Lecture 34

CSE 331

Nov 18, 2019

Please re-download Py templates!

note

stop following 36 views

Please re-download Python zips for Problems 2-5

If you are not using Python for your mini-project, then you can safely ignore this post. Otherwise read-on.

tl;dr: **Please download the Python zips for Problem 2-5 and work on the updated zips from now on**.

Longer version: There was a bug in the template code that could throw the routing simulator into a tizzy. (*The grader code on Autolab has been fine and so no need to worry about your scores on Autolab changing*.) The updated zips should fix this and issue and you should now see the same revenue on both the template and Autolab.

If you are still reading: the issue at a high level was the following-- if your code was changing the input (e.g. bandwidths of nodess for P2), then the Autolab grader code was ignoring these changes (as it should). Unfortunately, the previous template code was **not** ignoring these changes, leading to a potentially weird behavior for the simulator. Anyhow, this should be fixed now!

(Of course if your code was not changing any part of the input, then this change should not have any effect!)
#pin

coding_mini_project

edit · good note | 0

Updated 1 hour ago by Atri Rudra

Curving coding project grade

note

stop following 95 views

Curving the coding mini-project grade

Based on the feedback so far (go to @1242 if you have not given feedback yet!), it seems like the coding project has ended up being bit more work than we had anticipated.

Given this, I am thinking of curving the coding mini-project grade. Please see below for more details on this but **please let me know via EMAIL by 5pm Monday, Nov 18 if you have any OBJECTIONS.** Since what I'm proposing is technically changing the grading policy, I need to make sure everyone is OK with this change. *If there are objections, then I might not be able to make the proposed changes below*.

Details on proposed change

The basic idea is that I will increase points on coding project as a whole as follows:

- Every group that submits all 5 problems get some fixed number of points.
- In addition to the above, every group's score will be increased in proportion to their actual score.
    - The actual scaling need not be linear but if a group ends up getting a full score of 100, under this proposed scheme they will end up getting some bonus points.

Few comments/clarifications:

- Y'all might have noticed that I did not give specific numbers in the above. This is on purpose: *the actual score increases will be determined **after Dec 6***.
- Note that in the scheme above, no one's score will go down and those who do really well end up getting some bonus points so I'm hoping this would be agreeable to everyone.

If there are any questions, please do not hesitate to ask in the comments section below!

Hope this helps y'all as you plan for the rest of the semester with regard to working on 331 stuff. And I do hope this give y'all more motivation to work on the coding project :-)
#pin

grading coding_mini_project

~ An instructor (Elijah Einstein) thinks this is a good note ~

Coding project mega page

note

62 views

Coding project mega-post

There have been a few pinned posts on coding mini-project recently and it is cluttering the pinned posts. So I'm collecting those posts here in once place. Hope this helps!

- Make sure you download the latest Python templates for all problems: @1283
- Explanation for the input file format: @1277
- The grade on the coding mini-project will be curved: @1273
- Make sure you download the latest Java templates for all problems: @1258
- Some comments on how to approach the coding mini-project: @1233

#pin

coding_mini_project

edit · good note | 0

Updated Just now by Atri Rudra

Upcoming deadlines @11am

Problem 2: Friday, Nov 22

Problem 3: Tuesday, Nov 26

Questions?

Please give feedback!

note

stop following 133 views

Feedback on CSE 331

I generally try to get feedback earlier in the semester but for various reasons this got pushed back this fall. Anyhow, please do give feedback via this anonymous Gform:

https://docs.google.com/forms/d/e/1FAIpQLSce1q7pg7TYdI1jA7g-SKbizWX1OOQZ4C2HUKr0bU8cN_zuxA/viewform?usp=sf_link

Filling in this form is **completely optional and anonymous**.

In particular, I would love feedback (even if it is critical). Many of the aspects of CSE 331 that you like were suggested by someone in a previous incarnation of CSE 331. While I'll try and incorporate as much as I can this fall, some of your suggestions might have to wait for the next offering.

