CS 6515 · Exam 3 (Randomization & Hardness): iPad problem set

109 problems, in the same order and with the same numbers as the ✍ boxes in **exam_notes.html**. **All solutions are at the end**, after the last problem. Write your answer before you look.

- ★ = tied to something the professor flagged in lecture (85 problems). The insight IDs (I01–I79) match the “What the professor flagged” section of the notes.
- Reductions: always write the five parts. (1) In NP, (2) reduce FROM a known NP-hard problem, (3) the transformation f, (4) ⇒ and ⇐, (5) polynomial time and solution recovery g.
- Assumed exam rules (Unit 1): about 75 min, one page of notes, no calculator. Numerical answers are kept hand-computable.
- Section M is a full mock exam. Do it timed (75 min) on the day before the exam.

3 · Probability toolkit, Las Vegas vs Monte Carlo

Notes §3 · Oct 29 lecture · MU 2.1, 2.2, 2.4, 3.1

★ 3.1 Las Vegas or Monte Carlo? ⚑ I09

Source: Lecture question (Oct 29: “Las Vegas or Monte Carlo?”) + slide examples

Classify each algorithm as Las Vegas or Monte Carlo and justify in one line.

a. Randomized Quicksort (random pivots).
b. Fermat's test: pick random a, answer “prime” iff $a^{p-1} \equiv 1 \pmod{p}$.
c. Karger's contraction algorithm (one run).
d. Repeat “random cut; if it has ≥ m/2 edges, output it” until it outputs.
e. Fair-coin assignment for MAX-SAT, output whatever you get.

3.2 Bernoulli, binomial, geometric by hand

Source: Lecture definitions (Oct 29) + MU §2.2, §2.4

A trial succeeds with probability p = 1/4, independently.

a. X = number of successes in 4 trials. Compute Pr[X = 2], Pr[X = 0], and E[X].
b. Y = number of trials until the first success. Compute Pr[Y = 3], Pr[Y > 3], and E[Y].
c. Which of X, Y describes “run Karger until the first success”?

★ 3.3 Expected number of heads, and the binomial mean ⚑ I11, I12

Source: Lec 16 typed notes, “Quick example” (slide exercise)

Toss a fair coin n times and let X count heads. Without enumerating the 2^n outcomes, find E[X]. Then generalize to get the expectation of a Binomial(n, p) random variable.

★ 3.4 What Markov does and doesn't tell you ⚑ I14

Source: Lec 16 typed notes, “Check for understanding” (verbatim)

Suppose E[X] = 6 and X ≥ 0. What does Markov say about P(X ≥ 24)? Can it tell us P(X ≥ 4) is small?

★ 3.5 Prove Markov's inequality ⚑ I10

Source: Lecture: “It's very easy to prove, one line (it's in the homework)”. Proof from MU Thm 3.1

Let X ≥ 0 be a random variable and a > 0. Prove Pr[X ≥ a] ≤ E[X]/a.

3.6 Quicksort and Markov

Source: Lecture example (Oct 29) + variant

Let X be the number of comparisons of randomized Quicksort on n items.

a. If $E[X] = n \log n$, bound $\Pr[X \geq 10\, n \log n]$.
b. If instead $E[X] = 2\, n \ln n$, bound $\Pr[X \geq 10\, n \ln n]$.
c. Does (a) say that Quicksort is $O(n \log n)$ in the worst case?

★ 3.7 True/False: probability tools ⚑ I13

Source: Exam-style (Unit 1 short-answer format), built from Lec 16/17 typed notes

a. Linearity of expectation $E[X + Y] = E[X] + E[Y]$ requires X and Y to be independent.
b. Markov's inequality applies to every random variable with finite expectation.
c. For a 0/1 random variable I, $E[I] = \Pr[I = 1]$.
d. The probabilities in randomized-algorithm guarantees are taken over a random input distribution.
e. A Monte Carlo algorithm has a bounded running time but may output a wrong answer.
f. If $X \sim \text{Geo}(p)$, then $E[X] = 1/p$.

For each: True or False, with a 1–2 line justification (Unit 1 exam format).

3.8 Jensen's inequality (textbook only)

Source: MU §2.1.2 (assigned reading; not covered in lecture)

A square's side X is chosen uniformly from {1, 2, …, 99}. Compare $E[X^2]$ with $(E[X])^2$, and state Jensen's inequality.

3.9 Memoryless + coupon collector (textbook only)

Source: MU §2.4, Lemma 2.8–2.9, MU §2.4.1 (assigned reading; not covered in lecture)

a. For $X \sim \text{Geo}(p)$, show $\Pr[X = n + k \mid X > k] = \Pr[X = n]$ (memoryless).
b. Each cereal box holds one of n coupon types, uniformly at random. Use linearity and geometric expectations to find E[# boxes to collect all n]. Evaluate for n = 5 and n = 10.

4 · Karger's min-cut (watermelon) and boosting

Notes §4 · Oct 29 + Nov 3 · MU 1.5 (1st-ed. 1.4)

★ 4.1 Karger on the worked-example graph ⚑ I15

Source: Lecture analysis (Oct 29) applied to the §4.2 graph

Use the graph in §4.2 (two triangles {1,2,3}, {4,5,6} joined by bridges 3–4 and 2–5; n = 6, m = 8) and the fixed min cut Δ = {3–4, 2–5}.

a. What is the probability that the first contraction avoids Δ? Compare it with the bound $1 - 2/n$ used in the proof.
b. Check the degree bound $\deg(v) \geq |\Delta|$ for every vertex.
c. State the theorem's lower bound on Pr[return Δ]. The exact value (computed by enumerating all runs) is 6/35. Is that consistent?
d. Why is Pr[return *some* min cut] (19/35) larger?

★ 4.2 Reproduce the Karger bound (full proof) ⚑ I15

Source: Lecture derivation (Oct 29 board + Lec 16 typed notes): be able to write it from memory

Let G be a connected multigraph on n vertices and Δ a fixed minimum cut. Prove that one run of the contraction algorithm returns Δ with probability at least $2/(n(n-1))$.

