Lecture 34

CSE 331

Nov 28, 2022

HW 7 reminders

Homework 7

Due by **11:30pm, Tuesday, November 29, 2022**.

Make sure you follow all the homework policies.

All submissions should be done via Autolab.

Question 1 (Ex 1 in Chap 6) [50 points]

The Problem

Exercise 1 in Chapter 6. The part `(a)` and `(b)` for this problem correspond to the part (**(a)**+**(b)**) and part **(c)** in Exercise 1 in Chapter 6 in the textbook (respectively).

Sample Input/Output

See the textbook for a sample input and the corresponding optimal output solution.

! Note on Timeouts

For this problem the total timeout for Autolab is 480s, which is higher the the usual timeout of 180s in the earlier homeworks. So if your code takes a long time to run it'll take longer for you to get feedback on Autolab. **Please start early to avoid getting deadlocked out before the submission deadline.**

Also for this problem, `C++` and `Java` are way faster. The 480s timeout was chosen to accommodate the fact that Python is much slower than these two languages.

Sample final exam

note @440

stop following 22 views

Actions

Sample final exam

Since one of you asked for it, 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 (will post on how to prepare for the final exam *after* the Thanksgiving break).
    - However, the sample exam was an actual final exam in one of the past years. Your final exam will be of comparable difficulty.

final

Edit good note 0

Updated 21 hours ago by Atri Rudra

Final exam post

note @458 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 Friday, 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.
    - I know this does not give a huge lead time into the final exam but unfortunately the snow day means less lead time than in previous years.
- Exam will be from **12:00pm to 2:30m** on Monday, **Dec 12** in class (**Hoch 114**). Note that the exam will be for 2.5 hours and *not 3 hours* as it says on HUB.
    - Remember the deadline to request a makeup final due to exam conflict is tomorrow, Monday, Nov 28 (@432)
- **DO NOT FORGET TO BRING YOUR UB CARD TO THE** EXAM (@460)

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

1. Take a look at the sample final (@440) 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 (@440).
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 (@81)
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 extend then 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

Bring your UB card to final

note @460

stop following **1 view**

Actions

Assigned seating for final exam

Your seating for the final in Hoch 114 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 Saturday, Dec 10. 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).

final

Edit good note 0

Updated 23 minutes ago by Atri Rudra

Incentive to complete course evals

note @459 · stop following · 0 views · Actions

Incentive for filling in course evals

As I have done in the past few years, depending on the level of response on the official course evals, I will release come questions on the final exam. (See @440 to see what Q I mean below)

- If **>=85%** students submit the course evals, I will release **Q1(a)**
- If **>=90%** students submit the course evals, I will release **Q1(a) AND Q2(a)**

Some other relevant comments:

- I will post the current response rate in the comments section below every 3 days AFTER the Thanksgiving break till the deadline
- The % is based on current number of students registered (133): i.e. it does not include students who have resigned
- I believe this is the link to the course evals: https://sunyub.smartevals.com/
  - But double check the email you might have received on this.

feedback

Edit · good note | 0 · Updated 25 minutes ago by Atri Rudra

followup discussions, *for lingering questions and comments*

Resolved · Unresolved · @459_f1 · Actions

Atri Rudra 25 minutes ago

Update (as of *Nov 27 at 1:10pm*): **2%** have filled in the course evals.

good comment | 0

Reply to this followup discussion

CSE 331 UTA positions for 2023

note @441

stop following 15 views

Actions

Want to be a UTA for 331 in 2023?

Prof. Akhter be teaching 331 in the upcoming Spring semester and is looking for UTAs. I expect to be teaching 331 again in Fall 2023 (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 2023 (on **zoom** tentatively the final week (Dec 13 and after) and/or the week after that (week of Dec 19), 2022).

(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 23 (5-10) as compared to Fall 23 (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.