I might also dis-agree with your feedback but after a week or so, I'll post my response to the feedback from y'all. So at the very least y'all would get to hear my reasoning for why certain things are the way they are in CSE 331. And then we can agree to disagree :-)
#pin

feedback

~ An instructor (Sanchit Batra) thinks this is a good note ~

edit · good note | 3

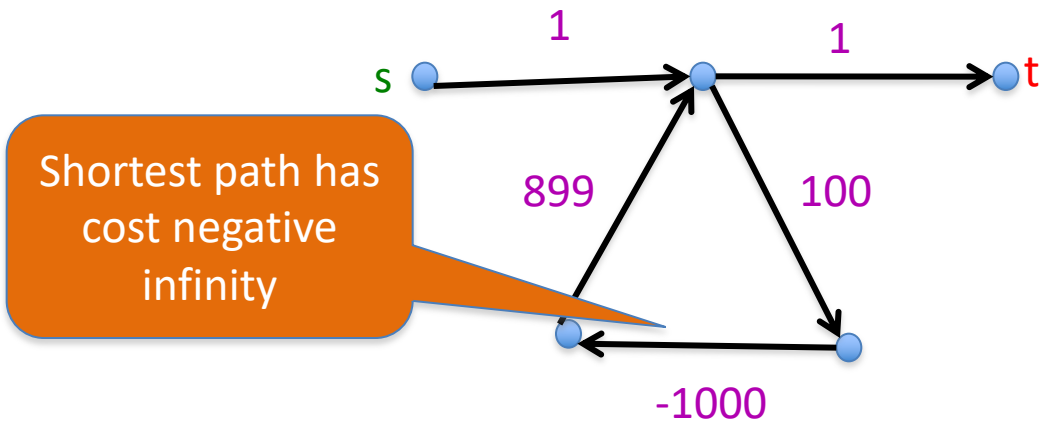
Updated 5 days ago by Atri Rudra

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



Assume that G has no negative cycle

The recurrence

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

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

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 ?

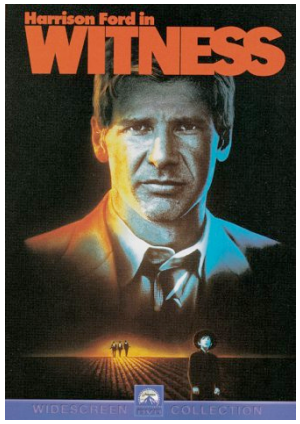
Longest vs Shortest Paths

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?

Clay Mathematics Institute

Dedicated to increasing and disseminating mathematical knowledge

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

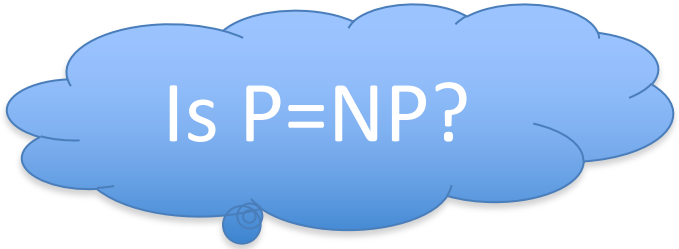
First Clay Mathematics Institute Millennium Prize Announced

