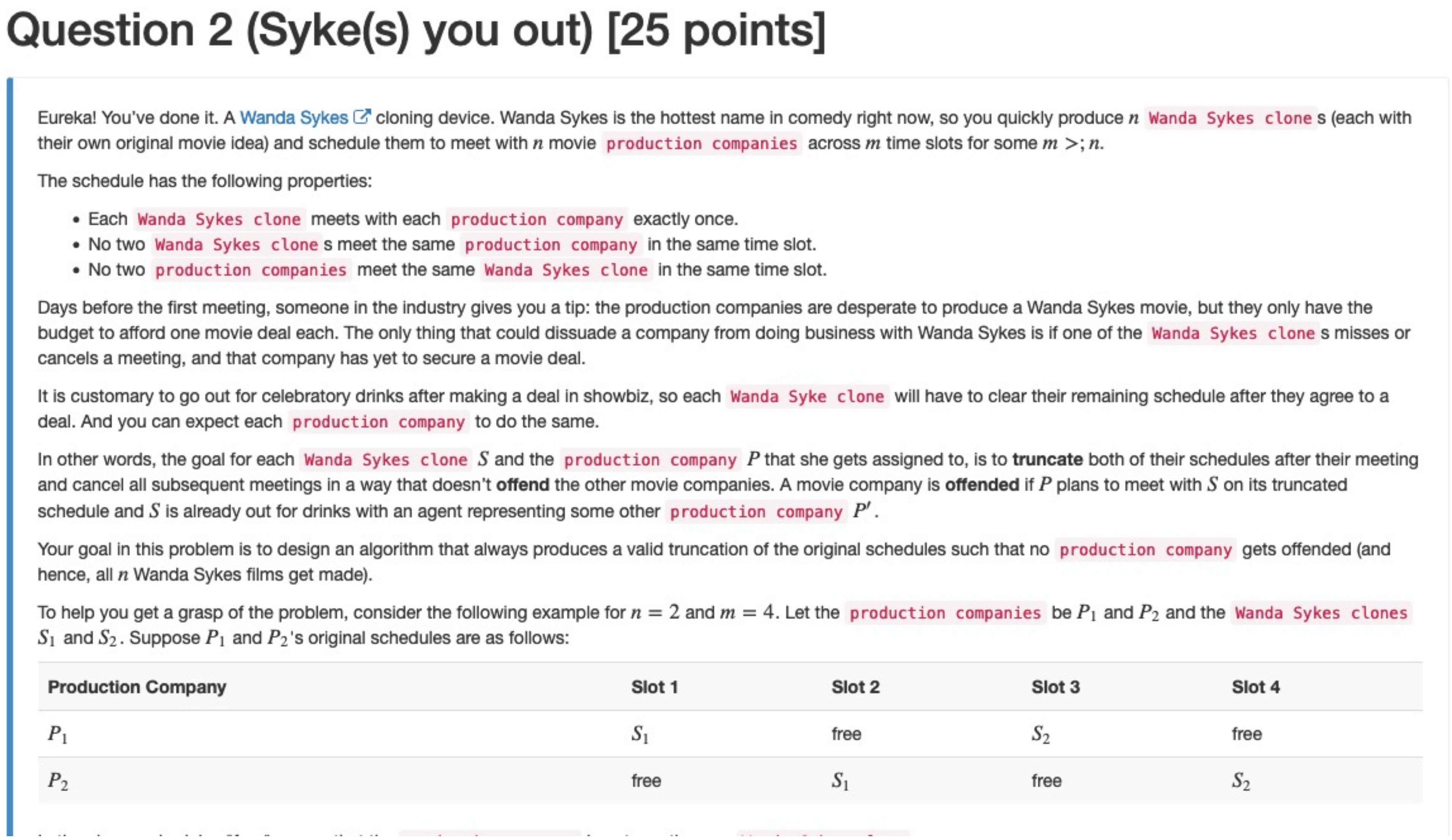
- `Question 1` How do you stop a rampaging blue elephant?
- `Answer 1` You shoot it with a blue-elephant tranquilizer gun.

