Portland State University

W'21 CS 584/684

Algorithm Design & Analysis

Fang Song

Lecture 20

- Approx./R. algorithms
- Review

Randomized quicksort

- Pick the pivot randomly

```
Rand-QuickSort(A):
 if (array A has zero or one element)
    Return
 Pick pivot p ∈ A uniformly at random
 (L, M, R) ← PARTITION − 3 − WAY(A, p)      → O(n)
 Rand-QuickSort(L)                           → T(i)
 Rand-QuickSort(R)                           → T(n − i − 1)
```

Theorem. The expected number of compares to quicksort an array of n distinct elements is $O(n\log n)$.

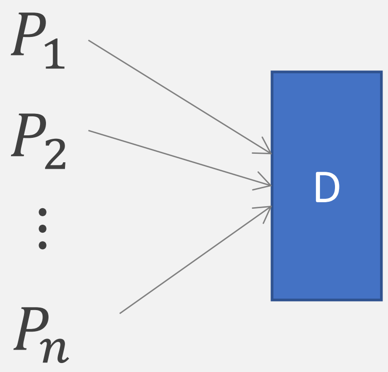
Contention resolution in a distributed system

Given: processes $P_1, \ldots, P_n$,

- each process competes for access to a shared database.
- If ≥ 2 processes access the database simultaneously, all processes are locked out.

Goal: a protocol so all processes get through on a regular basis

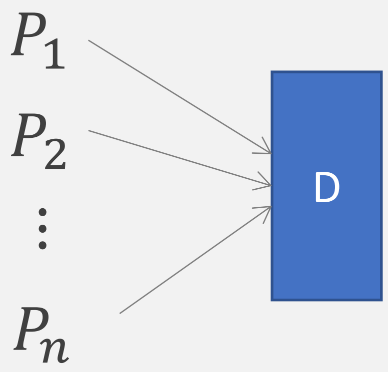
- Restriction: Processes can't communicate.



Contention resolution: randomized protocol

Protocol. Each process requests access to the database in round t with probability $p = 1/n$.

Theorem. All processes will succeed in accessing the database *at least once* within $O(n \ln n)$ rounds except with probability $\leq \frac{1}{n}$.



Randomized contention resolution: analysis 1

Def. $S[i,t]$ = event that process i succeeds in accessing the database in round t.

- Claim1. $\frac{1}{e \cdot n} \le \Pr(S[i,t]) \le \frac{1}{2n}$
- Pf. $\Pr(S[i,t]) = p(1-p)^{n-1}$

[Geometric distribution: independent Bernoulli trials]

Process i requests access None of remaining request access

$\Rightarrow \Pr(S[i,t]) = \frac{1}{n}(1-1/n)^{n-1} \in [\frac{1}{en}, \frac{1}{2n}]$ $[p = 1/n]$

- $(1-1/n)^n$ converges monotonically from $1/4$ up to $1/e$.
- $(1-1/n)^{n-1}$ converges monotonically from $1/2$ down to $1/e$.

Randomized contention resolution: analysis 2

- Claim2. The probability that process i fails to access the database in $e \cdot n$ rounds is at most $1/e$. After $e \cdot n\ (c \ln n)$ rounds, the probability $\leq\ n^{-c}$.
- Pf. Let $F[i,t]$ = event that process i fails to access database in rounds 1 through t.

$$\Pr(F[i,t]) = \Pr\left(\overline{S[i,1]}\right) \cdot \ldots \cdot \Pr\left(\overline{S[i,t]}\right) \leq \left(1 - \frac{1}{en}\right)^t$$ [Independence & Claim 1]

  - Choose $t = en$: $\Pr(F[i,t]) \leq \left(1 - \frac{1}{en}\right)^{en} \leq \frac{1}{e}$
  - Choose $t = en \cdot c\ln n$: $\Pr(F[i,t]) \leq \left(\frac{1}{e}\right)^{c\ln n} \leq n^{-c}$

Randomized contention resolution: analysis 3

Theorem. All processes will succeed in accessing the database *at least once* within $2en \ln n$ rounds except with probability $\leq \frac{1}{n}$.

- Pf. Let $F[t]$ = event that some process fails to access database in rounds 1 through t.