4.3 Why keep parallel edges?

Source: Lec 16 typed notes (“parallel edges must be retained”) + MU §1.5

In the contraction step we delete self-loops but keep parallel edges. Give two reasons the analysis needs the parallel edges.

★ 4.4 How many runs for n = 10? ⚑ I17

Source: Exam-style variant of the Nov 3 boosting calculation

Karger on n = 10 vertices succeeds with probability at least $p = 2/(n(n-1))$.

a. Compute p.
b. Lower-bound the success probability after N = 45 and after N = 100 independent runs (keeping the smallest cut).
c. Give a number of runs that guarantees success ≥ 0.99, using $(1 - p)^N \le e^{-pN}$.

★ 4.5 n = 1000: the lecture's numbers ⚑ I17, I20

Source: Nov 3 lecture ("If n is 1,000 and you pick α = 1...") + Lec 17 typed notes

For n = 1000, $p = 2/(n(n-1))$.

a. Show that $N = \alpha \cdot n(n-1) \cdot \ln n$ runs give failure $\le n^{-2\alpha}$. Evaluate for α = 1.
b. The typed notes use $N = \lceil \alpha \cdot n(n-1)/2 \cdot \ln n \rceil$ and get failure $\le n^{-\alpha}$. Reconcile the two.
c. How many runs is that, and is it polynomial?

★ 4.6 Slide exercise: from $1/n^2$ to failure $1/n^3$ ⚑ I17

Source: Lec 17 typed notes, Exercise 1 (verbatim)

If one randomized trial succeeds with probability at least $1/n^2$, how many independent repetitions suffice to make failure probability at most $1/n^3$?

★ 4.7 “Run until it outputs a min cut”? ⚑ I18

Source: Nov 3 lecture: “I'm going to put ‘until’ in red because we'll come back to that”

The first idea was to “run Karger until it outputs a min cut”. Why doesn't that work as an algorithm, and what do we do instead? What property of cuts makes the fix work?

★ 4.8 True/False: Karger and boosting ⚑ I17, I16

Source: Exam-style, from Oct 29 / Nov 3 lectures

a. The watermelon algorithm is a Las Vegas algorithm.
b. If an algorithm succeeds with probability $p = 1/n^5$ and runs in polynomial time, polynomially many repetitions give success probability $1 - o(1)$.
c. If an algorithm succeeds with probability 2^{-n}, repeating it polynomially many times gives success $1 - o(1)$.
d. Repeating Karger $n(n-1) \ln n$ times and keeping the smallest cut always returns a minimum cut.
e. In Karger's analysis, the probability that the k-th contraction avoids Δ (given the past was good) gets smaller as k grows.
f. Global min-cut can also be solved deterministically in polynomial time.

For each: True or False, with a 1–2 line justification (Unit 1 exam format).

4.9 The cycle shows the bound is tight

Source: Derived example (computed by full enumeration in calc.py); the corollary is a derivation, not stated in lecture

Let $G = C_5$ (cycle on 5 vertices). Every pair of edges is a min cut (size 2).

a. How many min cuts does C_5 have? Compare Pr[return a fixed one] with $2/(n(n-1))$.
b. Using only the theorem, show that every graph has at most $n(n-1)/2$ minimum cuts.

5 · Probabilistic method and derandomization

Notes §5 · Nov 3 · MU 6.1–6.4

★ 5.1 If E[X] = μ then both sides have positive probability ⚑ I21

Source: Nov 3 lecture: “I will not prove this, because it's in your homework”. Proof from MU Lemma 6.2

Let X be a discrete random variable on a probability space with E[X] = μ. Prove Pr[X ≥ μ] > 0 and Pr[X ≤ μ] > 0.

★ 5.2 Expected size of a random cut ⚑ I12, I13

Source: Nov 3 lecture / Lec 17 typed notes §2.1 / MU Thm 6.3

Each vertex of G = (V, E), |E| = m, is placed in A or B by an independent fair coin.

a. Show that each edge crosses with probability 1/2.
b. Show E[|C(A,B)|] = m/2.
c. Conclude that every graph has a cut with at least m/2 edges, and that this gives an expected ½-approximation for Max-Cut.

★ 5.3 From expectation to probability (max-cut) ⚑ I22, I23

Source: Nov 3 lecture / Lec 17 typed notes §2.2 / MU §6.2.1

Let $X = |C(A,B)|$ for the random cut, $p = \Pr[X \geq m/2]$.

a. Show $p \geq 1/(m/2 + 1)$ when m is even.
b. What does the same argument give for odd m? Evaluate both for $m = 8$ and $m = 9$.
c. Turn this into an algorithm that finds a cut of size $\geq m/2$. Is it Las Vegas or Monte Carlo, and what is its expected number of samples?

★ 5.4 Derandomize max-cut by hand ⚑ I24

Source: Nov 3 lecture (greedy placement) on the §5.4 graph D

Graph D has vertices v1..v6 and edges v1v2, v1v3, v2v3, v2v4, v3v5, v4v5, v4v6, v5v6, v1v6 (m = 9). Process the vertices in order v1, …, v6. At each step compute E[cut | placement] for "put v_k in A" and "put v_k in B", choose the larger (ties → A), and report the final cut size.

★ 5.5 Slide exercise: the majority rule directly ⚑ I24

Source: Lec 17 typed notes, Exercise 3 (verbatim)

Prove directly that assigning each vertex opposite to a majority of its already assigned neighbors produces at least m/2 crossing edges.

★ 5.6 Slide exercise: the indicators are dependent ⚑ I13

Source: Lec 17 typed notes, Exercise 2 (verbatim) + triangle computed in calc.py

Explain why the edge indicators in random Max-Cut need not be independent, even though their expectations can be added. Use a triangle.