- `Question 2` How do you stop a rampaging red elephant?
- `Answer 2` You hold the red elephant's trunk till it turns blue. Then apply Answer 1.

- `Question 3` How do you stop a rampaging yellow elephant?
- `Answer 3` Make sure you run faster than the elephant long enough so that it turns red. Then Apply Answer 2.

In the above both `Answers 2` and `3` are reductions. For example, in `Answer 2`, you do some work (in this case holding the elephant's trunk: in this course this work will be a mathematical argument) to change `Question 2` in a way so that you can map it to `Question 1`. Once you have the mapping, then you use the known `Answer 1` to `Question 1`

“Yes, we can” reductions (Example)



Question 2 (Syke(s) you out) [25 points]

Eureka! You've done it. A Wanda Sykes cloning device. Wanda Sykes is the hottest name in comedy right now, so you quickly produce n `Wanda Sykes clone`s (each with their own original movie idea) and schedule them to meet with n movie `production companies` across m time slots for some $m >; n$.

The schedule has the following properties:

- Each `Wanda Sykes clone` meets with each `production company` exactly once.
- No two `Wanda Sykes clone`s meet the same `production company` in the same time slot.
- No two `production companies` meet the same `Wanda Sykes clone` in the same time slot.

Days before the first meeting, someone in the industry gives you a tip: the production companies are desperate to produce a Wanda Sykes movie, but they only have the budget to afford one movie deal each. The only thing that could dissuade a company from doing business with Wanda Sykes is if one of the `Wanda Sykes clone`s misses or cancels a meeting, and that company has yet to secure a movie deal.

It is customary to go out for celebratory drinks after making a deal in showbiz, so each `Wanda Syke clone` will have to clear their remaining schedule after they agree to a deal. And you can expect each `production company` to do the same.

In other words, the goal for each `Wanda Sykes clone` S and the `production company` P that she gets assigned to, is to **truncate** both of their schedules after their meeting and cancel all subsequent meetings in a way that doesn't **offend** the other movie companies. A movie company is **offended** if P plans to meet with S on its truncated schedule and S is already out for drinks with an agent representing some other `production company` P'.

Your goal in this problem is to design an algorithm that always produces a valid truncation of the original schedules such that no `production company` gets offended (and hence, all n Wanda Sykes films get made).

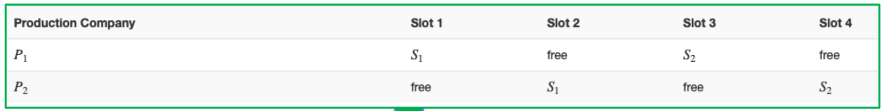
To help you get a grasp of the problem, consider the following example for $n = 2$ and $m = 4$. Let the `production companies` be P_1 and P_2 and the `Wanda Sykes clones` S_1 and S_2. Suppose P_1 and P_2's original schedules are as follows:

Production Company	Slot 1	Slot 2	Slot 3	Slot 4
P_1	S_1	free	S_2	free
P_2	free	S_1	free	S_2

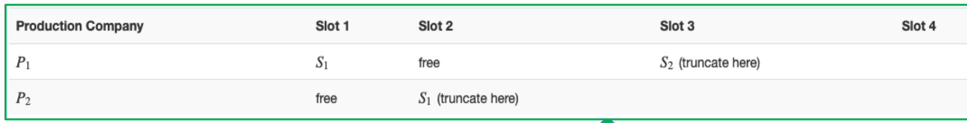
Overview of the reduction

Question 2 (Syke(s) you out)