Union Bound

Let E, F be two events. Then $\Pr(E \cup F) \leq \Pr(E) + \Pr(F)$.

$$\Pr(F[t]) = \Pr(\cup_{i=1}^{n} F[i,t]) \leq \sum_{i=1}^{n} \Pr(F[i,t]) \leq n \cdot \Pr(F[1,t])$$

- Choose $t = en \cdot 2\ln n$: $\Pr(F[t]) \leq n \cdot n^{-2} = 1/n$

Integer linear programming (ILP)

Input. Graph $G = (V, E)$

- Vertex cover $S \subseteq V$: subset of vertices that touches all edges

Goal. Find a vertex cover S of minimum size

- Formulating vertex cover as an integral linear program

For each $i \in V$, introduce $x_i \in \{0,1\}$
Min $\sum_{i=1}^{n} x_i$
Subject to:
$x_i + x_j \geq 1$ for each $(i, j) \in E$

[i.e., Pick i in vertex cover iff. $x_i = 1$]

☹ We don't know (expect) a poly-time algorithm (ILP)

- Without integrality (LP), we do know poly-time algorithms

Putting aside the integral constraint

(ILP Π) Min $\sum_{i=1}^{n} x_i$
Subject to:

$$x_i + x_j \geq 1, \qquad \forall (i,j) \in E$$
$$x_i \in \{0,1\}, \qquad \forall i \in V$$

(LP Σ) Min $\sum_{i=1}^{n} x_i$
Subject to:

$$x_i + x_j \geq 1, \qquad \forall (i,j) \in E$$
$$0 \leq x_i \leq 1, \qquad \forall i \in V$$

Let x^* be an optimal soln. for LP Σ
& optimal value $\mathrm{OPT} = \sum_i x_i^*$

?

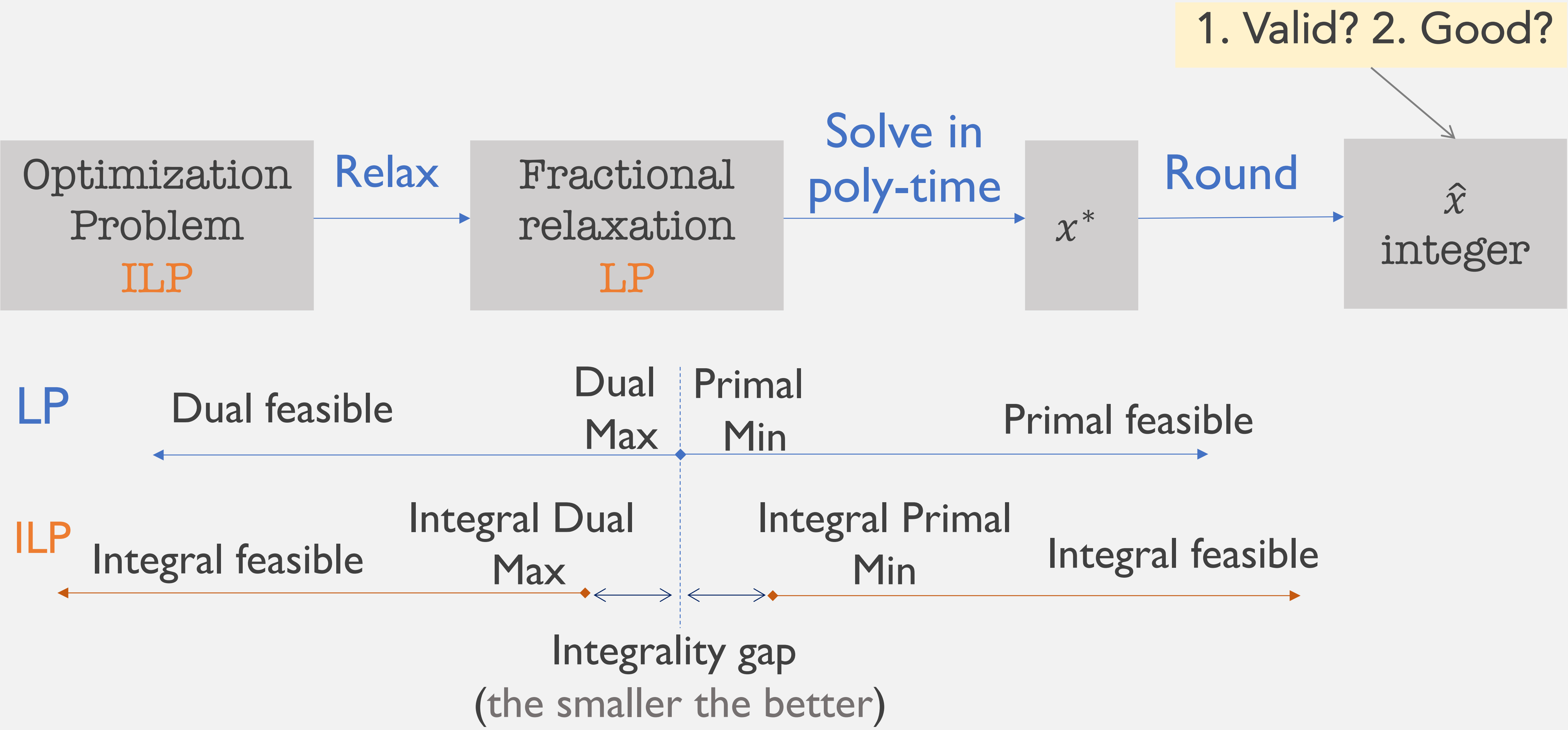
$$x_i := \lfloor x_i^* \rceil = \begin{cases} 1, & \text{if } x_i^* \geq \frac{1}{2} \\ 0, & \text{otherwise} \end{cases}$$