If you are interested in a UTA position, please fill this form.

piazza logistics

Edit good note 0

Updated 14 hours ago by Atri Rudra

Next two weeks are brutal

Mon, Nov 28	The P vs. NP problem [F21] [F19]	[KT, Sec 8.1]
Tue, Nov 29		(**HW 8 out, HW 7 in**)
Wed, Nov 30	More on reductions [F21] [F19]	[KT, Sec 8.1]
Fri, Dec 2	The SAT problem [F21] [F19]	[KT, Sec 8.2] (**Project (Problem 3 `Coding`) in**)
Mon, Dec 5	NP-Completeness [F21] [F19]	[KT, Sec. 8.3, 8.4] (**Quiz 2**) (**Project (Problem 3 `Reflection`) in**)
Tue, Dec 6		(**HW 8 in**)
Wed, Dec 7	*k*-coloring problem [F21] [F19]	[KT, Sec 8.7]
Fri, Dec 9	*k*-coloring is NP-complete [F21] [F19]	[KT, Sec 8.7] (**Project (Problems 4 & 5 `Coding`) in**)
Mon, Dec 12	**Final Exam**	(**noon-2:30pm** in **HOCH 114 (usual classroom)**)
Tue, Dec 13		(**Project (Problems 4 & 5 `Reflection`) in**) (**Project Survey in**)

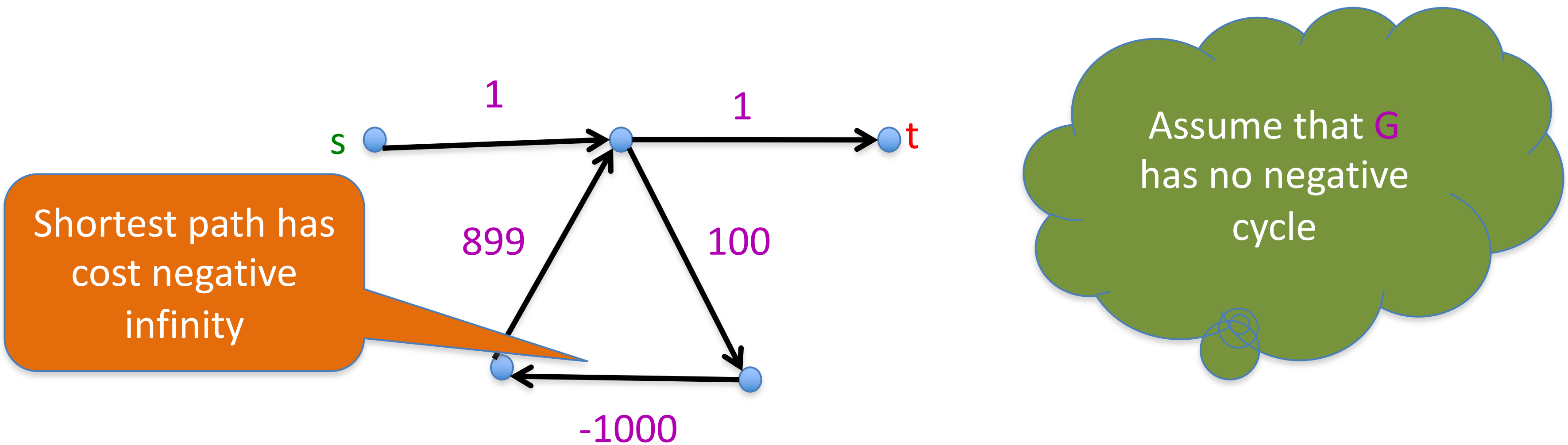
Questions?

Shortest Path Problem

Input: (Directed) Graph G=(V,E) and for every edge e has a cost c_e (can be <0)

t in V

Output: Shortest path from every s to t



The recurrence

$OPT(u,i)$ = shortest path from u to t with at most i edges

$$OPT(u,i) = \min \{ OPT(u,i-1), \min_{(u,w) \text{ in } E} \{ c_{u,w} + OPT(w, i-1)\} \}$$

Some consequences

OPT(u,i) = cost of shortest path from u to t with at most i edges

$$OPT(u,i) = \min \{ OPT(u, i-1), \min_{(u,w) \text{ in } E} \{ c_{u,w} + OPT(w,i-1)\} \}$$

OPT(u,n-1) is shortest path cost between u and t

Can compute the shortest path between s and t given all OPT(u,i) values

Bellman-Ford Algorithm

Runs in $O(n(m+n))$ time

Only needs $O(n)$ additional space

Questions?