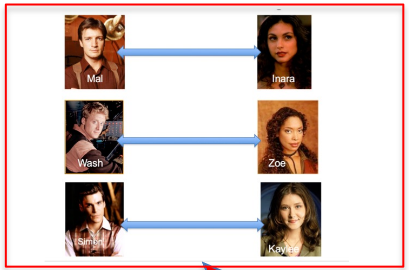
NRMP
National Resident Matching Program

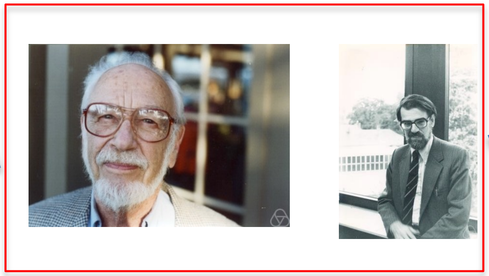


Production Company	Slot 1	Slot 2	Slot 3	Slot 4
P_1	S_1	free	S_2	free
P_2	free	S_1	free	S_2



Production Company	Slot 1	Slot 2	Slot 3	Slot 4
P_1	S_1	free	S_2 (truncate here)	
P_2	free	S_1 (truncate here)		

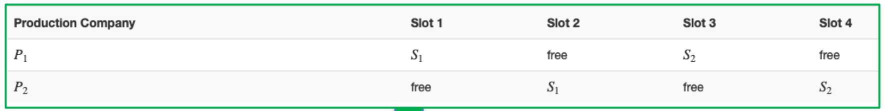




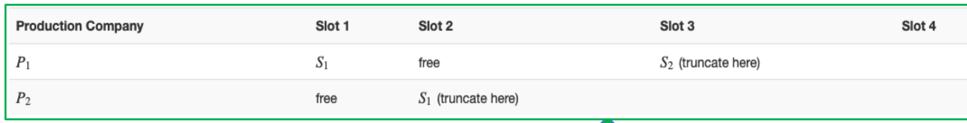
Nothing special about GS algo

Question 2 (Syke(s) you out)

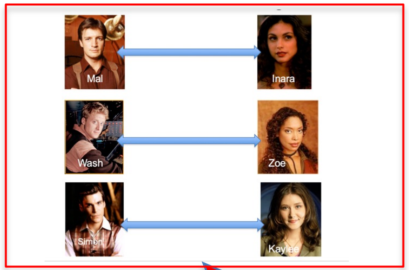
NRMP
National Resident Matching Program



Production Company	Slot 1	Slot 2	Slot 3	Slot 4
P_1	S_1	free	S_2	free
P_2	free	S_1	free	S_2



Production Company	Slot 1	Slot 2	Slot 3	Slot 4
P_1	S_1	free	S_2 (truncate here)	
P_2	free	S_1 (truncate here)		

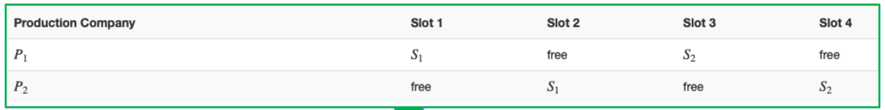


ANY algo for stable matching problem works!

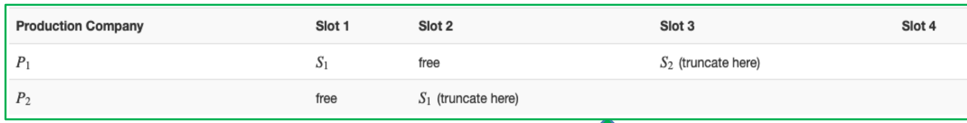
Another observation

Question 2 (Syke(s) you out)

NRMP
National Resident Matching Program

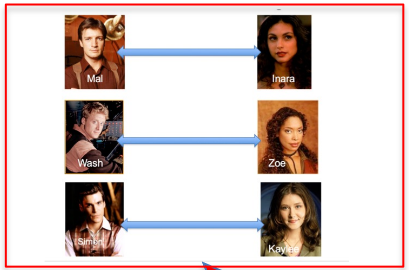


Production Company	Slot 1	Slot 2	Slot 3	Slot 4
P_1	S_1	free	S_2	free
P_2	free	S_1	free	S_2



Production Company	Slot 1	Slot 2	Slot 3	Slot 4
P_1	S_1	free	S_2 (truncate here)	
P_2	free	S_1 (truncate here)		

Poly time steps

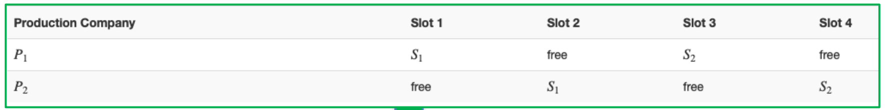


ANY algo for stable matching problem works!