- (Threshold) Rounding:

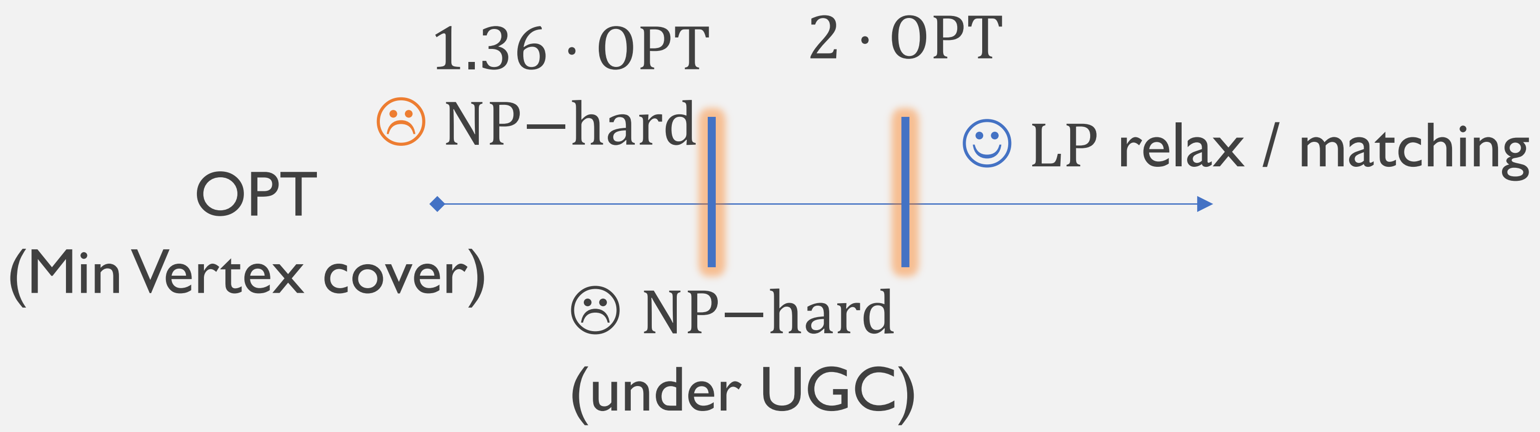
i. $\{x_i\}$ is a feasible integral solution: $\forall (i,j) \in E$, $x_i^* \geq \frac{1}{2}$ or $x_j^* \geq \frac{1}{2}$ or both

ii. $\sum_i x_i \leq \sum_i 2 \cdot x_i^* = 2 \cdot \mathrm{OPT} \leq 2 \cdot \mathrm{OPT}_{\mathrm{Int}}$ [optimal value of ILP Π, i.e. size of min vertex cover]

LP relaxation



Hardness of approximation



Theorem. It is NP-Hard to approximate Vertex Cover to with any factor below 1.36067. [i.e., otherwise, you can solve 3-SAT in poly-time]

Theorem’. It is NP-Hard to approximate Vertex Cover to with any factor below 2, assuming the unique games conjecture (UGC).

