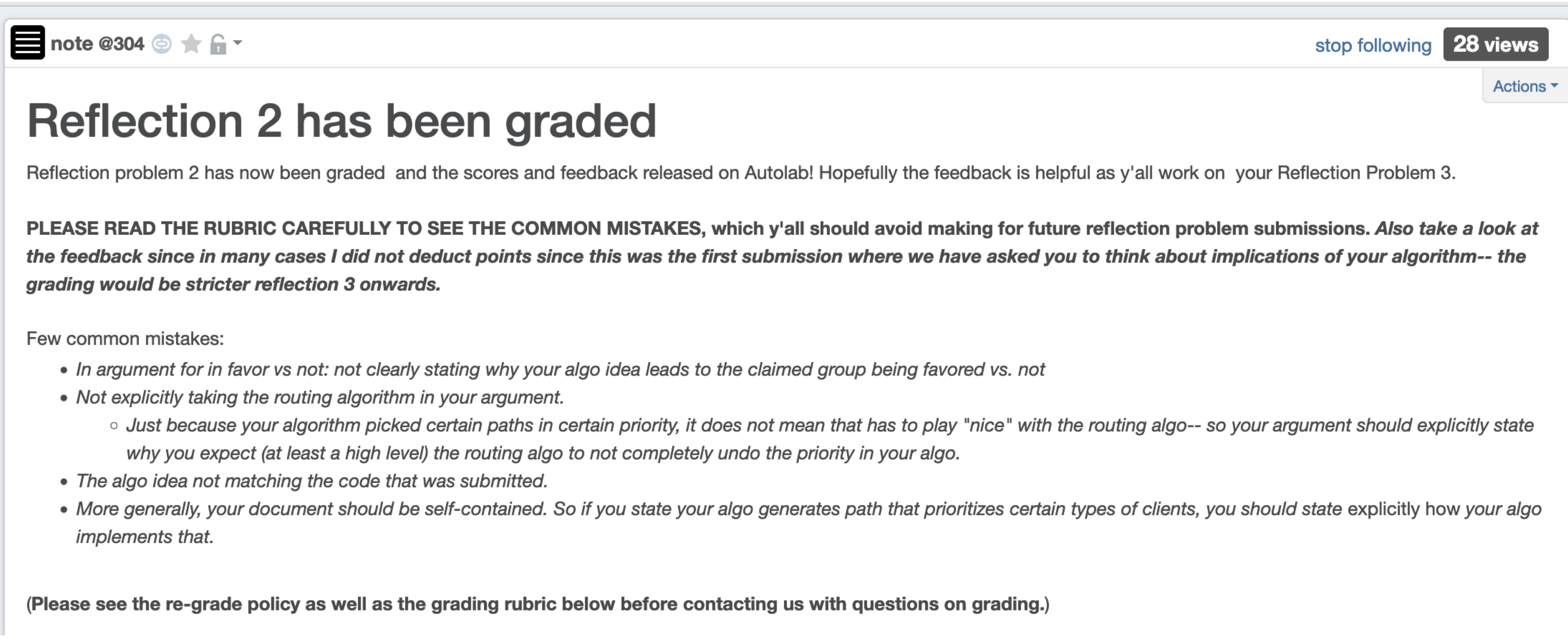
Lecture 35

CSE 331

Nov 22, 2024

Reflection 2 has been graded


note @304 stop following 28 views Actions

Reflection 2 has been graded

Reflection problem 2 has now been graded and the scores and feedback released on Autolab! Hopefully the feedback is helpful as y'all work on your Reflection Problem 3.

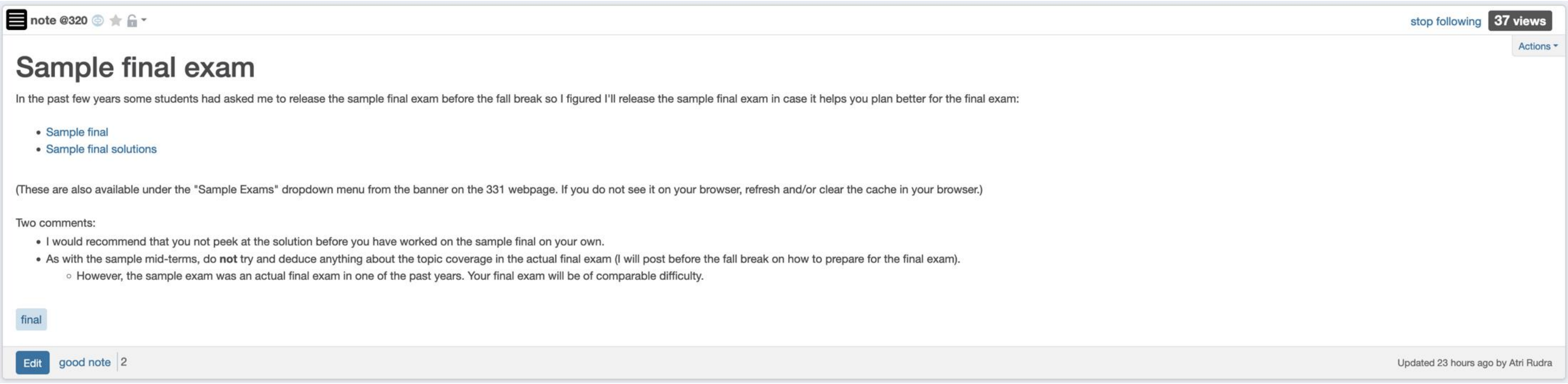
PLEASE READ THE RUBRIC CAREFULLY TO SEE THE COMMON MISTAKES, which y'all should avoid making for future reflection problem submissions. *Also take a look at the feedback since in many cases I did not deduct points since this was the first submission where we have asked you to think about implications of your algorithm-- the grading would be stricter reflection 3 onwards.*

Few common mistakes:

- *In argument for in favor vs not: not clearly stating why your algo idea leads to the claimed group being favored vs. not*
- *Not explicitly taking the routing algorithm in your argument.*
  - *Just because your algorithm picked certain paths in certain priority, it does not mean that has to play "nice" with the routing algo-- so your argument should explicitly state why you expect (at least a high level) the routing algo to not completely undo the priority in your algo.*
- *The algo idea not matching the code that was submitted.*
- *More generally, your document should be self-contained. So if you state your algo generates path that prioritizes certain types of clients, you should state* explicitly how *your algo implements that.*

(Please see the re-grade policy as well as the grading rubric below before contacting us with questions on grading.)

Sample final exam



Sample final exam

In the past few years some students had asked me to release the sample final exam before the fall break so I figured I'll release the sample final exam in case it helps you plan better for the final exam:

- Sample final
- Sample final solutions

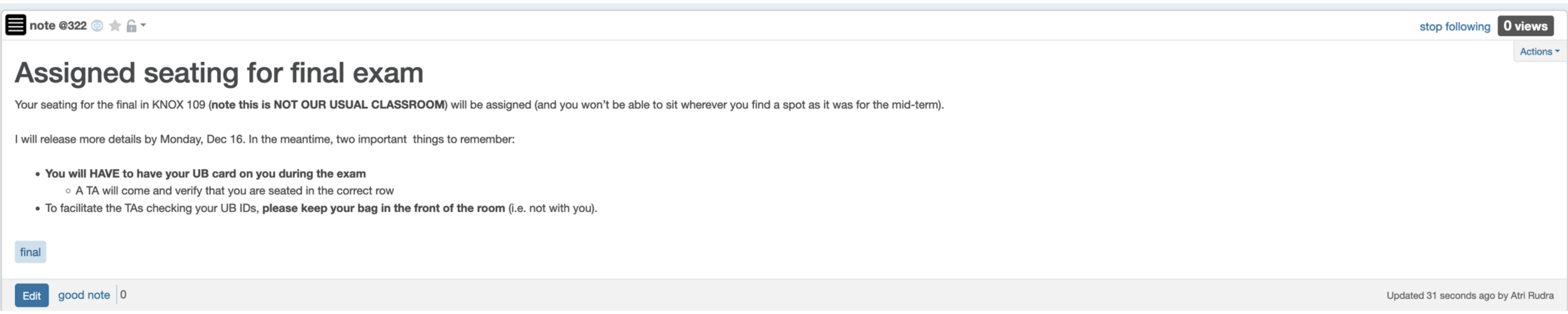
(These are also available under the "Sample Exams" dropdown menu from the banner on the 331 webpage. If you do not see it on your browser, refresh and/or clear the cache in your browser.)