Reading Assignment

Sec 6.8 of [KT]

Longest path problem

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

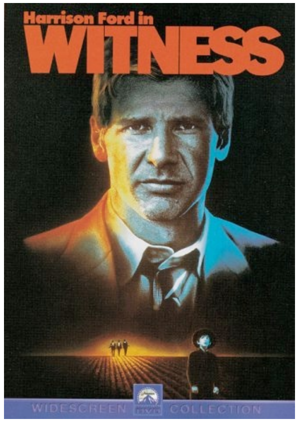
Longest vs Shortest Paths

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?

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

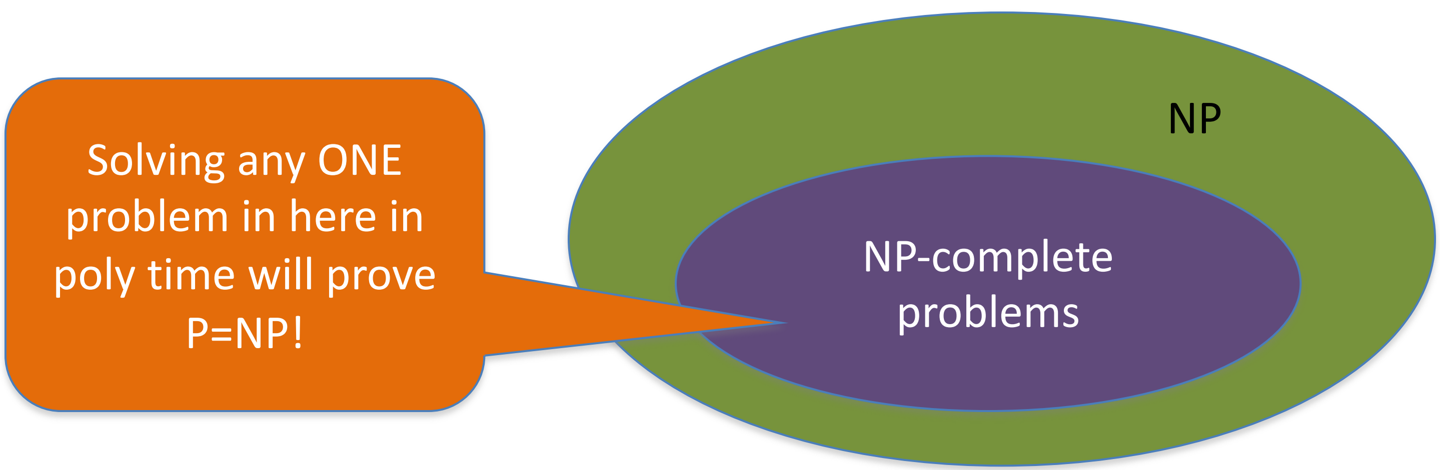
Pretty much all known proof techniques *provably* will not work