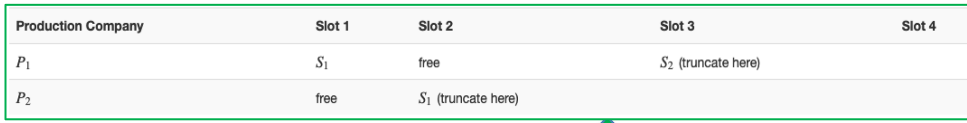
Poly time reductions

Question 2 (Syke(s) you out)

NRMP
National Resident Matching Program

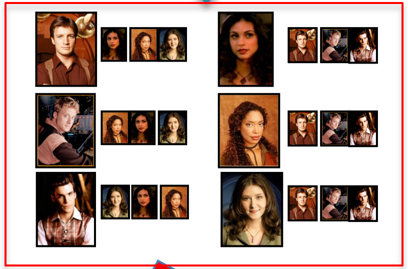


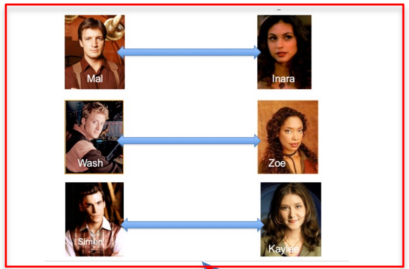
Production Company	Slot 1	Slot 2	Slot 3	Slot 4
P_1	S_1	free	S_2	free
P_2	free	S_1	free	S_2



Production Company	Slot 1	Slot 2	Slot 3	Slot 4
P_1	S_1	free	S_2 (truncate here)	
P_2	free	S_1 (truncate here)		

Poly time steps



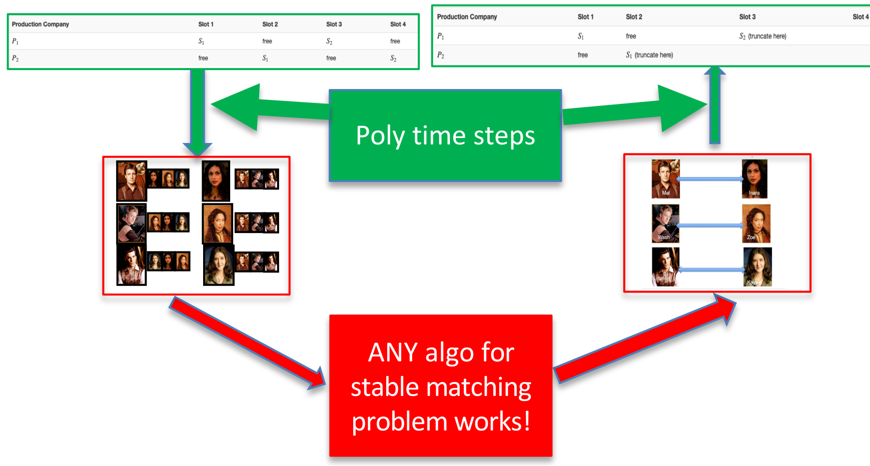


ANY algo for stable matching problem works!