5.7 MAX-SAT existence (probabilistic method)

Source: MU Thm 6.4 (textbook) + LP4 §7

Show that any CNF formula with m clauses, each with at least k distinct literals (no clause containing both x and ¬x), has an assignment satisfying at least $m(1 - 2^{-k})$ clauses. How many for m = 40 clauses of exactly 3 literals?

5.8 Derandomize MAX-SAT by hand

Source: LP4 §7 (method of conditional expectations over variables)

Formula: (x1 ∨ x2) ∧ (¬x1 ∨ x3) ∧ (¬x2 ∨ ¬x3) ∧ (x1) ∧ (¬x1 ∨ ¬x2 ∨ x3). Fix x1, x2, x3 in order. At each step compute the expected number of satisfied clauses if the remaining variables were fair coins, for both choices. Pick the larger (ties → T).

★ 5.9 True/False: probabilistic method ⚑ I25, I26

Source: Exam-style, from Nov 3 lecture

a. The probabilistic method proof that a cut of size ≥ m/2 exists also tells us how to find it efficiently.
b. The derandomized max-cut algorithm returns the same cut no matter in which order the vertices are processed.
c. Max-Cut (find a largest cut) can be solved in polynomial time, just like min-cut.
d. In the method of conditional expectations, $E[\text{cut} \mid x_1] = m/2$ for either placement of the first vertex.
e. If $\Pr[X \geq m/2] \geq 1/(m/2+1)$, then about $m/2 + 1$ samples are needed in expectation to see a cut of size ≥ m/2.
f. The derandomized greedy algorithm for max-cut is a ½-approximation.

For each: True or False, with a 1–2 line justification (Unit 1 exam format).

5.10 Union bound: 2-coloring K_{1000} (textbook only)

Source: MU §6.1, Thm 6.1 (assigned reading; not covered in lecture)

Color each edge of K_n red or blue by a fair coin. Use the union bound to show that if $C(n,k) \cdot 2^{-C(k,2)+1} < 1$, some coloring has no monochromatic K_k. Check n = 1000, k = 20.

5.11 **Sample and modify: independent sets (textbook only)**

Source: MU §6.4.1, Thm 6.5 (assigned reading; not covered in lecture)

G has n = 100 vertices and m = 300 edges. Using MU's sample-and-modify argument (keep each vertex with probability 1/d, d = 2m/n, then delete one endpoint of each surviving edge), what independent-set size is guaranteed to exist?

6 · Randomized rounding: MAX-SAT

Notes §6 · Nov 5 (OMS LP4) · Lec 18 typed notes

6.1 Check the ILP clause constraint

Source: Lec 18 typed notes, Exercise 1 (verbatim)

Verify the ILP constraints on the clause (x1 ∨ ¬x2 ∨ x3) for one satisfying and one unsatisfying assignment.

6.2 Write the ILP and the LP relaxation

Source: LP4 §10–14 / Lec 18 typed notes §1, §3, on the Lec 18 example formula

For $C_1 = (x_1 \vee \neg x_2)$, $C_2 = (x_2)$, $C_3 = (\neg x_1 \vee \neg x_2)$:

a. Write the ILP for MAX-SAT and its LP relaxation.
b. OPT = ? (brute force) LP* = ? (the LP solver gives an optimal solution $\hat{y}_1 = 0.5$, $\hat{y}_2 = 0.5$, $\hat{z}_1 = 1$, $\hat{z}_2 = 0.5$, $\hat{z}_3 = 1$)
c. Why must LP* ≥ OPT?
d. Compute the expected number of satisfied clauses under fair coins and under LP rounding with this $\hat{y}$.

6.3 One clause, fractional values

Source: LP4 §17–21 lemma, numeric instance

LP solution $\hat{y}_1 = 0.5$, $\hat{y}_2 = 0.3$, $\hat{y}_3 = 0.2$, $\hat{z} = 1$ for the clause $(x_1 \vee x_2 \vee x_3)$.

a. Exact Pr[clause satisfied] after randomized rounding?
b. Compare with the AM–GM bound $1 - (1 - \hat{z}/k)^k$, with $\beta_3\hat{z}$, and with $(1 - 1/e)\hat{z}$.
c. Now the clause $(\neg x_1 \vee \neg x_2)$ with $\hat{y}_1 = 0.9$, $\hat{y}_2 = 0.6$. What is the largest feasible $\hat{z}$, and Pr[satisfied]? Check it is $\geq \beta_2\hat{z}$.

6.4 Slide exercise: concavity without derivatives

Source: Lec 18 typed notes, Exercise 2 (verbatim)

Prove that $1 - e^{-z} \geq (1 - 1/e)z$ for $0 \leq z \leq 1$ without differentiation, using concavity.

★ 6.5 Slide exercise: why 7/8 needs exactly 3 literals ⚑ I28

Source: Lec 18 typed notes, Exercise 3 (verbatim)

Explain why the 7/8 fair-coin proof does not apply to MAX-3SAT if clauses of size one or two are permitted.

6.6 Fill the comparison table

Source: LP4 §23 table

For clause size k = 1, 2, 3, 4, give the fair-coin guarantee $\alpha_k = 1 - 2^{-k}$, the LP-rounding factor $\beta_k = 1 - (1 - 1/k)^k$, the better of the two, and $\alpha_k + \beta_k$.

★ **6.7** **The ¾-approximation, proved correctly** ⚑ I30

Source: LP4 §24 + the transcript's editorial correction (the video's per-clause argument is not a valid proof)

Algorithm: run fair coins (M_1 satisfied) and LP rounding (M_2 satisfied); output the better assignment. Prove $E[\max(M_1, M_2)] \geq \frac{3}{4} \cdot OPT$, using $\alpha_k + \beta_k \geq 3/2$ for all $k \geq 1$.

6.8 Prove the rounding lemma

Source: LP4 §17–21 / Lec 18 typed notes §3

Let $(\hat{y}, \hat{z})$ be an optimal LP solution and round each x_i to true independently with probability $\hat{y}_i$. Prove that every clause C_j satisfies $\Pr[C_j \text{ satisfied}] \geq (1 - 1/e)\hat{z}_j$, and conclude $E[\#\text{satisfied}] \geq (1 - 1/e) \cdot \text{OPT}$.