Proving P = NP

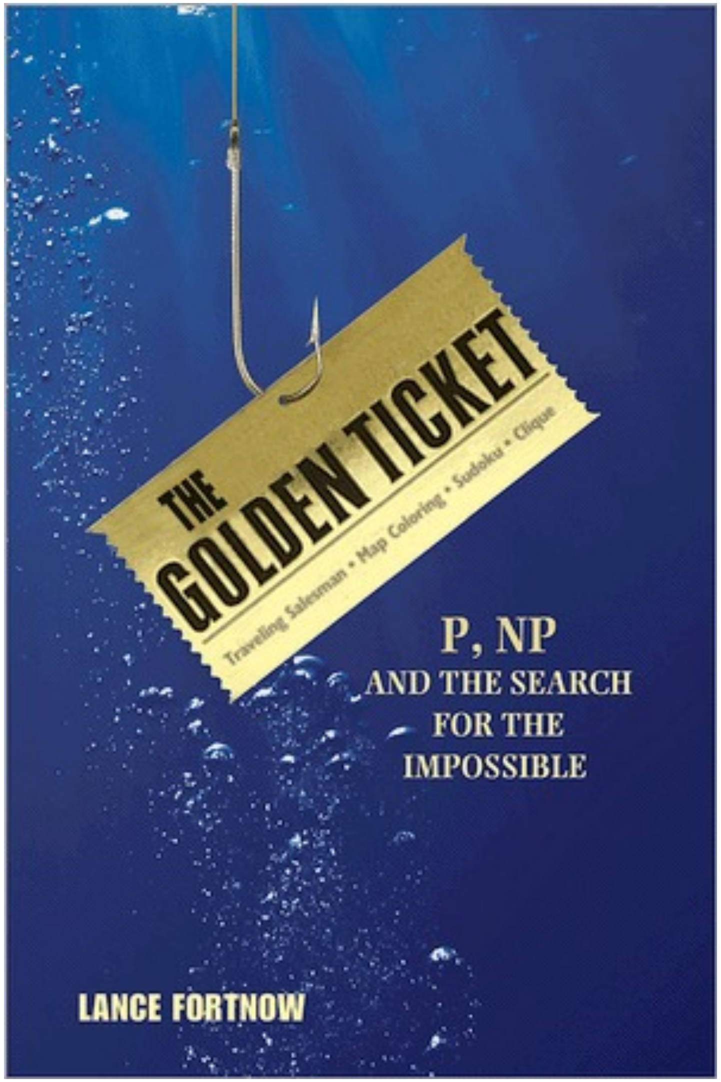
Will make cryptography collapse

Compute the encryption key!