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

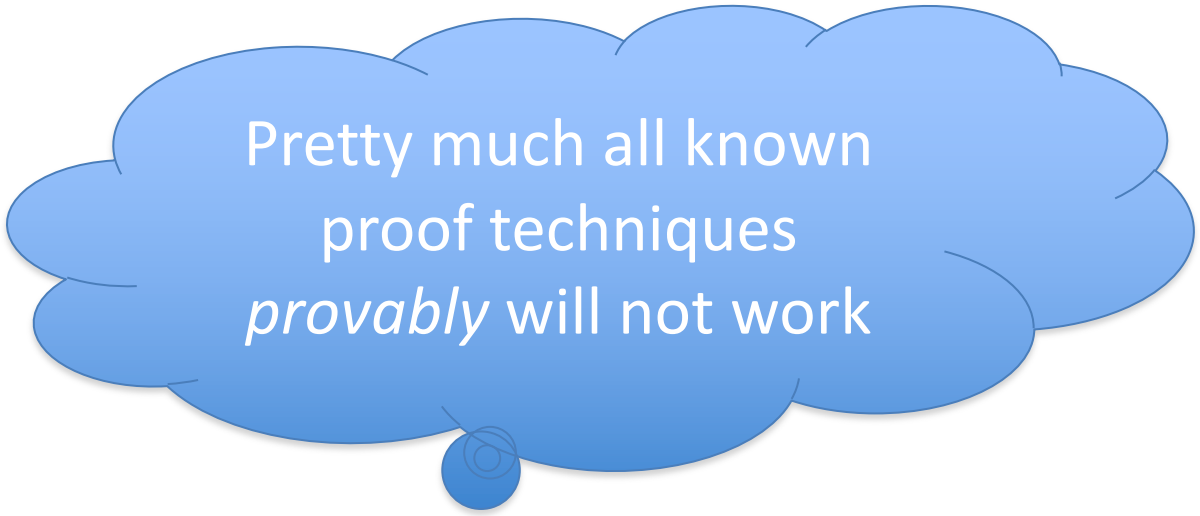


NP: problems that have polynomial time verifiable witness to optimal solution

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

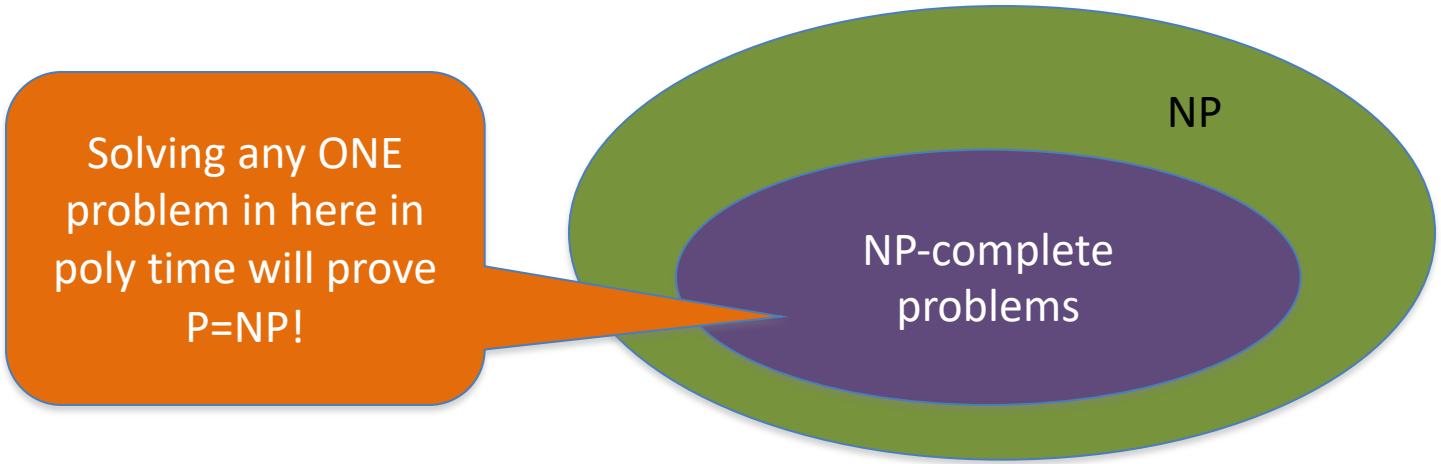