6.9 Rounding on formula R

Source: Worked instance (LP solved with scipy in calc.py; same formula as the simulator widget)

$R = (x1) \wedge (x2) \wedge (\neg x1 \vee \neg x2) \wedge (x1 \vee \neg x3) \wedge (x3 \vee \neg x2) \wedge (\neg x3) \wedge (x2 \vee x3 \vee \neg x1)$. The LP optimum is $LP^* = 5.66667$ with $\hat{y}_1 = 0.666667$, $\hat{y}_2 = 0.333333$, $\hat{y}_3 = 0.333333$.

a. Brute-force OPT is 5. Confirm $LP^* \geq \text{OPT}$ makes sense.
b. Expected satisfied clauses: fair coins = 37/8; LP rounding = 4.8148. Which guarantees do these numbers respect?
c. What does the best-of-two algorithm guarantee here?

★ 6.10 Why not round deterministically? ⚑ I27

Source: Derived example (recipe practice for “another problem”, see the Nov 3 homework remark)

Instead of coins, set $x_i = T$ iff $\hat{y}_i \geq ½$. Give a clause where this can satisfy nothing even though $\hat{z} = 1$, while randomized rounding satisfies it with probability $\geq 1 - 1/e$.

★ 6.11 True/False: rounding and MAX-SAT ⚑ I29, I31, I28

Source: Exam-style, from LP4 + Lec 18 typed notes

a. Each run of LP-guided rounding satisfies at least $(1 - 1/e)\cdot OPT$ clauses.
b. For a maximization ILP, the LP relaxation's optimal value is at most the ILP's optimal value.
c. In MAX-SAT randomized rounding we must also round the z_j variables.
d. A polynomial-time 0.88-approximation for MAX-E3SAT would imply P = NP.
e. Fair coins give a 7/8-approximation for MAX-3SAT when clauses have at most 3 literals.
f. Integer linear programming is NP-hard, while linear programming is in P.

For each: True or False, with a 1–2 line justification (Unit 1 exam format).

★ **6.12** **Strongest bounds on OPT (Q1(f) drill)** ⚑ I79

Source: Exam 2 Q1(f) format (8 points); my numbers

For each, give the strongest lower and upper bound on OPT that follows, with one sentence of justification.

a. A Set Cover instance: the LP relaxation has optimal value 7.4, and randomized rounding + repetition returned a cover with 19 sets.
b. MAX-SAT with m = 30 clauses, every clause with exactly 3 distinct literals. One run of the LP-rounding algorithm (expected (1 − 1/e)-approximation) satisfied 24 clauses.
c. Vertex Cover: a 2-approximation returned 14 vertices, and someone shows you an independent set of size 5 in this 20-vertex graph.

9 · P, NP, NP-hard, reductions

Notes §9 · Nov 10

★ 9.1 Definitions, in the course's words ⚑ I37, I38

Source: Nov 10 lecture / board

Define: search problem, NP, P, NP-hard, NP-complete. Then answer: is every NP-hard problem in NP?

★ 9.2 In NP or not (course definition)? ⚑ I37

Source: Nov 10 lecture examples (shortest path, independent set, knapsack) + LP4 §1

Using “NP = search problems with a poly-time verifier”, decide whether each is in NP, and say whether it is known to be in P.

a. Shortest s–t path.
b. Independent set: given G and k, find an independent set of size ≥ k.
c. Knapsack as in class: find the most valuable subset within capacity B.
d. Max-SAT: find an assignment maximizing the number of satisfied clauses.
e. 2-SAT.

★ 9.3 Direction of the reduction ⚑ I33

Source: Nov 10 Q&A: “that's a great multiple-choice question to think about”

You want to prove a new problem B is NP-hard and you know A is NP-hard. Which do you show?

a. B reduces to A.
b. A reduces to B.
c. Either direction works.
d. Both directions are needed.

Also: shortest path reduces to SAT. Does SAT reduce to shortest path?

★ 9.4 What follows from A → B? ⚑ I40, I03

Source: Nov 12 summary (“what does a reduction preserve?”)

Suppose A reduces to B in polynomial time. For each extra fact, state what you can conclude (or “nothing”).

a. A is NP-hard.
b. B $\in$ P.
c. A $\in$ P.
d. B is NP-hard.
e. B $\in$ NP and A is NP-hard.

★ 9.5 The dramatic case ⚑ I56

Source: Nov 12: “if A is NP-hard and B is in P, then?”

A is NP-hard, B ∈ P, and A reduces to B. What follows?

★ 9.6 Why is this “reduction” invalid? ⚑ I36

Source: Nov 10 lecture (the K_{2026} magic trick)

“Reduce” any problem A to shortest path: f maps every input I of A to the complete graph on 2026 vertices with s = 1, t = 2. Run shortest path. Why is this not a reduction?

★ 9.7 True/False: P, NP, reductions ⚑ I33, I35, I38, I39, I34

Source: Exam-style, from Nov 10 Q&A

a. In a reduction A → B, the map f must be injective (one-to-one).
b. NP-hard problems cannot be solved in polynomial time.
c. P ⊆ NP.
d. Every NP-hard problem belongs to NP.
e. The functions f and g in a reduction can be chosen independently of A and B.
f. If one NP-complete problem has a polynomial-time algorithm, then P = NP.
g. If A → B and B → C (poly-time reductions), then A → C.

For each: True or False, with a 1–2 line justification (Unit 1 exam format).

9.8 Why polynomial time composes

Source: Nov 10 lecture (“composing efficient steps is efficient”)

A reduction's f runs in $O(n^2)$ time on inputs of size n, and B can be solved in $O(N^3)$ time on inputs of size N. Bound the time to solve A this way (ignore g). Why is this the whole reason reductions use polynomial time?