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



A book on P vs. NP

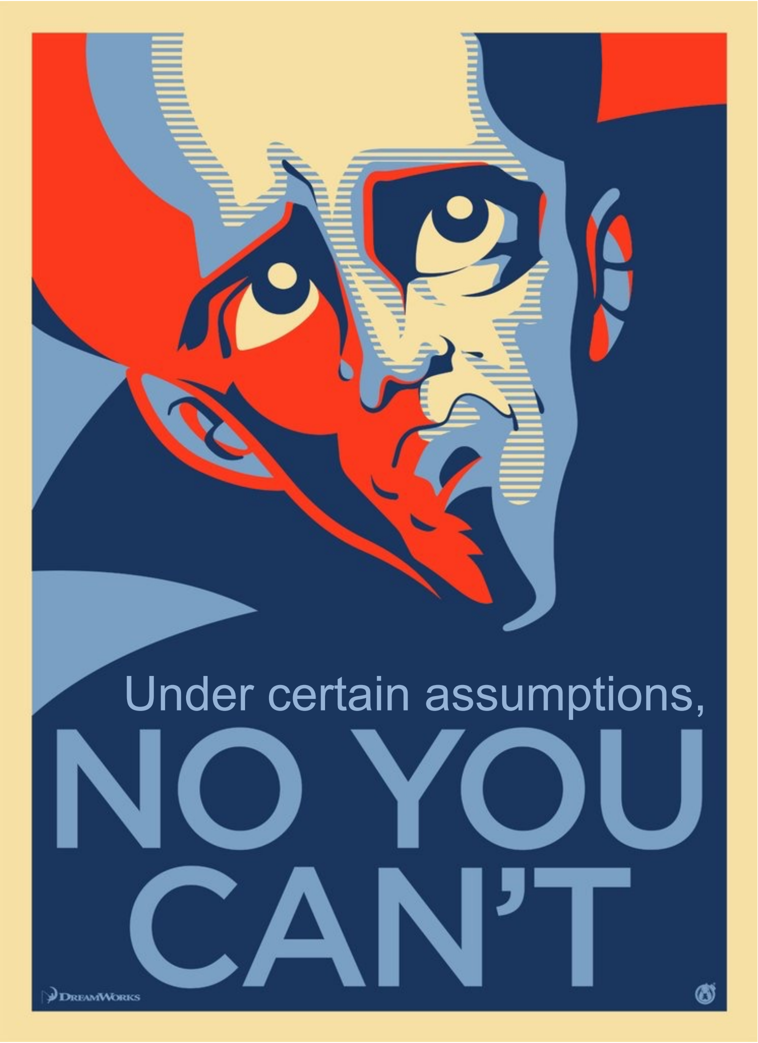


Questions?

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

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.

- `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

"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

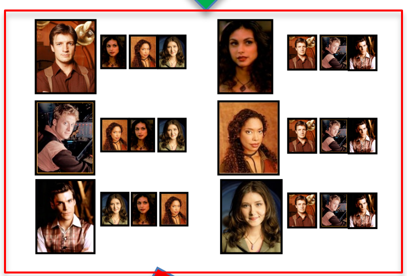
Overview of the reduction

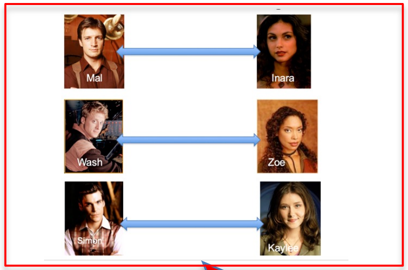
Question 2 (Syke(s) you out)

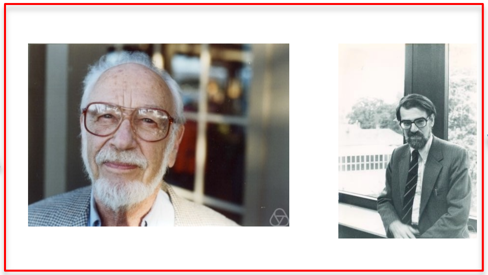
NRMP
National Resident Matching Program

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)		







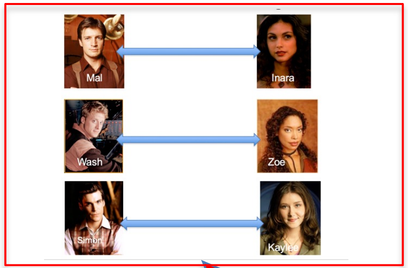
Nothing special about GS algo

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

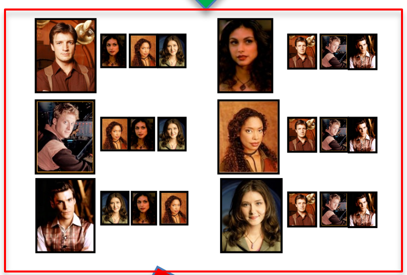
Question 2 (Syke(s) you out)

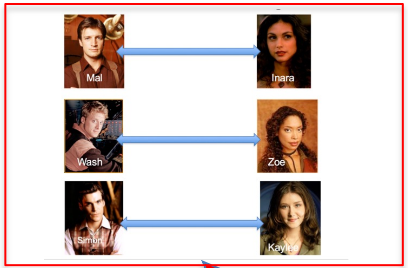
NRMP
National Resident Matching Program

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

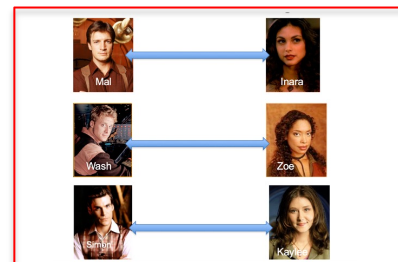
Question 2 (Syke(s) you out)

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)		