Proving P = NP

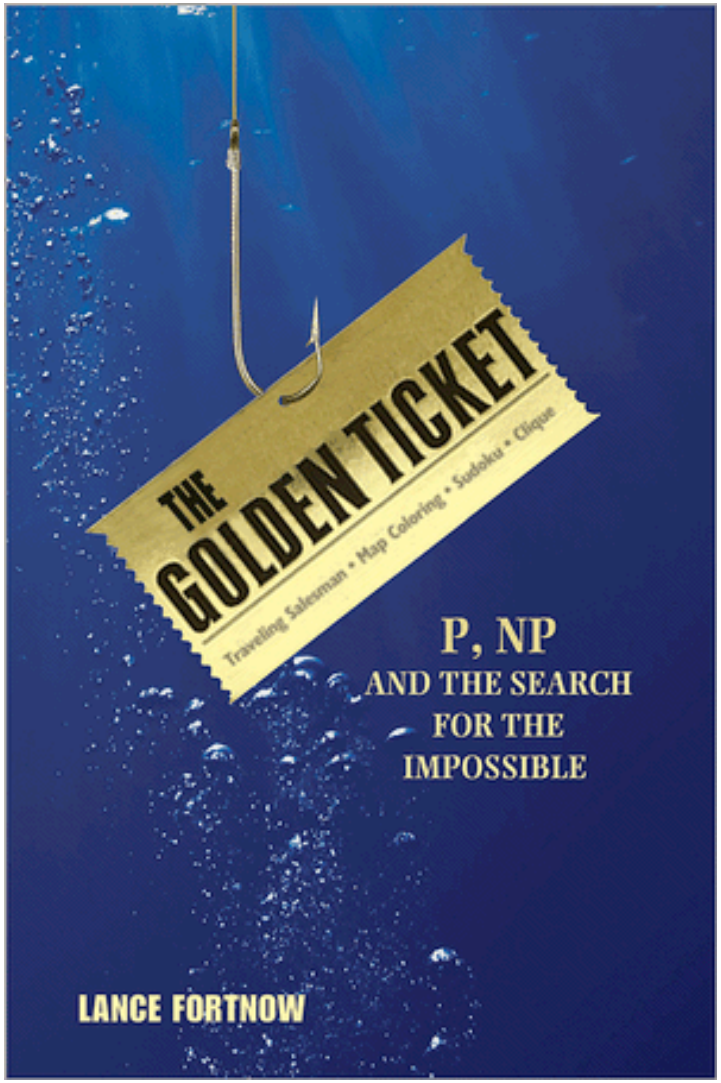
Will make cryptography collapse

Compute the encryption key!

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



A book on P vs. NP

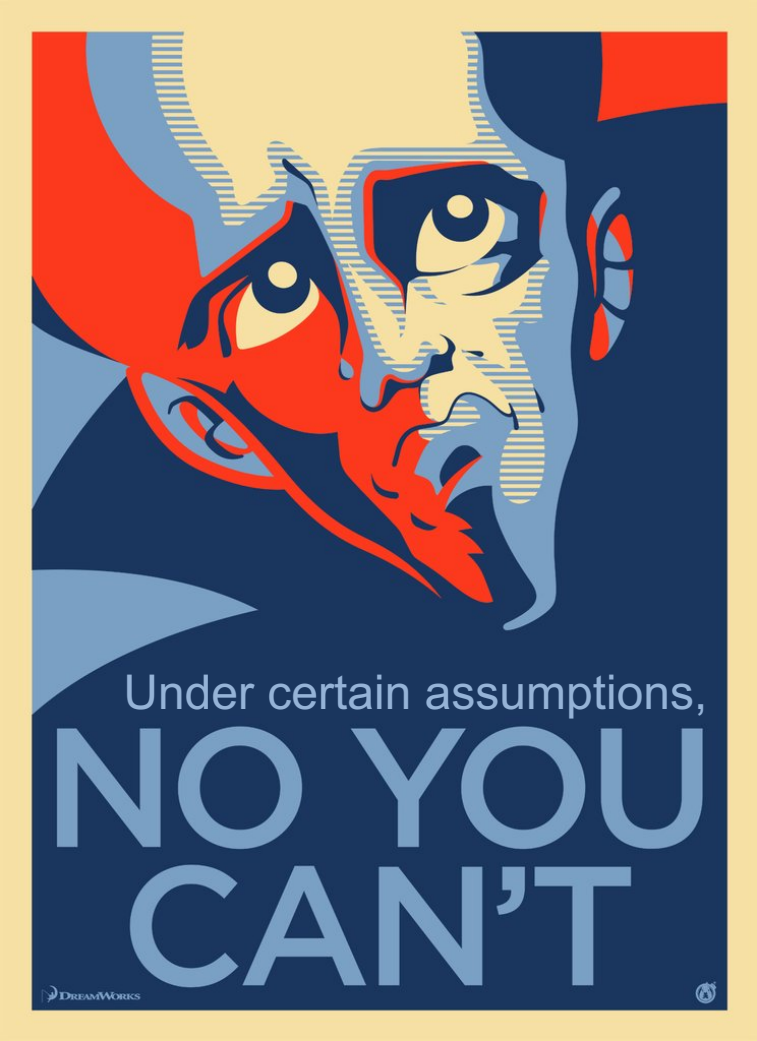


Questions?