★ 9.9 Decision vs. search (self-reducibility) ⚑ I57

Source: Nov 12 opening (“for your own entertainment”) + KT §8.1 binary-search remark; the vertex-deletion step is a derivation, not from the materials

Suppose a black box answers “does G have an independent set of size ≥ k?”.

a. Find the size of a maximum independent set with O(log n) calls.
b. Then find an actual maximum independent set with polynomially many calls.

★ 9.10 Knapsack-search and binary search ⚑ I72

Source: OMS NP1 video, sections 15–17 (extra material); example numbers simulated in calc.py

Knapsack-search: input weights $w_1..w_n$, values $v_1..v_n$, capacity B and a goal g. Output a subset S with $\Sigma_{i\in S} w_i \le B$ and $\Sigma_{i\in S} v_i \ge g$, or NO.

a. (2) Show knapsack-search is in NP.
b. (4) Suppose a black box solves knapsack-search in polynomial time. Solve optimization knapsack in polynomial time and bound the number of calls in terms of $V = \Sigma v_i$. Why is that polynomial?
c. (1) Why does the definition say value ≥ g and not value ≤ g?
d. (3) Run your method on (w, v) = (3, 4), (4, 5), (2, 3), (5, 8), B = 9: how many calls, and what is the optimum?
e. (2) Is optimization knapsack in NP? Answer in the course's terms.

★ **9.11** **Select all: search problems, P and NP** ⚑ I71, I73, I74, I70

Source: OMS NP1 video quizzes (MST, knapsack, SAT verification, the NP acronym, direction) in the Exam 2 select-all format

Select ALL that apply.

- ○ MST is in NP.
- ○ MST is in P.
- ○ Knapsack (optimization, as in class) is known to be in NP.
- ○ “NP” stands for “not polynomial time”.
- ○ Checking that an assignment satisfies a CNF formula with n variables and m clauses takes O(nm) time.
- ○ If P ≠ NP, then no NP-complete problem can be solved in polynomial time.
- ○ To show B is NP-complete it suffices to show B ∈ NP and B reduces to SAT.

10 · SAT, Max-SAT, 2-SAT, 3-SAT

Notes §10 · Nov 10

★ 10.1 Why is Max-SAT hard? (predicted exam question) ⚑ I32

Source: Nov 10: “Maybe I'll ask this on the exam”

Prove that Max-SAT is NP-hard, in the reduction format.

10.2 ILP is NP-hard

Source: LP4 §10–13 + Nov 10 (“ILP solves max-SAT, which solves SAT”)

Show ILP is NP-hard via Max-SAT → ILP. Describe f and g and argue correctness briefly.

★ 10.3 2-SAT on the board example ⚑ I43

Source: Nov 10 board: f = (x1∨x2)∧(x1∨¬x3)∧(¬x2∨¬x4)∧(¬x3∨x4)

For f = (x1 ∨ x2) ∧ (x1 ∨ ¬x3) ∧ (¬x2 ∨ ¬x4) ∧ (¬x3 ∨ x4):

a. List the 8 edges of the implication graph.
b. Find the SCCs. Is f satisfiable?
c. Give a satisfying assignment.

★ 10.4 An unsatisfiable 2-SAT formula ⚑ I43

Source: Derived example (verified by the 2-SAT solver in calc.py)

Show (x1 ∨ x2) ∧ (x1 ∨ ¬x2) ∧ (¬x1 ∨ x2) ∧ (¬x1 ∨ ¬x2) is unsatisfiable using the implication graph.

★ 10.5 Why SCCs decide 2-SAT ⚑ I43

Source: Nov 10 lecture (implication encoding + SCC observation)

(a) Show $(a \vee b) \equiv (\neg a \to b) \wedge (\neg b \to a)$. (b) Show that in any satisfying assignment, all literals in one SCC have the same value. (c) Conclude: if x and $\neg x$ share an SCC, the formula is unsatisfiable.

★ 10.6 Clause splitting: SAT → 3-SAT ⚑ I45

Source: Nov 10 board (5-literal clause) + general case

(a) Write C' for $C = (x1 \vee x2 \vee x3 \vee x4 \vee x5 \vee x6 \vee x7)$. (b) Prove C satisfiable $\Leftrightarrow$ C' satisfiable (for the general k-literal case). (c) How many clauses and new variables for a 10-literal clause? Why is the whole reduction polynomial?

★ 10.7 Why doesn't the trick give 2-SAT? ⚑ I46

Source: Nov 10: “Why not 2SAT? I need this intermediate structure”

Why can't the same splitting idea reduce SAT to 2-SAT?

10.8 3-SAT is NP-complete

Source: Nov 10 + KT §8.4 (in-NP part)

State the two things needed to show 3-SAT is NP-complete and give each in one or two sentences.

10.9 k-SAT for k ≥ 3

Source: Exam-style practice (definition of k-SAT from the Nov 10 board: at most k literals per clause)

With k-SAT defined as “at most k literals per clause”, show k-SAT is NP-hard for every fixed k ≥ 3.

★ 10.10 Exact-4-SAT is NP-complete (DPV 8.8) ⚑ I76

Source: Named as practice in the OMS NP2 video (DPV 8.8); padding gadget is my construction, brute-force verified on 150 random formulas in calc.py

Exact-4-SAT: a CNF formula in which every clause has exactly 4 literals on 4 distinct variables. Find a satisfying assignment or output NO. Prove Exact-4-SAT is NP-complete.

★ 10.11 2-SAT: eliminate unit clauses (GR2 example) ⚑ I78

Source: OMS GR2 video, section 5 (Simplifying Input); unique satisfying assignment checked by brute force in calc.py

$f = (x_3 \vee \neg x_2) \wedge (\neg x_1) \wedge (x_1 \vee x_4) \wedge (\neg x_4 \vee x_2) \wedge (\neg x_3 \vee x_4)$.

a. Apply unit-clause elimination step by step until no unit clause is left. Write each intermediate formula.
b. Is f satisfiable? Give the assignment.
c. Why is the formula after one elimination step satisfiable ⇔ the formula before it is?
d. What should the algorithm do if elimination ever produces an empty clause?