Two comments:

- I would recommend that you not peek at the solution before you have worked on the sample final on your own.
- As with the sample mid-terms, do **not** try and deduce anything about the topic coverage in the actual final exam (I will post before the fall break on how to prepare for the final exam).
    - However, the sample exam was an actual final exam in one of the past years. Your final exam will be of comparable difficulty.

Updated 23 hours ago by Atri Rudra

Bring UB card to final exam



Assigned seating for final exam

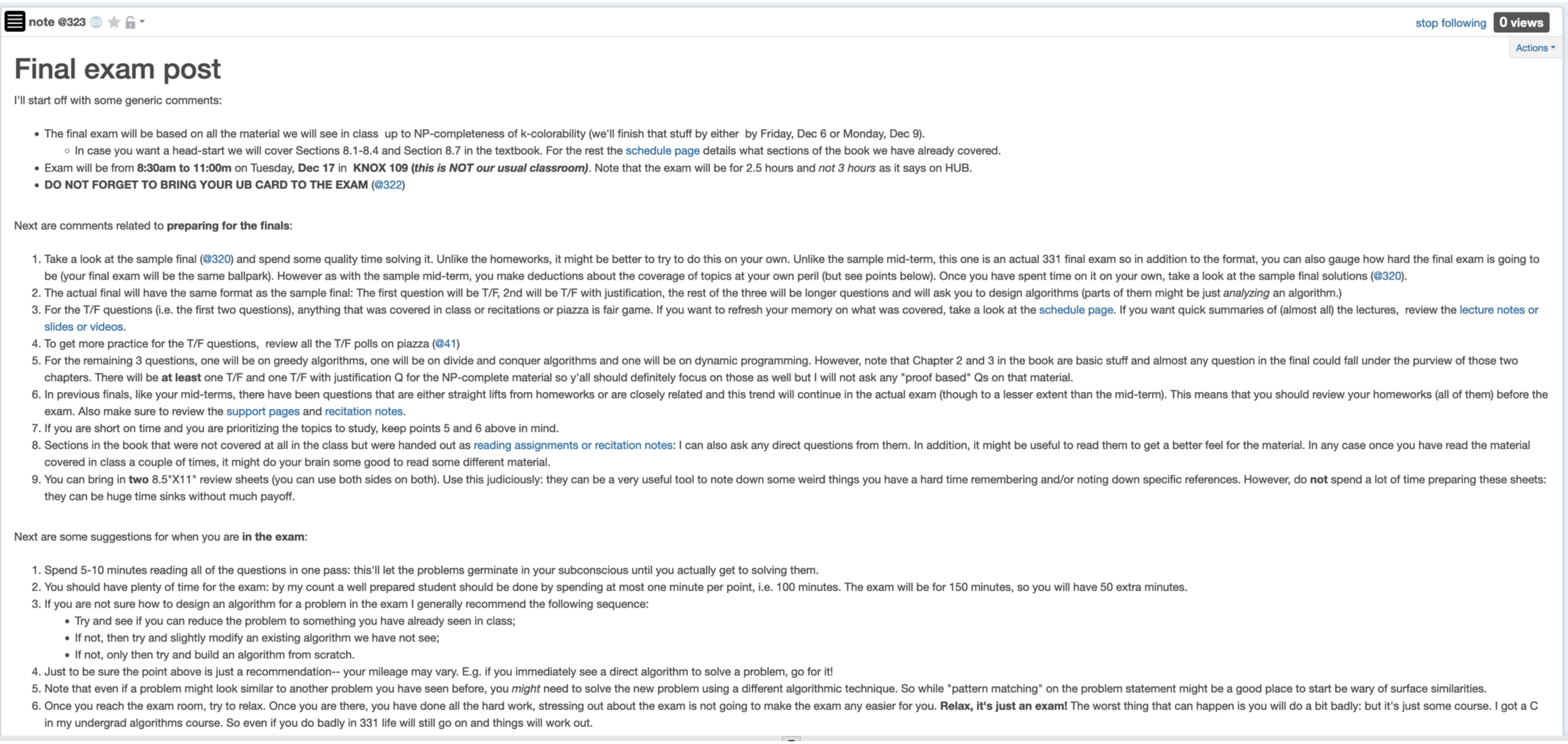
Your seating for the final in KNOX 109 (**note this is NOT OUR USUAL CLASSROOM**) will be assigned (and you won't be able to sit wherever you find a spot as it was for the mid-term).

I will release more details by Monday, Dec 16. In the meantime, two important things to remember:

- **You will HAVE to have your UB card on you during the exam**
  - A TA will come and verify that you are seated in the correct row
- To facilitate the TAs checking your UB IDs, **please keep your bag in the front of the room** (i.e. not with you).

Updated 31 seconds ago by Atri Rudra

Final exam post



note @323

stop following | 0 views

Actions

Final exam post

I'll start off with some generic comments:

- The final exam will be based on all the material we will see in class up to NP-completeness of k-colorability (we'll finish that stuff by either by Friday, Dec 6 or Monday, Dec 9).
    - In case you want a head-start we will cover Sections 8.1-8.4 and Section 8.7 in the textbook. For the rest the schedule page details what sections of the book we have already covered.
- Exam will be from **8:30am to 11:00m** on Tuesday, **Dec 17** in **KNOX 109 (*this is NOT our usual classroom)***. Note that the exam will be for 2.5 hours and *not 3 hours* as it says on HUB.
- **DO NOT FORGET TO BRING YOUR UB CARD TO THE EXAM** (@322)

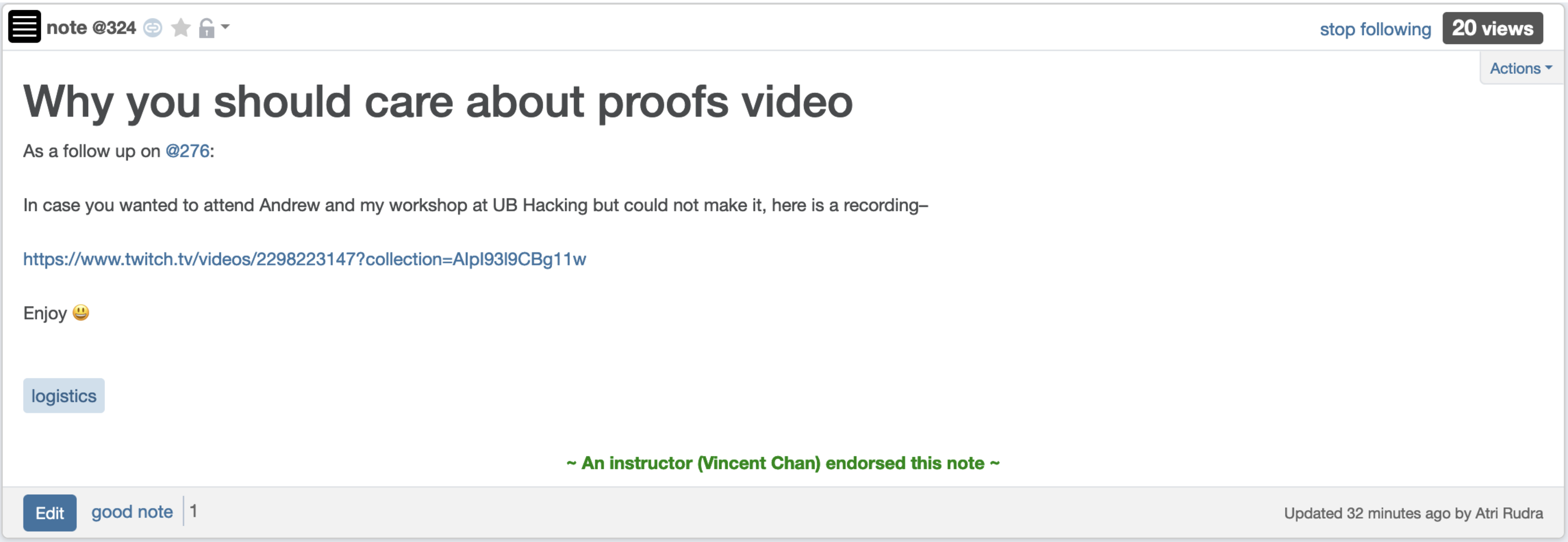
Next are comments related to **preparing for the finals**:

1. Take a look at the sample final (@320) and spend some quality time solving it. Unlike the homeworks, it might be better to try to do this on your own. Unlike the sample mid-term, this one is an actual 331 final exam so in addition to the format, you can also gauge how hard the final exam is going to be (your final exam will be the same ballpark). However as with the sample mid-term, you make deductions about the coverage of topics at your own peril (but see points below). Once you have spent time on it on your own, take a look at the sample final solutions (@320).
2. The actual final will have the same format as the sample final: The first question will be T/F, 2nd will be T/F with justification, the rest of the three will be longer questions and will ask you to design algorithms (parts of them might be just *analyzing* an algorithm.)
3. For the T/F questions (i.e. the first two questions), anything that was covered in class or recitations or piazza is fair game. If you want to refresh your memory on what was covered, take a look at the schedule page. If you want quick summaries of (almost all) the lectures, review the lecture notes or slides or videos.
4. To get more practice for the T/F questions, review all the T/F polls on piazza (@41)
5. For the remaining 3 questions, one will be on greedy algorithms, one will be on divide and conquer algorithms and one will be on dynamic programming. However, note that Chapter 2 and 3 in the book are basic stuff and almost any question in the final could fall under the purview of those two chapters. There will be **at least** one T/F and one T/F with justification Q for the NP-complete material so y'all should definitely focus on those as well but I will not ask any "proof based" Qs on that material.
6. In previous finals, like your mid-terms, there have been questions that are either straight lifts from homeworks or are closely related and this trend will continue in the actual exam (though to a lesser extent than the mid-term). This means that you should review your homeworks (all of them) before the exam. Also make sure to review the support pages and recitation notes.
7. If you are short on time and you are prioritizing the topics to study, keep points 5 and 6 above in mind.
8. Sections in the book that were not covered at all in the class but were handed out as reading assignments or recitation notes: I can also ask any direct questions from them. In addition, it might be useful to read them to get a better feel for the material. In any case once you have read the material covered in class a couple of times, it might do your brain some good to read some different material.
9. You can bring in **two** 8.5"X11" review sheets (you can use both sides on both). Use this judiciously: they can be a very useful tool to note down some weird things you have a hard time remembering and/or noting down specific references. However, do **not** spend a lot of time preparing these sheets: they can be huge time sinks without much payoff.

Next are some suggestions for when you are **in the exam**:

1. Spend 5-10 minutes reading all of the questions in one pass: this'll let the problems germinate in your subconscious until you actually get to solving them.
2. You should have plenty of time for the exam: by my count a well prepared student should be done by spending at most one minute per point, i.e. 100 minutes. The exam will be for 150 minutes, so you will have 50 extra minutes.
3. If you are not sure how to design an algorithm for a problem in the exam I generally recommend the following sequence:
    - Try and see if you can reduce the problem to something you have already seen in class;
    - If not, then try and slightly modify an existing algorithm we have not see;
    - If not, only then try and build an algorithm from scratch.
4. Just to be sure the point above is just a recommendation-- your mileage may vary. E.g. if you immediately see a direct algorithm to solve a problem, go for it!
5. Note that even if a problem might look similar to another problem you have seen before, you *might* need to solve the new problem using a different algorithmic technique. So while "pattern matching" on the problem statement might be a good place to start be wary of surface similarities.
6. Once you reach the exam room, try to relax. Once you are there, you have done all the hard work, stressing out about the exam is not going to make the exam any easier for you. **Relax, it's just an exam!** The worst thing that can happen is you will do a bit badly: but it's just some course. I got a C in my undergrad algorithms course. So even if you do badly in 331 life will still go on and things will work out.

In case you missed it



Why you should care about proofs video

As a follow up on @276:

In case you wanted to attend Andrew and my workshop at UB Hacking but could not make it, here is a recording–

https://www.twitch.tv/videos/2298223147?collection=AlpI93l9CBg11w

Enjoy 😃

logistics

~ An instructor (Vincent Chan) endorsed this note ~

Updated 32 minutes ago by Atri Rudra

Longest path problem

Given G, does there exist a simple path of length n-1 ?

Questions?

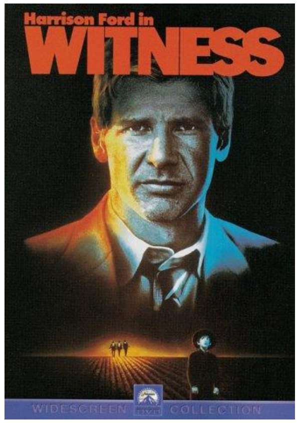
Longest vs Shortest Paths

Two sides of the "same" coin

Shortest Path problem

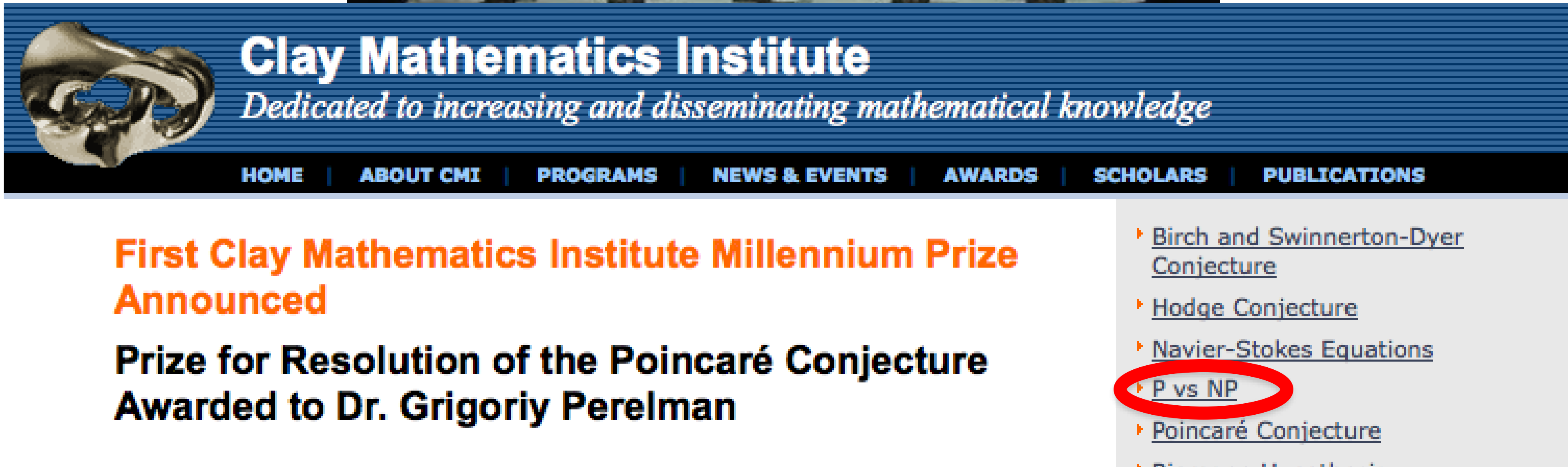
Can be solved by a polynomial time algorithm

Is there a longest path of length n-1?