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

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"

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

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 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 (Big G is in town) [25 points]

The Problem

The `Big G` company in the bay area decides it has not been doing enough to hire CSE grads from UB so it decides to do an exclusive recruitment drive for UB students. The `Big G` decides to fly over n CSE majors from UB to the bay area during December for on-site interview on a single day. The company sets up m slots in the day and arranges for n `Big G` engineers to interview the n UB CSE majors. (You can and should assume that $m > n$.) The fabulous scheduling algorithms at `Big G`'s offices draw up a schedule for each of the n majors so that the following conditions are satisfied:

- Each CSE major talks with every `Big G` engineer exactly once;
- No two CSE majors meet the same `Big G` engineer in the same time slot; and
- No two `Big G` engineers meet the same CSE major in the same time slot.

In between the schedule being fixed and the CSE majors being flown over, the `Big G` engineers were very impressed with the CVs of the CSE majors (including, ahem, their performance in CSE 331) and decide that `Big G` should hire all of the n UB CSE majors. They decide as a group that it would make sense to assign each CSE major S to a `Big G` engineer E in such a way that after S meets E during her/his scheduled slot, all of S's and E's subsequent meetings are canceled. Given that this is December, the `Big G` engineers figure that taking the CSE majors out to the nice farmer market at the ferry building in San Francisco during a sunny December day would be a good way to entice the CSE majors to the bay area.

In other words, the goal for each engineer E and the major S who gets assigned to her/him, is to **truncate** both of their schedules after their meeting and cancel all subsequent meeting, so that no major gets **stood-up**. A major S is stood-up if when S arrives to meet with E on her/his truncated schedule and E has already left for the day with some other major S'.

Your goal in this problem is to design an algorithm that always finds a valid truncation of the original schedules so that no CSE major gets stood-up.

To help you get a grasp of the problem, consider the following example for $n = 2$ and $m = 4$. Let the majors be S_1 and S_2 and the `Big G` engineers be E_1 and E_2. Suppose S_1 and S_2's original schedules are as follows:

CSE Major	Slot 1	Slot 2	Slot 3	Slot 4
S_1	E_1	free	E_2	free

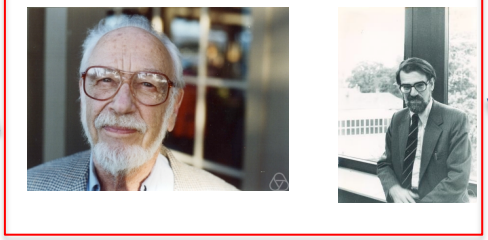
Overview of the reduction

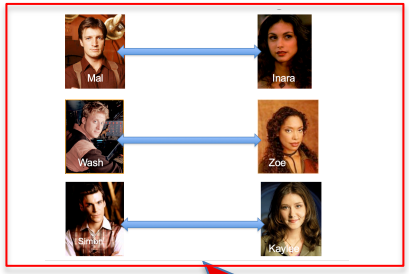
Question 2 (Big G is in town)

NRMP
National Resident Matching Program

CSE Major	Slot 1	Slot 2	Slot 3	Slot 4
S_1	E_1	free	E_2	free
S_2	free	E_1	free	E_2

CSE Major	Slot 1	Slot 2	Slot 3	Slot 4
S_1	E_1	free	E_2 (truncate here)	
S_2	free	E_1 (truncate here)		





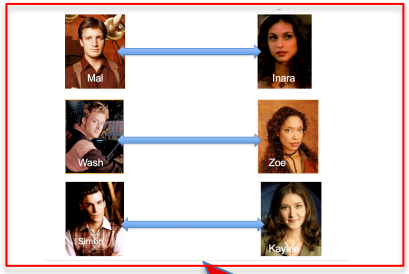
Nothing special about GS algo

Question 2 (Big G is in town)

NRMP
National Resident Matching Program

CSE Major	Slot 1	Slot 2	Slot 3	Slot 4
S_1	E_1	free	E_2	free
S_2	free	E_1	free	E_2

CSE Major	Slot 1	Slot 2	Slot 3	Slot 4
S_1	E_1	free	E_2 (truncate here)	
S_2	free	E_1 (truncate here)		



ANY algo for stable matching problem works!