★ 10.12 Stingy SAT is NP-complete (DPV 8.3) ⚑ I76

Source: Named as practice in the OMS NP2 video (DPV 8.3); solution is the standard one-line reduction

Stingy SAT: given a CNF formula φ and an integer k, find a satisfying assignment with *at most k* variables set to true, or output NO. Prove it is NP-complete.

★ 10.13 Why the 2-SAT assignment rule works (key fact) ⚑ I77

Source: OMS GR2 video, sections 11–14 (extra material); part (c)'s clause check is spelled out by me

In the implication graph of a 2-SAT formula (every clause $(a \vee b)$ gives edges $\neg a \to b$ and $\neg b \to a$), for a set S of literals let $\bar{S} = \{\neg \ell : \ell \in S\}$.

a. Show: if there is a path $\alpha \rightsquigarrow \beta$, then there is a path $\neg\beta \rightsquigarrow \neg\alpha$.
b. Show: if S is an SCC then $\bar{S}$ is an SCC; and if S is a sink SCC then $\bar{S}$ is a source SCC.
c. Assume no x shares an SCC with $\neg x$. The algorithm sets every literal of a sink SCC S true (so $\bar{S}$ false), removes $S \cup \bar{S}$ and repeats. Show every clause that contains a literal of $S \cup \bar{S}$ is satisfied.

11 · Graph reductions: IS, Clique, VC, matching

Notes §11 · Nov 12

★ 11.1 Build the 3-SAT → IS graph ⚑ I50

Source: Nov 12 construction applied to φ (own example: the lecture's exact formula isn't recoverable from the recording)

φ = (x) ∧ (¬x ∨ y) ∧ (¬y ∨ z ∨ x).

a. List the vertices, the clause-clique edges, the complement edges, and k.
b. Find an independent set of size k and the assignment it gives.
c. Can S = {x@C1, ¬x@C2, z@C3} be chosen? Why not?

★ 11.2 Prove Independent Set is NP-hard (full write-up) ⚑ I47, I48, I49, I50

Source: Nov 12 lecture (three steps: transformation, ⇔, recovery) + KT 8.8

Prove Independent Set (search version: given G, k, find an independent set of size k) is NP-complete, by reducing from 3-SAT.

11.3 **Independent set ⇔ clique in the complement**

Source: Nov 12 lecture (Ĝ construction)

(a) Prove: S is independent in G ⇔ S is a clique in Ĝ (same vertices, complementary edges). (b) Use it to show Clique is NP-complete. (c) Does the same idea give Clique → IS?

★ 11.4 Independent set ⇔ complement is a vertex cover ⚑ I52, I53

Source: Nov 12 lecture + KT (8.3)–(8.5)

(a) Prove S is independent ⇔ V\S is a vertex cover. (b) Give IS → VC and VC → IS. (c) “Is it always true that the complement of a vertex cover is an independent set?”

★ 11.5 Do reductions preserve approximation? ⚑ I54

Source: Nov 12 lecture (star) + the transcript editor's perfect-matching example

Vertex cover has a 2-approximation. IS → VC is an exact reduction (S ↦ V\S). Does complementing a 2-approximate vertex cover give an approximate independent set? Prove your answer with an example.

★ 11.6 Bipartite matching via max flow ⚑ I55

Source: Nov 12 lecture

(a) Describe the network built from a bipartite graph (V_1, V_2, E). (b) Prove: for an integral flow f, $M = \{(u,v) \in V_1 \times V_2 : f(u,v) = 1\}$ is a matching. (c) What does this reduction prove about bipartite matching?

★ 11.7 Lecture Q&A on the IS construction ⚑ I50

Source: Nov 12 student questions

(a) “What if more than one literal in a clause is true?” (b) “What if you pick x, y and ¬y?” (c) Why k = m and not more?

11.8 Vertex Cover → Set Cover (textbook only)

Source: KT (8.6) (Kleinberg–Tardos stands in for the missing DPV ch. 8)

Set Cover: given a universe U, subsets S_1, …, S_m and k, are there ≤ k subsets whose union is U? Show VC → Set Cover.

★ 11.9 True/False: graph reductions ⚑ I41, I54

Source: Exam-style, from Nov 12 lecture

a. Since 3-SAT → IS, the reduction IS → SAT given by Cook–Levin is just the 3-SAT → IS construction run backwards.
b. S is a maximum independent set of G if and only if V \ S is a minimum vertex cover of G.
c. A 2-approximation for vertex cover yields a 2-approximation for independent set via complements.
d. In 3-SAT → IS, if the graph has an independent set of size m, the formula is satisfiable.
e. Reducing bipartite matching to max flow shows bipartite matching is NP-hard.
f. Clique and Independent Set reduce to each other.

For each: True or False, with a 1–2 line justification (Unit 1 exam format).

12 · Writing reductions: template drills

Notes §12

★ 12.1 Vertex Cover is NP-complete (full template) ⚑ I47, I45

Source: Template drill (Nov 12 IS ↔ VC + KT 8.4)

Write a complete NP-completeness proof for Vertex Cover (given G, k: find a vertex cover of size ≤ k), using all five parts of the template.

★ 12.2 HALF-IS: independent set of exactly half the vertices ⚑ I42

Source: Exam-style practice problem (not from the materials). The padding construction is checked by brute force on 300 random graphs in calc.py

HALF-IS: given a graph G' with an even number n' of vertices, find an independent set of size $n'/2$ (or report none). Prove HALF-IS is NP-complete.

12.3 Set Packing is NP-hard

Source: KT (8.7) (stated, proof left as an exercise there). Full solution here

Set Packing: given U, subsets $S_1, \ldots, S_m$ and k, find $\geq k$ pairwise disjoint subsets. Show IS $\rightarrow$ Set Packing.