Given a path can verify in polynomial time if the answer is yes

Poly time algo for longest path?



Clay Mathematics Institute

Dedicated to increasing and disseminating mathematical knowledge

HOME | ABOUT CMI | PROGRAMS | NEWS & EVENTS | AWARDS | SCHOLARS | PUBLICATIONS

First Clay Mathematics Institute Millennium Prize Announced

Prize for Resolution of the Poincaré Conjecture Awarded to Dr. Grigoriy Perelman

- Birch and Swinnerton-Dyer Conjecture
- Hodge Conjecture
- Navier-Stokes Equations
- P vs NP
- Poincaré Conjecture

P vs NP question

P: problems that can be solved by poly time algorithms

Is P=NP?

NP: problems that have polynomial time verifiable witness to optimal solution

Alternate NP definition: Guess witness and verify!

Proving P ≠ NP

Pick any one problem in NP and show it cannot be solved in poly time

Pretty much all known proof techniques *provably* will not work

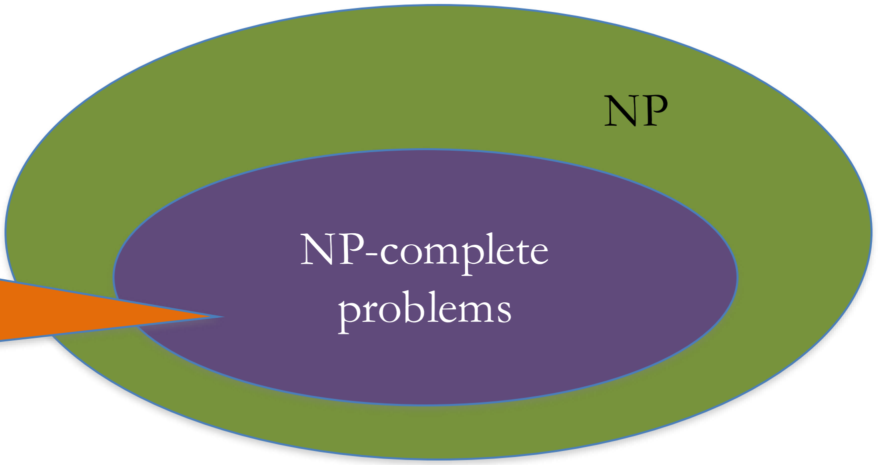
Proving P = NP

Will make cryptography collapse

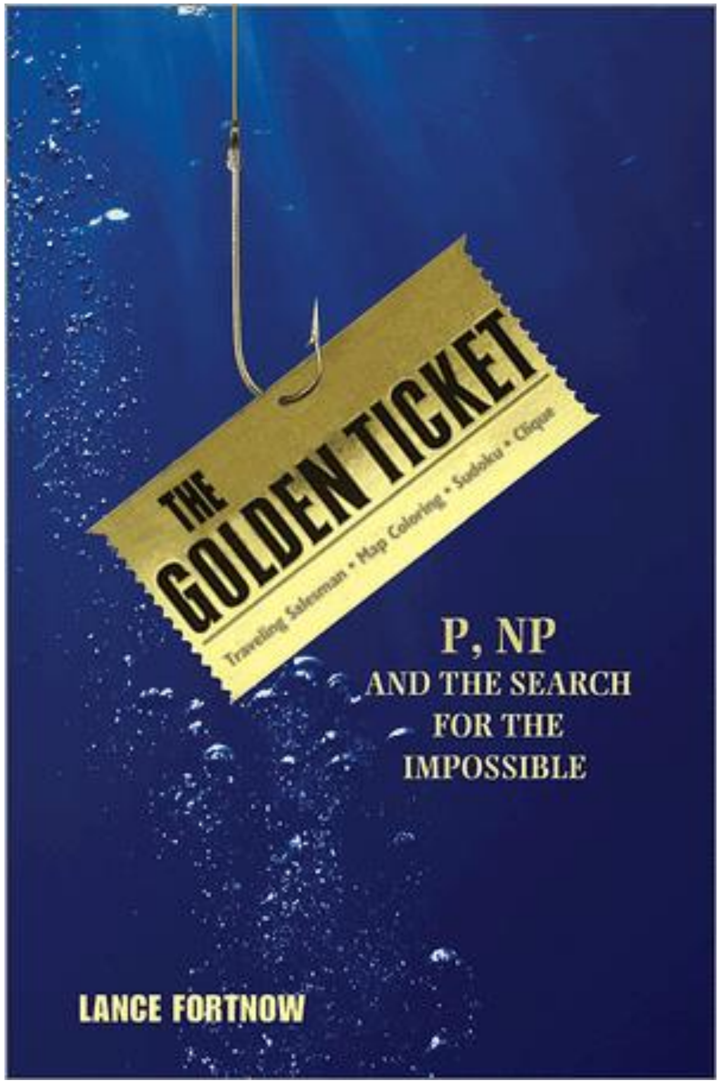
Compute the encryption key!

Prove that all problems in NP can be solved by polynomial time algorithms

Solving any ONE problem in here in poly time will prove P=NP!



A book on P vs. NP


THE GOLDEN TICKET
Traveling Salesman • Map Coloring • Sudoku • Clique
P, NP
AND THE SEARCH FOR THE IMPOSSIBLE
LANCE FORTNOW

Questions?

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

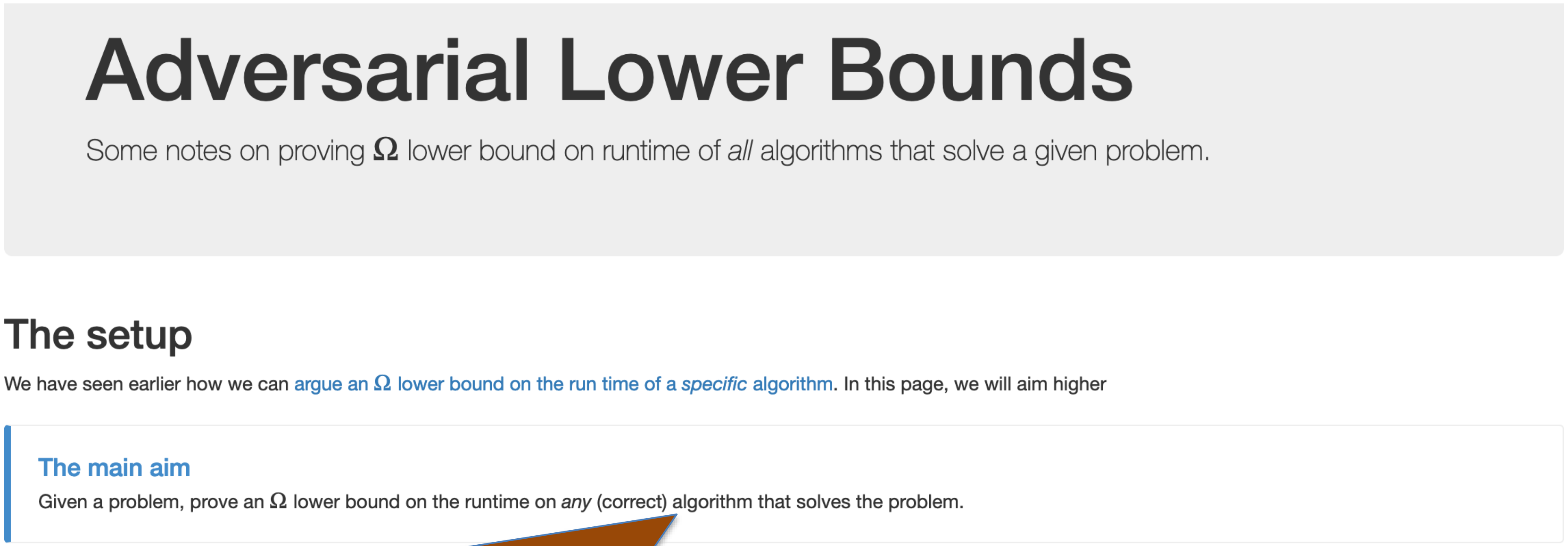
Questions?

