12 April 2021 Prelim 2 Review

Announcement:

- Prelim 2 Zoom links are not the same as the Zoom link for this lecture.
- You should have gotten email 3 days ago with your prelim time slot assignment and Zoom link.
- If you don't know your time slot and Zoom link, email me and Ruqi Zhang (rz297).
- Same rules as last time: open book, open notes (includes past homeworks and course handouts), if you are using an online resource such as Google Drive, Overleaf, DO NOT SHARE the document(s) with anyone!
- Time limit 2 hrs + 15 minutes. ← increased if you have extra time from SDS.
  Submit on CMS.

Prelim Coverage

Chapters 5, 7, 8.1–8.4

Div + Conq, Flow + flow reductions,

NP Completeness (reducing from 3SAT, CLIQUE, INDEP SET, VTX COVER).

See solution set 7 for example reductions. (Also review sheet listed on Prelim Review Materials post on Edi)

Types of questions...

* Asking for knowledge of particular algorithms that were taught.

Be prepared to run by hand...

* Ford Fulkerson
  + post-processing step to find ~ minimum cut given a max flow
* Push Relabel

... also know associated terms

- residual graph, residual capacity
- augmenting path
- preflow, excess, height

(+ algorithms listed above)

Other algorithms: know what problem they solve, know their running time
- Karatsuba's
- FFT
- Closest pair of points

* Divide and Conquer

You may be asked to judge correctness of a D&C algorithm that's given to you, prove its correctness, or analyze running time.
You won't be asked to design one.

Proving correctness: induction on input size.
Analyzing running time: solving a recurrence like

$$T(n) = 2 \cdot T(\tfrac{n}{2}) + O(n).$$

$$\Rightarrow T(n) = O(n \log n).$$

See §5.1–5.2 of textbook for general methodology of recurrences.

* Network flow and/or Flow Reductions.

- Prove or disprove something simple about max-flow problem.

  E.g. if f is a flow and (A,B) is a cut and $v(f) = f^{out}(A)$ then $f^{in}(A) = 0$.

  Proof. $v(f) = f^{out}(A) - f^{in}(A)$ for every flow f and cut (A,B), so $v(f) = f^{out}(A)$ implies $f^{in}(A) = 0$.

- Design + analyze a flow reduction. (Similar to PSet 6, Q1.)

- Circulation with demands + lower bds
  - You may use it if you wish, but the prelim won't contain problems whose intended solution uses it.

- Structure of a flow reduction.

[1] Design algorithm.

[1a] Specify how to convert problem data into vertices, edges, capacities.

Give names to vertices.
Make sure you indicate directions, capacities of edges.
Diagram can help, doesn't replace writing this info.

[1b] Specify how to convert a flow solution (integer flow if capacities are integers) back to a problem solution.

[2] Running time.
Specify whether using FF or Push-Relabel.
Running time is $O(m \cdot C)$ or $O(n^3)$ respectively.

↑ ↑
Convert these expressions into functions of original problem size

E.g. problem has k students and t classes, graph has $n = k \cdot t$ vertices.
Then running time of Push Relabel should be expressed as $O(k^3 t^3)$.

[3] Proof of correctness.

Show a way to transform

{valid problem solutions} ⟶ ⟵ {flow solutions}

Show that both of these transforms obey their respective problem constraints.

If the problem is an optimization problem, also show these transformations preserve optimality.

* NP Completeness.

DON'T REDUCE IN WRONG DIRECTION.

When asked to show COVID VACCINATION is NP-Complete, if you decide to use VERTEX COVER:

- Don't take a COVID VACCINATION as input and transform into a graph.
- Instead assume given an arbitrary graph (and $k>0$) as input, transform into an instance of COVID VACCINATION.

Steps in more detail:

(1) Show problem (e.g. CV) belongs to NP.
Describe a poly-time verifier that is given a CV problem and a proposed solution, checks the solution is valid.

(2) Reduce from a known NP-Comp problem to CV. E.g. VTX COVER $\leq_p$ CV.
Describe how the reduction works.

How do you take a VTX COVER problem and transform it to a CV problem?

(3) Show reduction runs in poly time.
(Most likely reason why your reduction wouldn't run in poly time: it first needs to find the vertex cover solution and makes use of that solution to create the CV problem instance.)

(4) Proof Step 1: If VC has a solution then CV created by reduction has a solution.
("Gadgets work as intended.")

(5) Proof Step 2: If the CV problem created by reduction has a solution then VC has a solution.
("No unintended way to use the gadgets.")

Problem 8.6 from book

MSFTV is the following decision problem.

Given an instance of CNF-SAT with no negated variables, e.g.

$$(x_1 \vee x_2) \wedge (x_1 \vee x_3) \wedge (x_2 \vee x_3)$$

does $\exists$ a satisfying truth assignment with $\leq k$ variables evaluating to T?

Prove NP-Complete.

1. Verifier takes ϕ and truth assignment y and checks:
   - at most k vars are true in y
   - y satisfies every clause of ϕ

   This takes linear time.

2. VTX COVER → MSFTV

	VTX COVER	MSFTV
Choices:	include vertex or not?	T or F?
Constraints:	$\leq k$ vertices	$\leq k$ true vars
	cover each edge	satisfy each clause

Given (G, k), create ϕ with variables $x_1, \ldots, x_n$ corresponding to vertices $v_1, \ldots, v_n$ of G.

One clause for each edge.
Edge $\{v_i, v_j\}$ becomes clause $(x_i \vee x_j)$

Use same value of k in both problems.

3. Running time: linear.
(# vars = # vert, # clauses = # edge)

4. If G has k-element vertex cover $S = \{v_{i_1}, v_{i_2}, \ldots, v_{i_k}\}$ set

$$x_i = \begin{cases} \text{TRUE} & \text{if } v_i \in S \\ \text{FALSE} & \text{if } v_i \notin S. \end{cases}$$

This sets $\leq k$ variables TRUE because $|S| = k$.
It satisfies every clause because of the vertex cover property.
For clause $(x_i \vee x_j)$ we know (v_i, v_j) is an edge of G
$\Rightarrow$ v_i or v_j belongs to S

$\Rightarrow x_i$ or x_j is TRUE

$\Rightarrow (x_i \vee x_j)$ is satisfied.

5. If ϕ is satisfied by truth assignment with $\leq k$ true vars, G has a k-element Vtx cover.

Proof. Let $S = \{ v_i \mid x_i = \text{TRUE in the satisfying assignment} \}$

By assumption on # true vars, $|S| \leq k$.

By the fact that each clause is satisfied, S contains at least one endpoint of each edge.

$\therefore$ S is a vertex cover of size $\leq k$.