★ 12.4 Which problem should I reduce from? ⚑ I42

Source: Decision guide practice (KT §8.10 genres + lecture reductions)

Pick the most natural source problem for each new problem and name the gadget idea.

a. Choose ≥ k servers so that no two conflicting servers are both chosen.
b. Choose ≤ k cameras so that every corridor (a pair of rooms) is watched by a camera in one of its rooms.
c. Find k people who all know each other.
d. A Boolean constraint problem with ≤ 4 literals per clause.
e. Satisfy as many constraints as possible.

★ 12.5 Spot the broken reduction ⚑ I33, I45

Source: Exam-style error analysis (common mistakes from the lecture warnings)

A student “proves” 2-SAT is NP-hard: “Take any 2-SAT formula and run the 3-SAT → IS construction; IS is NP-hard, so 2-SAT is NP-hard.” A second student proves Clique is NP-hard by showing that from a satisfying assignment of φ you get a clique in $\hat{G}$, and stops there. What is wrong in each?

15 · HW5 decoded: originals (hints only) + exam-style re-skins

Notes §15 · homeworks/hw5.pdf · how Exam 2 re-skinned HW4

★ 15.1 HW5 #19: Min cuts (original wording, full solution) ⚑ I66, I63

Source: homeworks/hw5.pdf, Problem 19 (verbatim)

For this problem, let G = (V, E) be a connected, undirected graph. A min cut is a partition (S, T) of the vertices such that the number of edges from S to T is minimized.

1. Show that a graph has at most $O(n^2)$ min-cuts. (There are many ways to do this, a probabilistic argument is desired. The Watermelon algorithm may be of inspiration.)
2. Let $\alpha > 1$ be a constant. A cut is $\alpha-$small if it has size at most $\alpha|(S, T)|$, for any min-cut (S, T). Prove that the number of $\alpha-$small cuts is $O_\alpha(n^{2\alpha})$. (Hint: Stop your watermelon earlier!)

★ 15.2 HW5 #20: Linearity of expectation (original wording, full solution)

⚑ I68, I65

Source: homeworks/hw5.pdf, Problem 20 (verbatim)

Recall the Max-SAT problem. The input is a Boolean formula in conjunctive normal form and you wish to output an assignment of the variables that satisfy the maximum number of clauses. This problem is (famously) known to be NP-hard.

a. Consider an input of MAX-Exact-3SAT (every clause is the disjunction of exactly three literals). Show that a random assignment will satisfy at least 87.5% of the clauses in expectation.
b. Show that for MAX-Exact-3SAT, if there is an assignment that satisfies strictly less than 87.5% of the clauses, then there is another assignment that satisfies strictly more than 87.5% of the clauses. Hint: this is the Probabilistic Method idea discussed in lecture.
c. Give an example of an input of MAX-Exact-3SAT such that every assignment satisfies exactly 87.5% of the clauses. Hint: $2^3 = 8$.

★ 15.3 HW5 #21: Randomized rounding for Hitting Set (original wording, full solution) ⚑ I67, I64

Source: homeworks/hw5.pdf, Problem 21 (verbatim)

Input: Subsets $S_1, S_2, \ldots, S_m \subset \{1, 2, \ldots, n\}$. Output: the smallest set S such that $S \cap S_k \neq \emptyset$ for all $1 \leq k \leq m$. (S does not have to be one of the S_i.)

a. Write an instance of ILP that is equivalent to Hitting-Set.
b. As in class, consider the corresponding LP relaxation. Let $\{x^*_i\}$ be an optimal of the relaxation. Define $x_i = 1$ with probability x^*_i, else 0 (i.e.: with probability $1 - x^*_i$), and

set $S = \{i \mid x_i = 1\}$. Compute $P(S \cap S_j \neq \emptyset)$. Briefly justify your answer.

c. Conclude that $P(S \cap S_j \neq \emptyset) \geq 1 - 1/e$. You can use the Arithmetic Mean-Geometric Mean inequality as well as others we used in class without proof.

d. Consider the following algorithm: sample S repeatedly as in part (b), take the union until you hit all sets. Show that it is enough to do $O(\ln(m))$ rounds in expectation. Hint: use (c) to bound the probability that a set S_i is not covered after $\ln(m)$ rounds. Conclude using linearity of expectation.

e. Prove this algorithm yields a $\ln(m)-$Approximation algorithm for Hitting-Set.

★ 15.4 HW5 #22: Zero-knowledge proofs and 3-coloring (original wording, full solution) ⚑ I63

Source: homeworks/hw5.pdf, Problem 22 (verbatim, protocol summarized in §15.4)

Raquel knows a proper 3-coloring C of G and repeats N rounds: (1) choose a uniformly random permutation π of the three colors, C′(v) = π(C(v)); (2) commit to every vertex's color (perfectly hiding and binding); (3) Brito selects an edge {u, v} uniformly at random; (4) Raquel reveals the colors of u and v, and Brito accepts the round if they differ. Brito accepts only if he accepts all N rounds.

a. Explain why Brito learns no information about Raquel's particular coloring C, beyond what is implied by the fact that G is 3-colorable (this is the Zero knowledge part). In particular, indicate the distribution of the two colors revealed in a round.
b. Suppose G is not 3-colorable and has m = |E| edges (thus, every assignment C : V → {1, 2, 3} must assign the same color to the endpoints of at least one edge). Give an upper bound on the probability that Raquel passes all N rounds.
c. Given 0 < ε < 1, determine how large N must be (in terms of m and ε) to guarantee that the probability of accepting a dishonest prover in part (b) is at most ε.

★ 15.5 Re-skin of 19.1: the cycle is tight ⚑ I65, I66

Source: Exam-style re-skin of HW5 #19.1 + tight-example part (Exam 2 pattern); counts verified by enumeration in calc.py

a. (5) Let λ be the min-cut size. Using only "one run of the contraction algorithm returns any *fixed* min cut with probability ≥ 2/(n(n−1))", prove that G has at most n(n−1)/2 distinct min cuts.
b. (3) Show the bound is tight: give a graph on n vertices with exactly n(n−1)/2 min cuts.