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

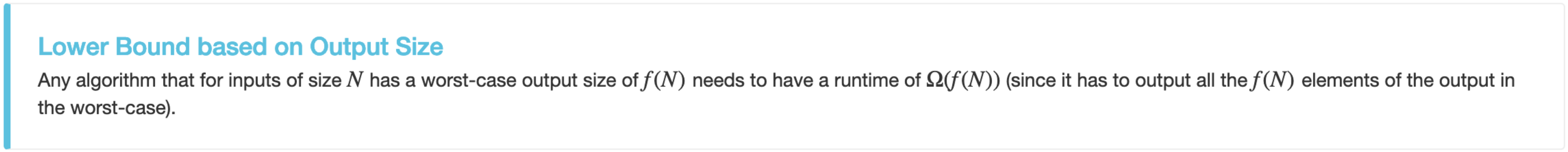
Given a problem, prove an Ω lower bound on the runtime on *any* (correct) algorithm that solves the problem.

What is the best lower bound you can prove?

$\Omega(N)$

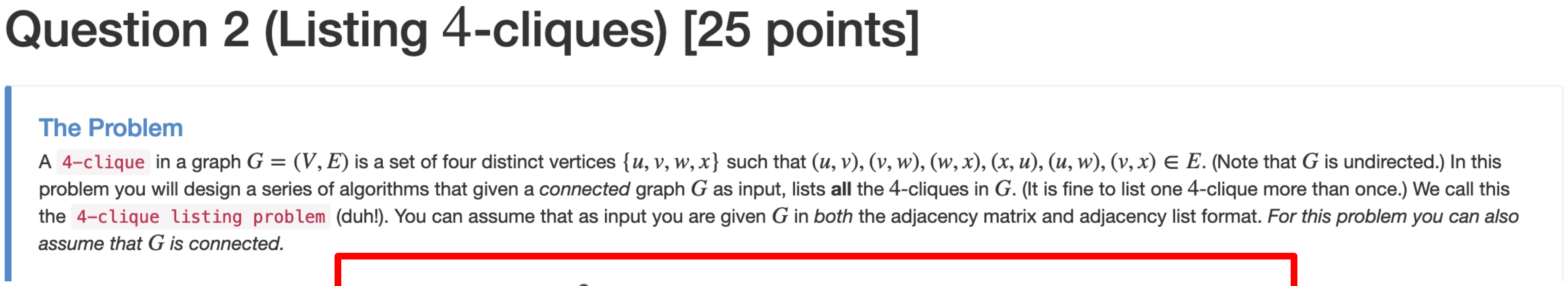
No, you can’t take- 2

Lower bounds based on output size



Lower Bound based on Output Size

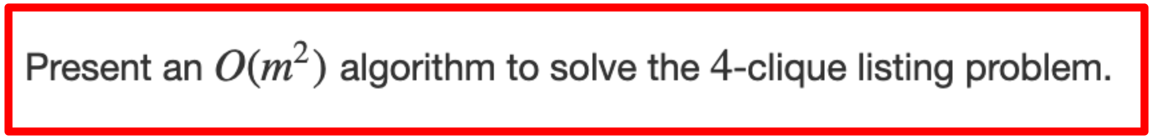
Any algorithm that for inputs of size N has a worst-case output size of $f(N)$ needs to have a runtime of $\Omega(f(N))$ (since it has to output all the $f(N)$ elements of the output in the worst-case).



Question 2 (Listing 4-cliques) [25 points]

The Problem

A `4-clique` in a graph $G = (V, E)$ is a set of four distinct vertices $\{u, v, w, x\}$ such that $(u, v), (v, w), (w, x), (x, u), (u, w), (v, x) \in E$. (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 4-cliques in G. (It is fine to list one 4-clique more than once.) We call this the `4-clique 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.*



Present an $O(m^2)$ algorithm to solve the 4-clique listing problem.

Exists graphs with m^2 4-cliques

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

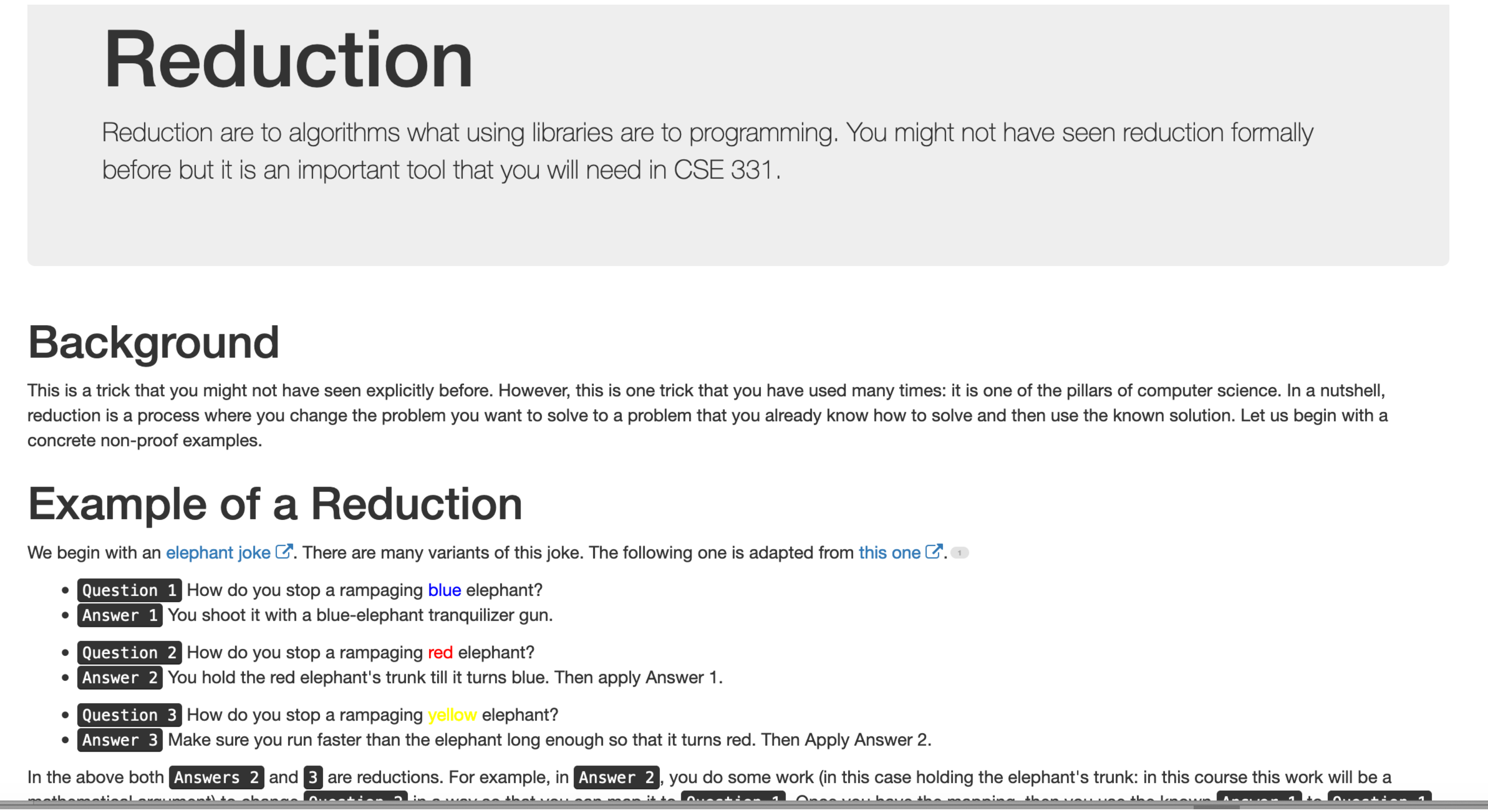
How can we argue something like this?