Want to read more?
https://cs.nyu.edu/~khot/papers/UGCSurvey.pdf
https://cs.stanford.edu/people/trevisan/pubs/inapprox.pdf

Approximating set cover

Input. Set U of n elements, $S_1, \ldots, S_m$ of subsets of U

Goal. Find $I \subseteq \{1, \ldots, m\}$ of minimum size such that $\bigcup_{i \in I} S_i = U$

(ILP Π for Set cover)

For each $i \in \{1, \ldots, m\}$, introduce $x_i \in \{0,1\}$

Min $\sum_{i=1}^{m} x_i$

Subject to:

$$\sum_{i:u \in S_i} x_i \geq 1, \qquad \forall u \in U$$

LP relaxation for set cover

(Set cover ILP Π)
Min $\sum_{i=1}^{m} x_i$
Subject to:

$$\sum_{i:u\in S_i} x_i \geq 1, \qquad \forall u \in U$$

$$x_i \in \{0,1\}, \qquad \forall i \in \{1,\ldots,m\}$$

(Set cover Σ)
Min $\sum_{i=1}^{m} x_i$
Subject to:

$$\sum_{i:u\in S_i} x_i \geq 1, \qquad \forall u \in U$$

$$0 \leq x_i \leq 1, \forall i \in \{1,\ldots,m\}$$

Let x^* be an optimal soln. for LP Σ & optimal value $\mathrm{OPT} = \sum_i x_i^*$

? $x_i := \lfloor x_i^* \rceil$

- Threshold rounding: does it cover all elements?
  - Ex. $u \in S_1, \ldots, S_{100}; x_1^*, \ldots x_{100}^* = \frac{1}{100} \Rightarrow x_1 = \cdots = x_{100} = 0$. u is missed!
- Randomized rounding!

LP relaxation for set cover

(Set cover ILP Π) Min $\sum_{i=1}^{m} x_i$
Subject to:
$$\sum_{i:u\in S_i} x_i \geq 1, \ \forall u \in U$$
$$x_i \in \{0,1\}, \qquad \forall i \in \{1, \dots, m\}$$

(Set cover LP Σ) Min $\sum_{i=1}^{m} x_i$
Subject to:
$$\sum_{i:u\in S_i} x_i \geq 1, \ \forall u \in U$$
$$0 \leq x_i \leq 1, \forall i \in \{1, \dots, m\}$$

Let x^* be an optimal soln. for LP Σ & optimal value $\mathrm{OPT} = \sum_i x_i^*$

☹ $x_i := \lfloor x_i^* \rceil$

- Randomized rounding: set $x_i = 1$ with probability x_i^*

$$\mathbb{E}[\sum_{i=1}^{m} x_i] = \sum_{i=1}^{m} \mathbb{E}[x_i] = \sum_{i=1}^{m} x_i{}^*$$

- But is it feasible? [Further analysis on Panigrahi's notes]

Theorem. There is a poly-time randomized algorithm achieving $O(\log n)$ expected approximation ratio, except w. probability $O(1/n)$.

You've accomplished a lot!

Be proud of yourselves!

Final exam

- When
  - Take-home. Release on Tuesday (03/16)at 4pm, due on Wednesday(03/17) at 4pm.
  - I will be online (slack) during 5:30 – 7:20pm to answer clarification questions.
- What
  - Comprehensive, slightly more focused on 2nd half.
- How
  - Similar format as mid-term: short-answer questions and algorithm designs.
  - No external resource permitted. Violations will be taken seriously.
  - No credit for unintelligible hand writing.

Principal questions

0 What problem to solve?

be as precise as possible

1 Is the algorithm correct?

DESIGN

2 How much resource it costs?

(time, space, …)

ANALYSIS

3 Can we do better?

Major topics

- Basics: asymptotic, graphs (BFS/DFS), data structures
- Algorithmic techniques
  1. Divide-&-Conquer
  2. Dynamic Programming
  3. Greedy
  4. Network flow & linear programming
  5. Randomization

  ★ Reduction
- Computational intractability: P, NP, NPC, approximation

1st half

2nd half

1. Divide-&-Conquer

- Idea
  - Divide into independent subproblems – recurse - combine
- Examples
  - Merge sort $O(n \log n)$
  - Fast multiplication $O(n^{1.59})$ [Karatsuba60]; $O(n \log n)$ [HarveyHoeven19]
  - Matrix multiplication $O(n^{2.81})$ [Strassen69]; $O(n^{2.376})$ [CoppersmithWinograd90];
  - Exponentiation $O(n \log n)$
  - Quick sort $O(n^2)$ worst-case; Expected $O(n \log n)$ random pivoting
- Analysis.
  - Solving recurrence: $T(n) = aT(n/b) + f(n)$
  - Recursion tree & Master theorem