Question 2 (Syke(s) you out)

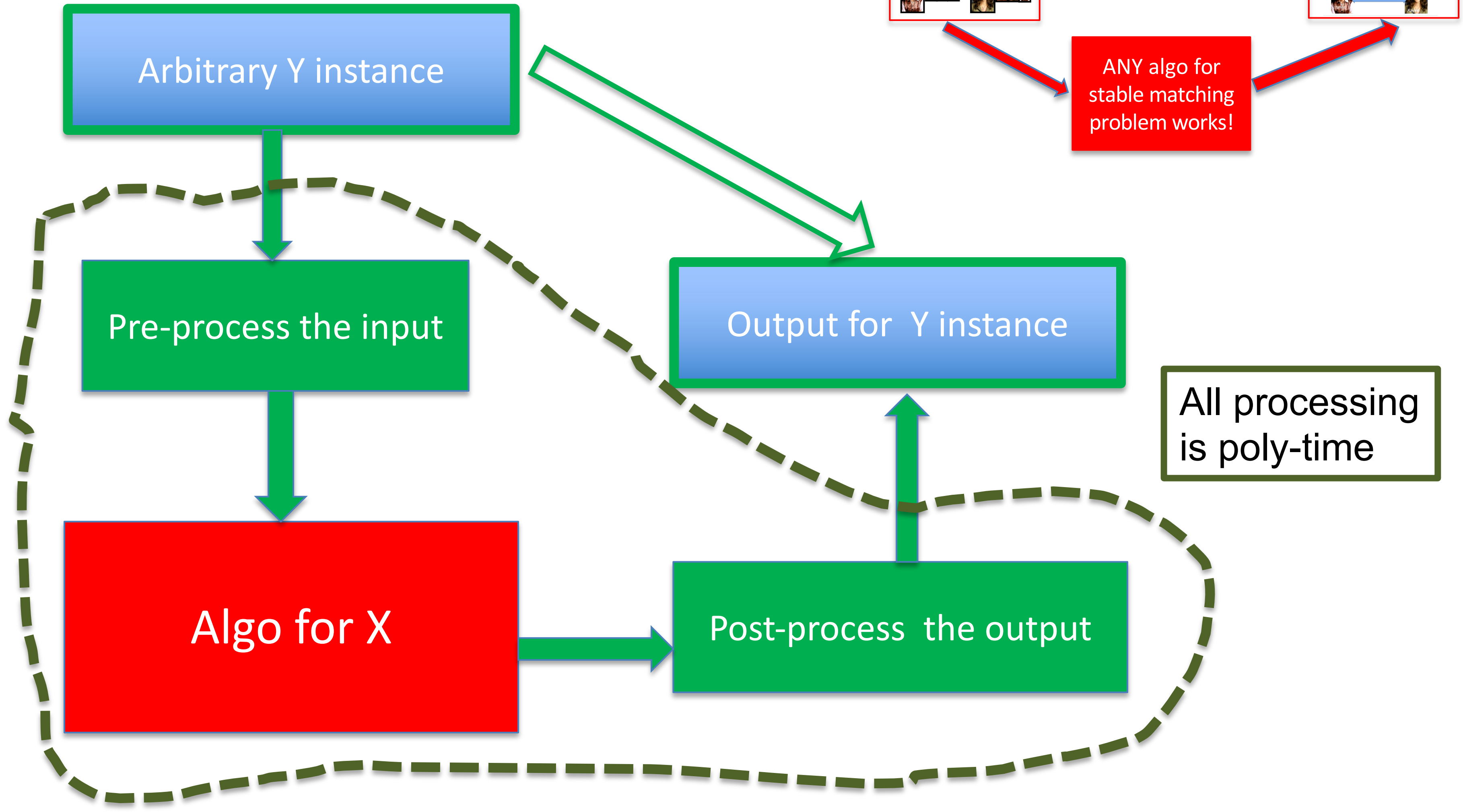
NRMP
National Resident Matching Program

$$Y \leq_P X$$

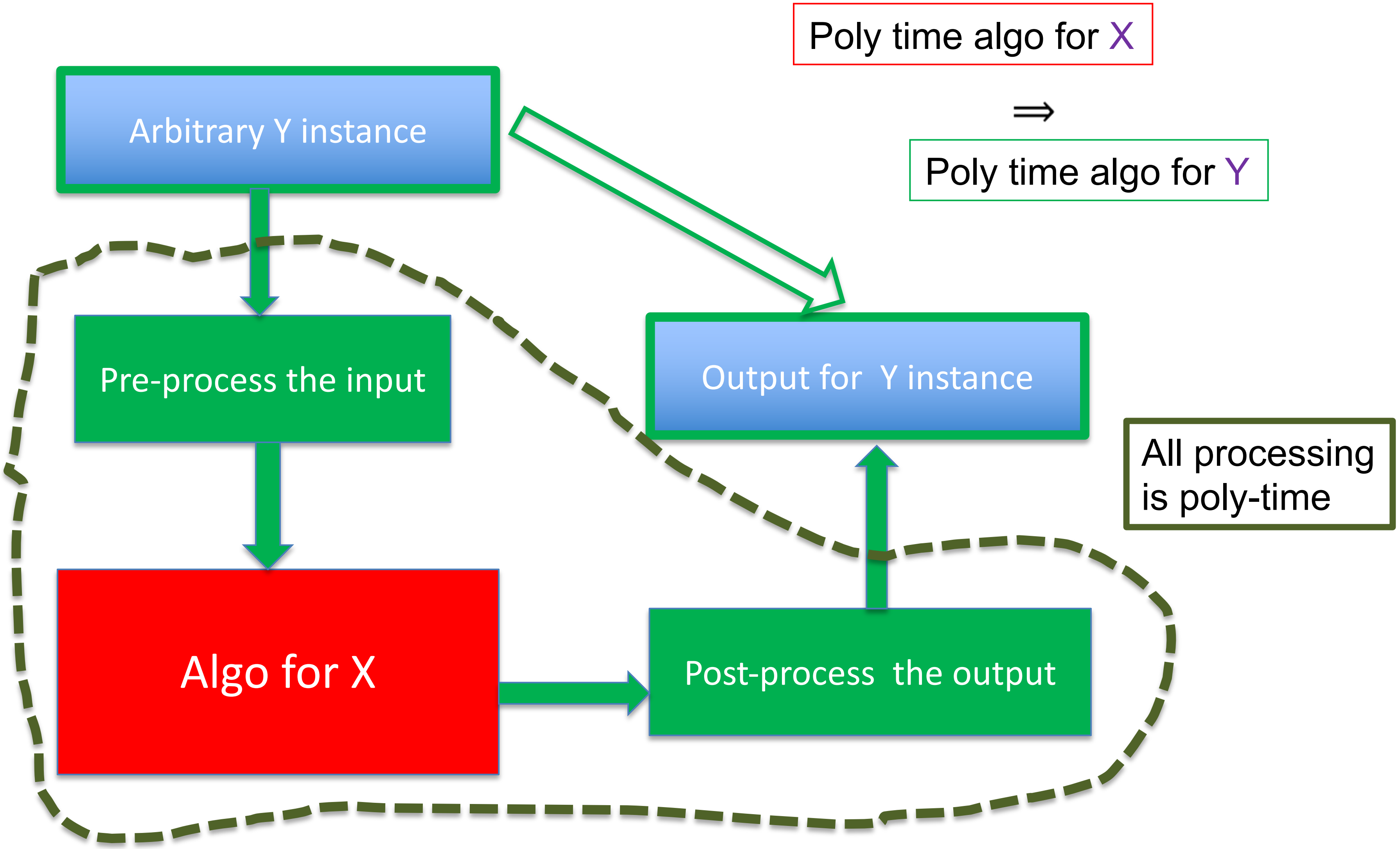


Production Company	Slot 1	Slot 2	Slot 3	Slot 4
P_1	S_1	free	S_2	free
P_2	free	S_1	free	S_2

Production Company	Slot 1	Slot 2	Slot 3	Slot 4
P_1	S_1	free	S_2 (truncate here)	
P_2	free	S_1 (truncate here)		