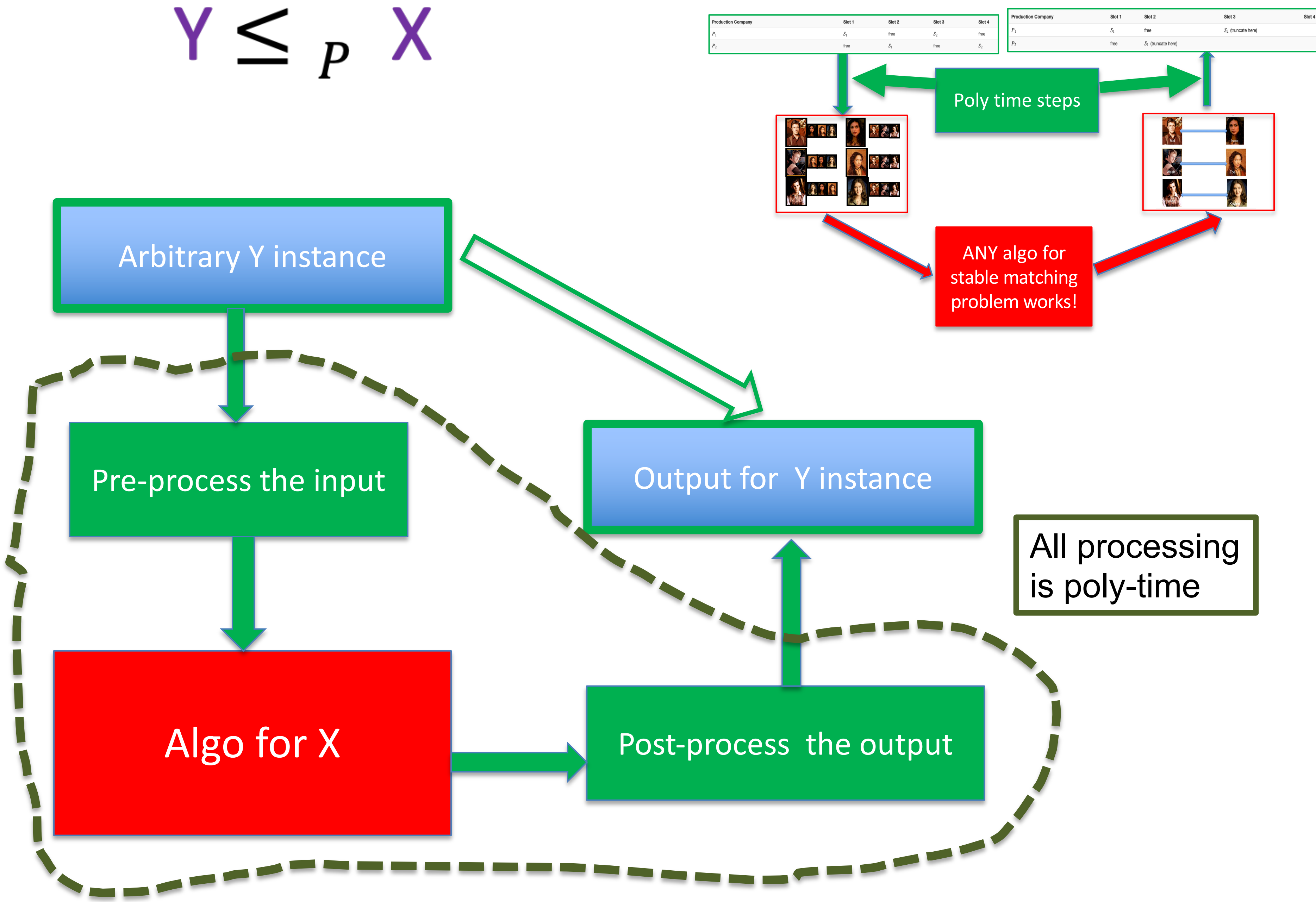
Poly time steps



ANY algo for stable matching problem works!