Another observation

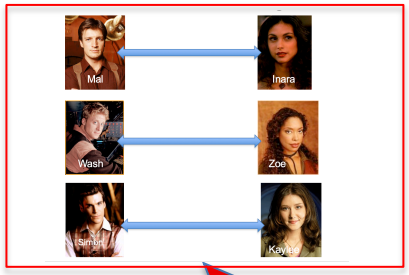
Question 2 (Big G is in town)

NRMP
National Resident Matching Program

CSE Major	Slot 1	Slot 2	Slot 3	Slot 4
S_1	E_1	free	E_2	free
S_2	free	E_1	free	E_2

CSE Major	Slot 1	Slot 2	Slot 3	Slot 4
S_1	E_1	free	E_2 (truncate here)	
S_2	free	E_1 (truncate here)		

Poly time steps



ANY algo for stable matching problem works!

Poly time reductions

Question 2 (Big G is in town)

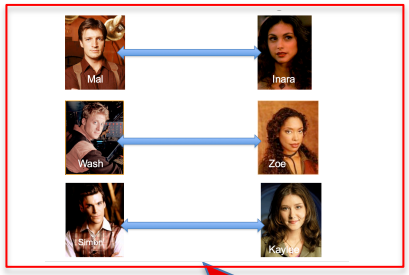
$\leq_P$

NRMP
National Resident Matching Program

CSE Major	Slot 1	Slot 2	Slot 3	Slot 4
S_1	E_1	free	E_2	free
S_2	free	E_1	free	E_2

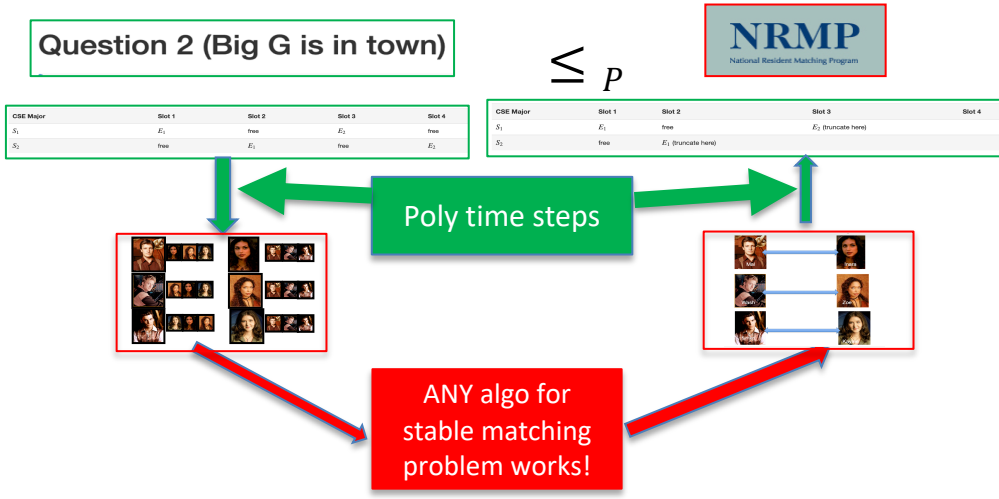
CSE Major	Slot 1	Slot 2	Slot 3	Slot 4
S_1	E_1	free	E_2 (truncate here)	
S_2	free	E_1 (truncate here)		

Poly time steps



ANY algo for stable matching problem works!