2. Dynamic programming

- Idea
  - Divide into overlapping subproblems – smart recurse by memoization
  - Usually bottom-up iteration (topological order of implicit DAG)
- Examples
  - Fibonacci $O(n)$
  - Longest increasing subsequence $O(n^2)$
  - Weighted interval scheduling $O(n \log n)$
  - Matrix-chain multiplication $O(n^3)$
  - Longest common subsequence (aka Edit Distance) $O(mn)$
  - Shortest path (w. negative lengths) $O(mn)$ [Bellman-Ford]

3. Greedy

- Idea
  - Special case of DP: when lucky, lazy choice works
- Examples
  - Shortest path (w. non-negative lengths) $O((m+n)\log n)$ [Dijkstra]
  - Interval scheduling (weight = 1) $O(n \log n)$
  - Interval partitioning $O(n \log n)$
  - Minimum spanning tree $O(m \log n)$ [Kruskal]; $O((m+n)\log n)$ [Prim]
- Warning! 0 credit in exam without correctness proofs

Detour

- data structures [Prioirity Queue, Union Find]
- amortized analysis

4. Network flow - Linear programming

Network flow $\leq$ **Linear programming**

- Analytical
  - Max-Flow $\equiv$ Min-Cut
- Algorithms
  - Augmenting path: $O(mnC)$ [Ford-Fulkerson]
  - Capacity scaling: $O(m^2 \log C)$
  - In exam: quote $O(mn)$
- Applications
  - Bipartite perfect matching

- Analytical
  - Duality: OPT(Primal) = OPT(Dual)
- Algorithms
  - Simplex [efficient in practice/ but not poly-time worst-case]
  - Ellipsoid [poly-time but not practical]
  - Interior point [poly-time & practical]
- Warning: don't reduce to LP unless stated explicitly

5. Randomization

- Idea
  - Make random choices to get correct answers with high probability in (expected) poly-time.
- Examples
  - Contention resolution
  - Randomized quicksort
  - Randomized rounding for LP relaxation
- Important probabilistic tools
  - Union bound
  - Linearity of expectation

Computational intractability

- Classify problems by "hardness"
  - **P**: feasible problems (solvable in poly-time).
  - **NP**: ∃ poly-time certifier verifying a solution.

P vs. NP?

- Reduction: relating hardness ($A \leq B \Rightarrow A$ no harder than B)
  - Cook reduction [aka poly-time reduction]
  - Karp reduction [aka poly-time transformation]
- $\mathbf{NP{-}complete}$: 1) $A \in \mathbf{NP}$ & 2) $\forall B \in \mathbf{NP}, B \leq_{Karp,P} A$ [$\mathbf{NP}{-}\mathrm{hard}$]
  - $\mathrm{Circuit{-}SAT}$ is $\mathbf{NPC}$
  - $\mathrm{Circuit{-}SAT} \leq \mathrm{3{-}SAT} \leq \mathrm{INDEPENDENT{-}SET} \leq \mathrm{VERTEX{-}COVER} \leq \mathrm{SET{-}COVER} \leq \mathrm{IntegerLP}$
  - $\mathrm{3{-}SAT} \leq \mathrm{HAM{-}CYCLE}$

Coping with NPC: approximation algorithms

- Greedy
  - Vertex cover & set cover

- LP relaxation
  - Threshold rounding: 2-approx. vertex cover
  - Randomized rounding: $O(\log n)$-approx. set cover

★ Know the facts and ideas! Details less important

FAQs

- How should I study for it?
  - Review the fundamentals
  - Reproduce the algorithms & analysis for all problems you've seen (lecs, text, hw…)
  - Practice exam: emulate a real exam environment

- Reminders
  - If no running time requirement, always aim for fastest algorithms you can think of.
  - Asked or not, always provided analysis (correctness and runtime) on algorithm design problems.
  - Always start with a short description of the main idea of your algorithm.
  - Reductions: mind the direction (e.g., in NPC proofs).
  - A guideline on grading rubrics will be posted.

- Questions?