Implications of $Y \leq_p X$



$$A \Longrightarrow B$$

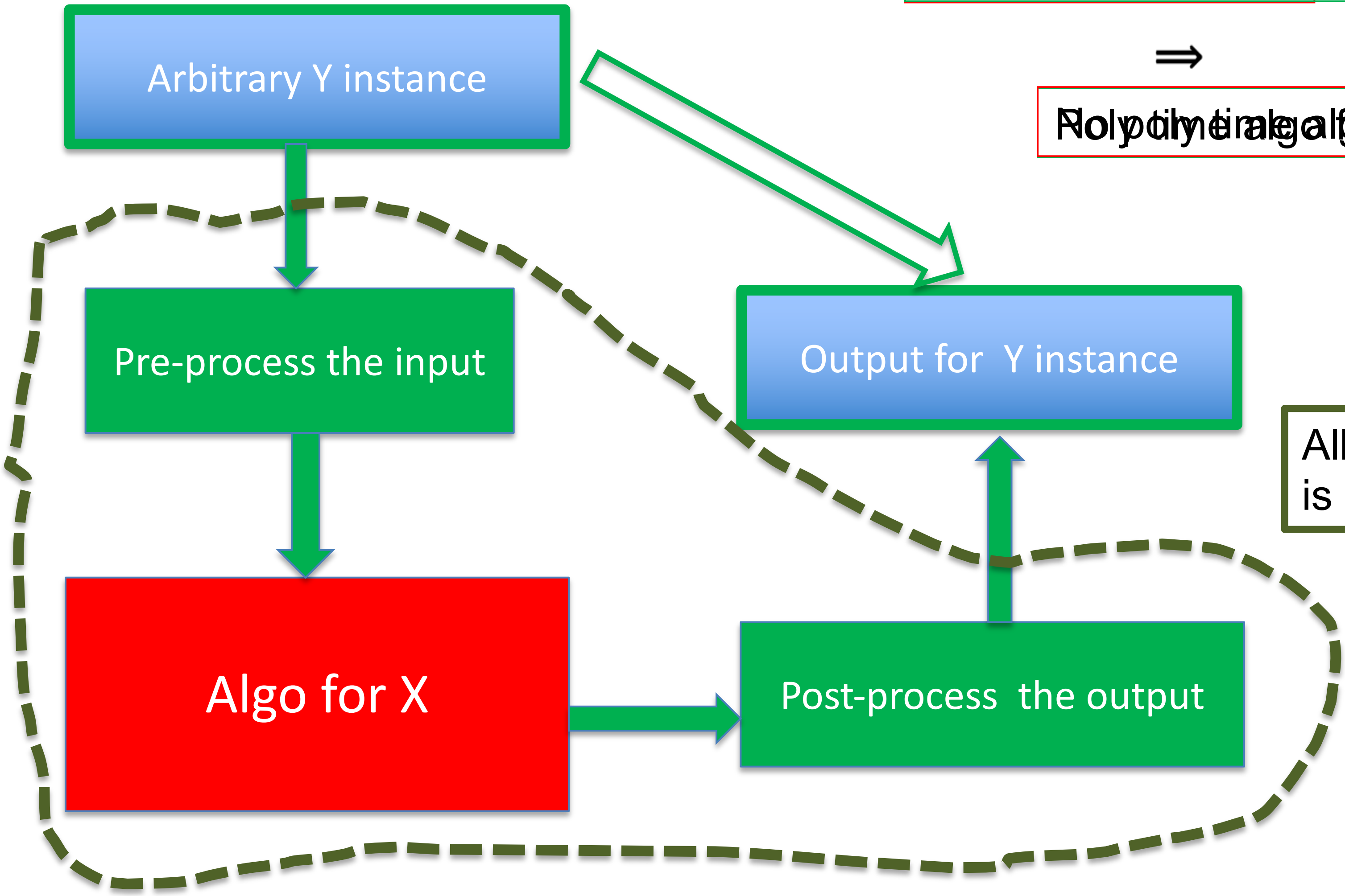
$$!B \Longrightarrow !A$$

Implications of Y $\leq_P$ X

Polytime algo for X
No polytime algo for Y

$\Rightarrow$

Polytime algo for Y
No polytime algo for X



All processing is poly-time

Plan for rest of today

More on reductions

(If there is time) Quick definition of P and NP