$Y \leq_P X$



Arbitrary Y instance

Pre-process the input

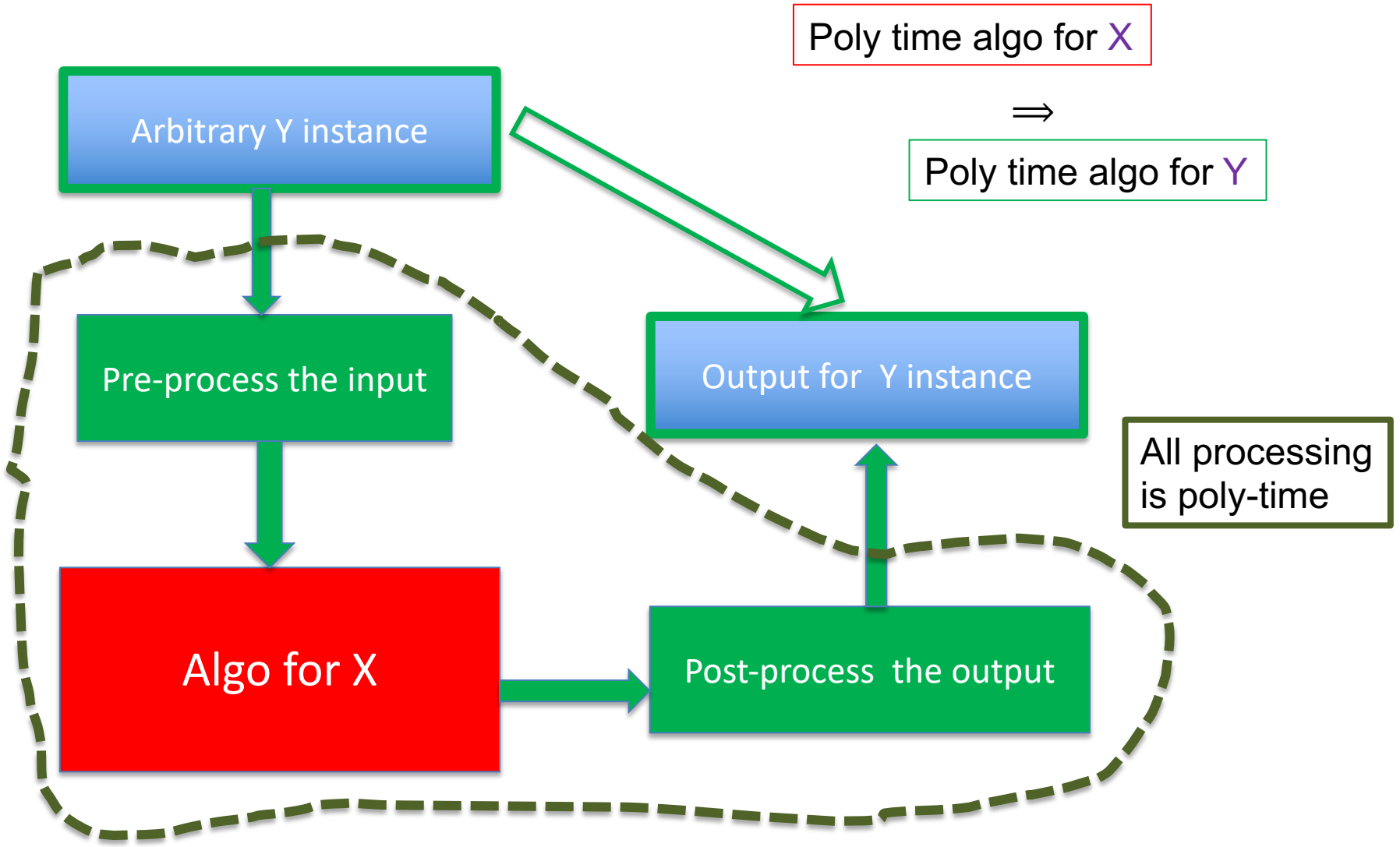
Algo for X

Post-process the output

Output for Y instance

All processing is poly-time

Implications of $Y \leq_P X$



$$A \Longrightarrow B$$

$$!B \Longrightarrow !A$$

Implications of $Y \leq_P X$