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

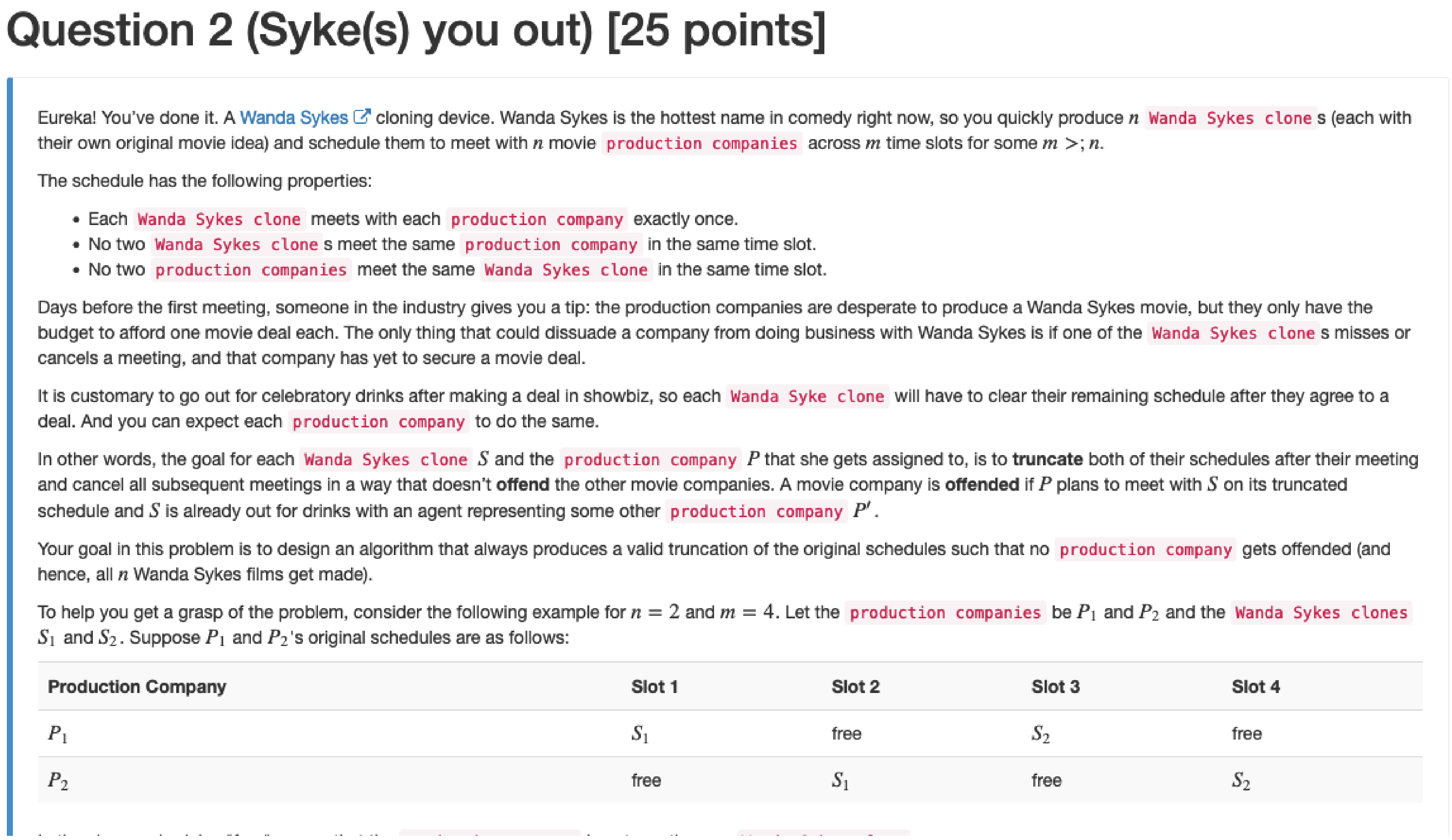
- `Question 1` How do you stop a rampaging blue elephant?
- `Answer 1` You shoot it with a blue-elephant tranquilizer gun.

- `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).

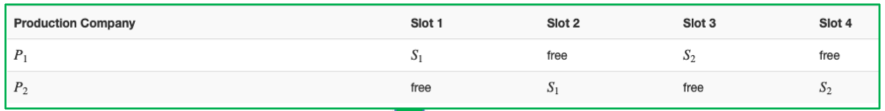
To help you get a grasp of the problem, consider the following example for $n = 2$ and $m = 4$. Let the `production companies` be P_1 and P_2 and the `Wanda Sykes clones` S_1 and S_2. Suppose P_1 and P_2's original schedules are as follows:

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

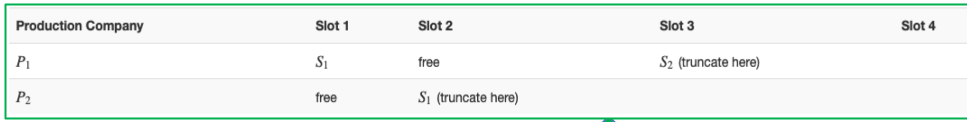
Overview of the reduction

Question 2 (Syke(s) you out)

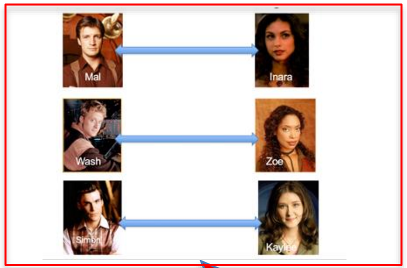
NRMP
National Resident Matching Program

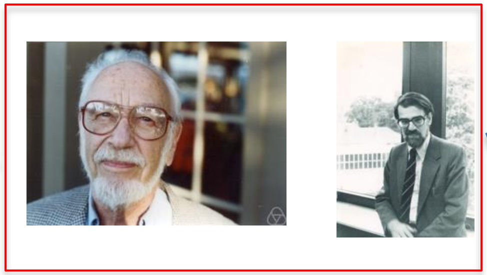


Production Company	Slot 1	Slot 2	Slot 3	Slot 4
P_1	S_1	free	S_2	free
P_2	free	S_1	free	S_2



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)		

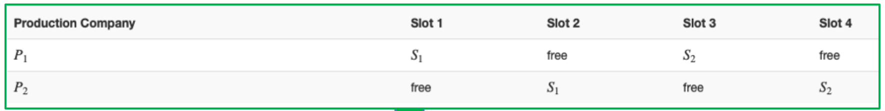




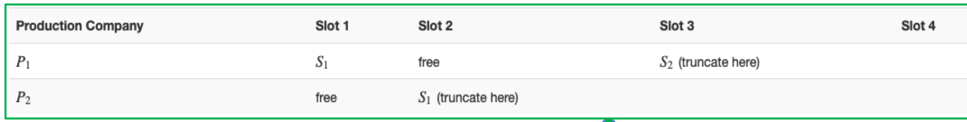
Nothing special about GS algo

Question 2 (Syke(s) you out)

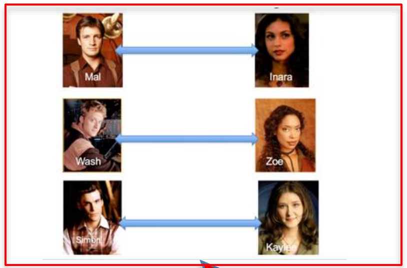
NRMP
National Resident Matching Program



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)		

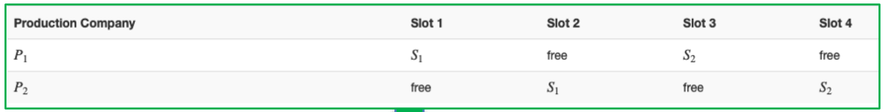


ANY algo for stable matching problem works!