Question 2 (Syke(s) you out)

$$Y \le_p X$$

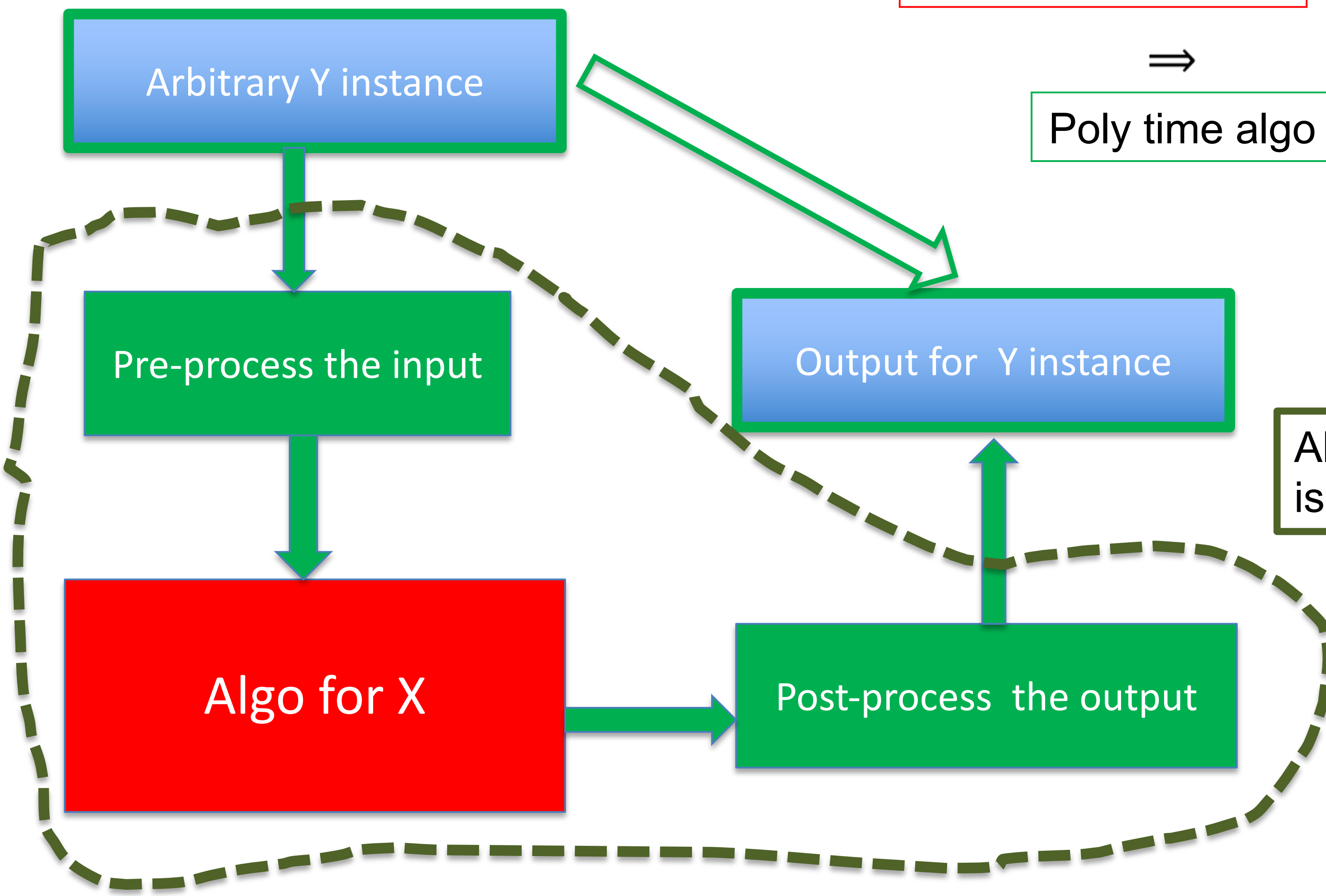


Implications of Y $\leq_p$ X

Poly time algo for X

$\Rightarrow$

Poly time algo for Y



All processing is poly-time

$$A \Longrightarrow B$$

$$!B \Longrightarrow !A$$

Implications of Y $\leq_P$ X