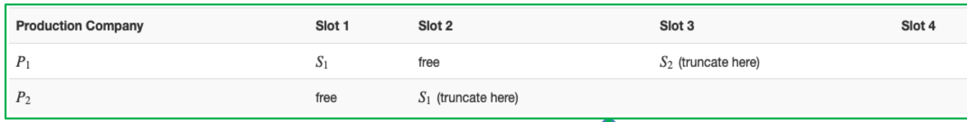
Another observation

Question 2 (Syke(s) you out)

NRMP
National Resident Matching Program

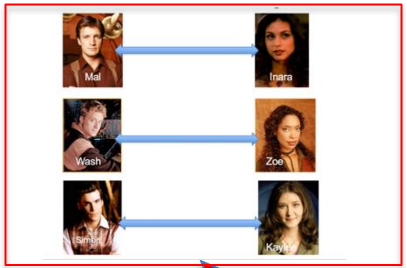


Production Company	Slot 1	Slot 2	Slot 3	Slot 4
P_1	S_1	free	S_2	free
P_2	free	S_1	free	S_2



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)		

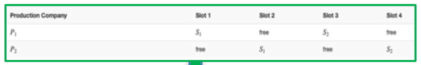
Poly time steps

ANY algo for stable matching problem works!

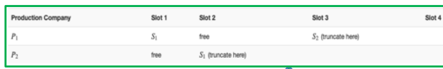
Question 2 (Syke(s) you out)

NRMP
National Resident Matching Program

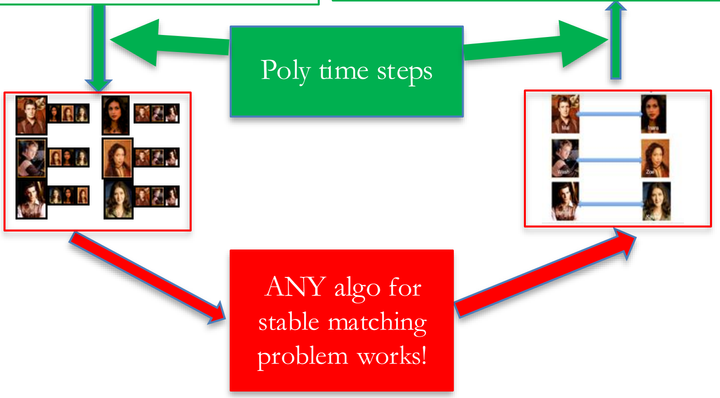
$Y \leq_P X$

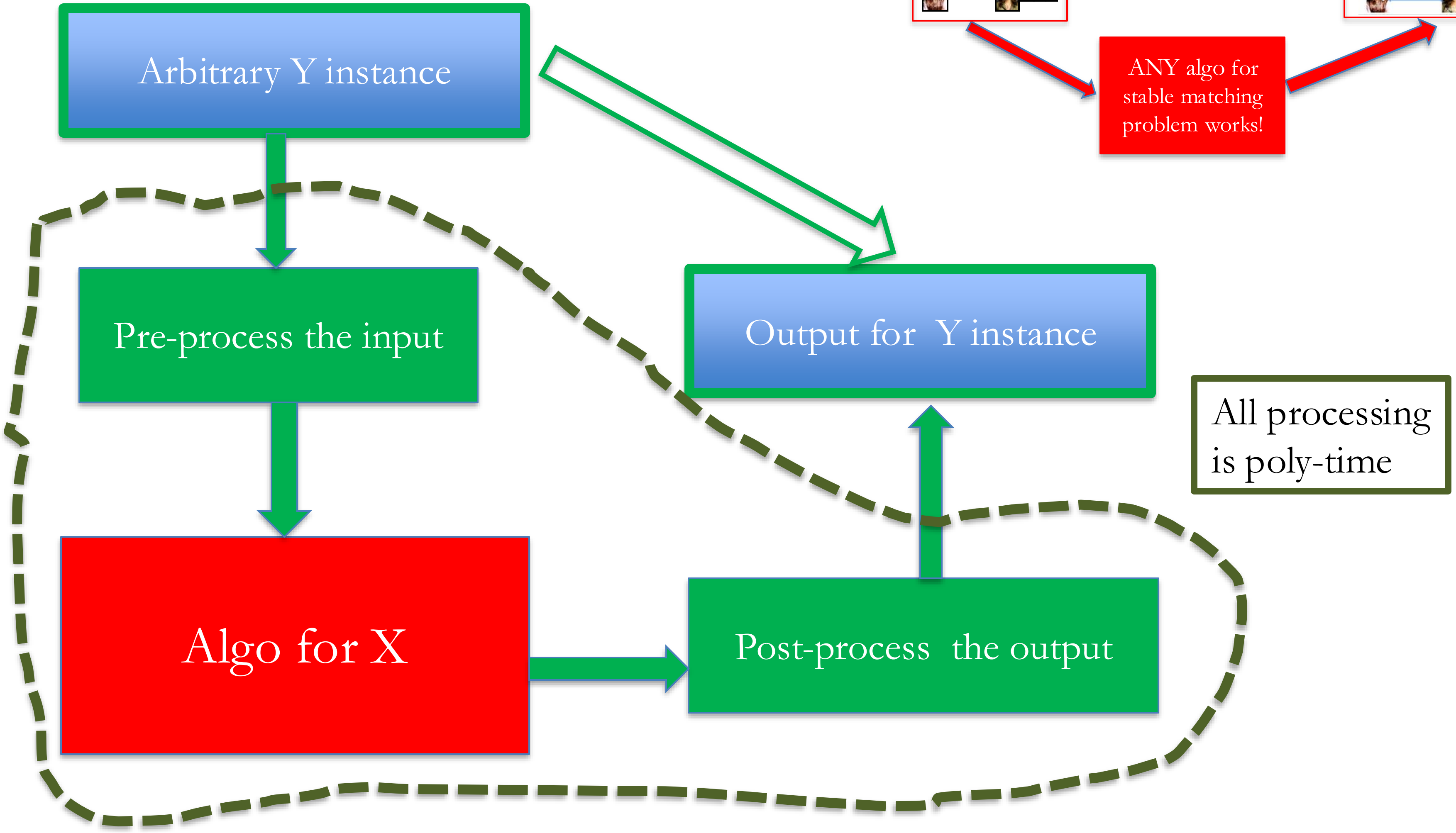


Production Company	Slot 1	Slot 2	Slot 3	Slot 4
P_1	S_1	free	S_2	free
P_2	free	S_1	free	S_2

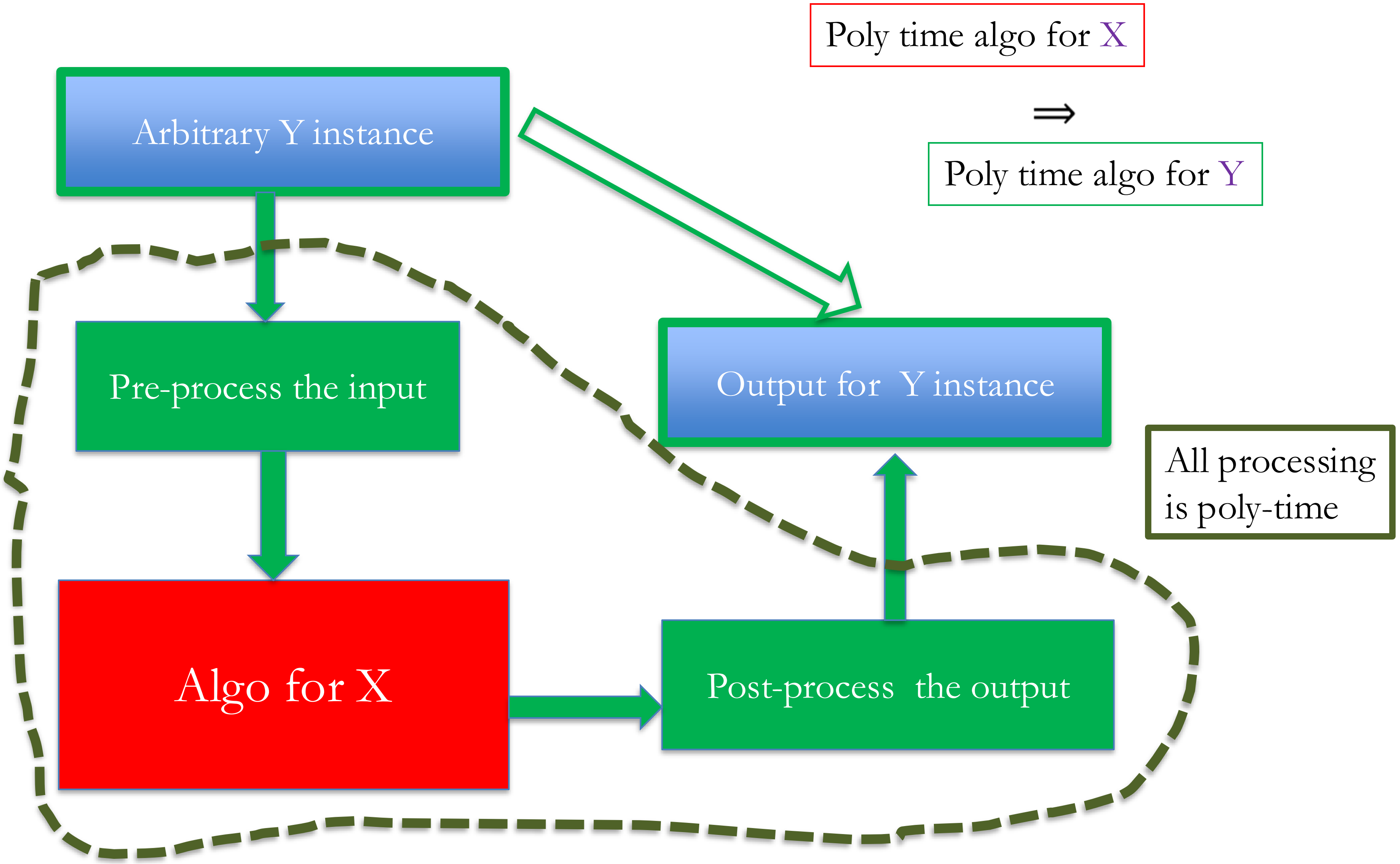


Production Company	Slot 1	Slot 2	Slot 3	Slot 4
P_1	S_1	free	S_2 (truncate here)	
P_2	free	S_1 (truncate here)		





Implications of Y $\leq_p$ X



$$A \Longrightarrow B$$

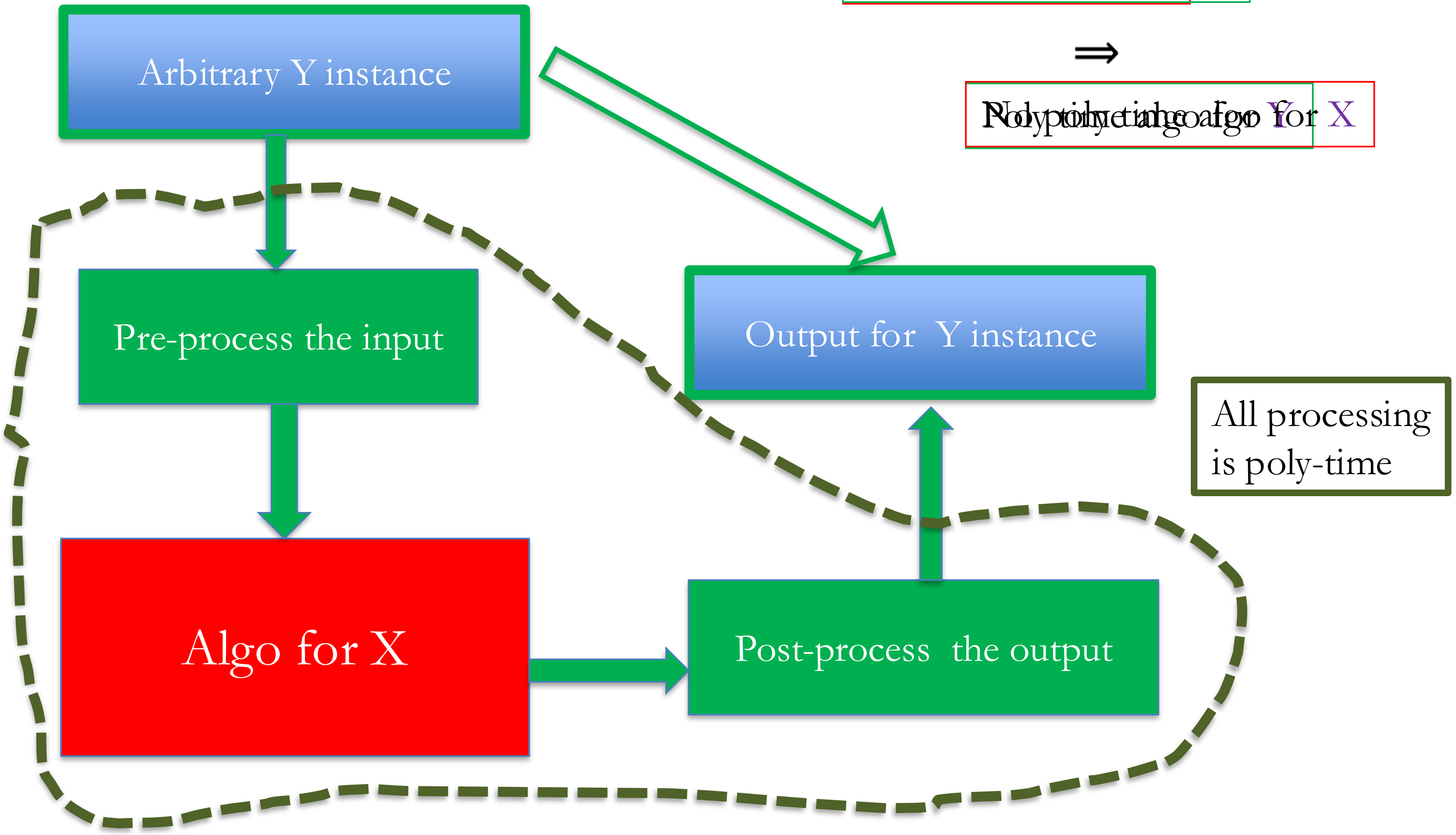
$$!B \Longrightarrow !A$$

Implications of $Y \leq_p X$

Poly time algo for X / No poly time algo for Y

$\Rightarrow$

Poly time algo for Y / No poly time algo for X



Plan for rest of today

More on reductions

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