★ 15.6 Re-skin of 19.2: α-small cuts of a cycle ⚑ I66

Source: Exam-style re-skin of HW5 #19.2 (counts brute-force checked in calc.py)

Take $G = C_n$ ($\lambda = 2$) and $\alpha = 2$, so α-small cuts have at most 4 edges.

a. Count the α-small cuts exactly (in terms of n), and evaluate for n = 10.
b. Compare with the early-stop bound $C(n, 2\alpha)\cdot 2^{2\alpha-1}$ and with $n^{2\alpha}$. Is the $O(n^{2\alpha})$ growth rate tight here?

★ **15.7** **Re-skin of 19.2: stopping the watermelon early (guided)** ⚑ I66, I61

Source: Exam-style guided version of the HW5 #19.2 technique; telescoping identity checked in calc.py

Run the contraction algorithm only until r supervertices remain ($2 \le r \le n$). Fix a min cut Δ of size λ.

a. (4) Show that while Δ is intact and t supervertices remain, the next contraction avoids Δ with probability $\ge 1 - 2/t$.
b. (5) Conclude $\Pr[\Delta \text{ survives until r supervertices remain}] \ge r(r-1)/(n(n-1))$. Evaluate for $n = 10$, $r = 4$.
c. (4) Now let C be a cut of size $\le \alpha\lambda$ with 2α an integer k. Show $\Pr[C \text{ survives until k supervertices remain}] \ge 1/C(n, k)$.
d. (4) Finish: output a uniformly random cut of the k-vertex graph (there are $2^{k-1} - 1$). Conclude that there are $O(n^{2\alpha})$ α-small cuts.

(You may use earlier parts even if you didn't prove them.)

★ **15.8** **Re-skin of #20: MAX-Exact-4SAT** ⚑ I68, I65

Source: Exam-style re-skin of HW5 #20 with k = 4 (tight example verified by brute force in calc.py)

Every clause has exactly 4 literals on 4 distinct variables.

a. (4) Show a uniformly random assignment satisfies 15/16 of the clauses in expectation.
b. (5) Show: if some assignment satisfies strictly fewer than 15m/16 clauses, then some assignment satisfies strictly more.
c. (4) Construct a formula on which *every* assignment satisfies exactly 15/16 of the clauses.
d. (2) A formula has m = 20 such clauses. How many clauses can you guarantee to satisfy simultaneously?

★ 15.9 Re-skin of #20(c): exact-2 clauses ⚑ I65

Source: Exam-style tight-example drill (figure in §15.2)

(a) Give a MAX-Exact-2SAT formula where every assignment satisfies exactly 3/4 of the clauses. (b) Why does no formula with exactly-2 clauses force *fewer* than 3/4 for every assignment? (c) m = 5 exact-2 clauses: how many can always be satisfied together?

★ 15.10 Predicted re-skin of #21: randomized rounding for SET COVER (guided)

⚑ I64, I67, I61

Source: Predicted exam version: HW5 #21 with roles swapped, the same move Exam 2 Q4 made with HW4 #18

Universe $U = \{1, \ldots, n\}$; sets $S_1, \ldots, S_m \subseteq U$ with union U. Choose the fewest sets covering U.

a. (4) Write an ILP with a variable y_j per set, and its LP relaxation. Why is $LP^* \leq OPT$?
b. (5) Pick each set S_j independently with probability y^*_j. For an element e in sets $S_{j_1}, \ldots, S_{j_k}$, show $\Pr[e \text{ covered}] \geq 1 - 1/e$.
c. (4) Repeat the rounding t times independently and take the union. Show $\Pr[\text{some element uncovered}] \leq n \cdot e^{-t}$. Pick t so this is $\leq 1/4$.
d. (4) Show $E[\#\text{ sets chosen}] \leq t \cdot OPT$.
e. (3) Conclude: with probability $\geq 1/2$ the output is a cover of size $\leq 4t \cdot OPT = O(\log n) \cdot OPT$.

★ 15.11 Re-skin of #21(d)–(e): expected number of rounds and cost ⚑ I67

Source: Exam-style re-skin of HW5 #21(d)–(e) on set cover; uses MU Lemma 2.9 (tail sum)

Now repeat rounds *until* every element is covered; let T be the number of rounds.

a. Show $\Pr[T > t] \le n \cdot e^{-t}$, and conclude $E[T] \le \lceil \ln n \rceil + 1/(1 - 1/e) = O(\log n)$.
b. Show E[total # sets picked] ≤ E[T]·LP*. (Careful: T depends on the rounds.)

★ 15.12 Tight instance for one round of rounding ⚑ I65

Source: Exam-style "construct an instance" (cf. Exam 2 Q4.4)

Give a set-cover instance and an optimal LP solution where one round of randomized rounding covers some element with probability exactly $1 - (1 - 1/k)^k$ (close to $1 - 1/e$ for large k). Also: what does deterministic threshold rounding (pick S_j iff $y^*_j \ge 1/k$, Exam 2 Q4) pick on this instance?

★ 15.13 Re-skin of #22: a 2-coloring protocol ⚑ I63

Source: Exam-style re-skin of HW5 #22 with 2 colors (numbers in calc.py)

Raquel claims G is bipartite (2-colorable) and runs the same protocol with a uniformly random permutation of 2 colors each round. Brito challenges a uniformly random edge.

a. (4) What is the distribution of the two revealed colors in a round, for an honest prover?
b. (4) If G is *not* bipartite, with m edges, bound the probability that Raquel passes all N rounds.
c. (4) How many rounds guarantee cheating success $\leq \varepsilon$? Evaluate for $m = 15$, $\varepsilon = 10^{-3}$.
d. (3) Is the overall test Las Vegas or Monte Carlo? One-sided or two-sided error?

★ 15.14 What breaks the protocol? ⚑ I63

Source: Exam-style conceptual variant of HW5 #22

For the 3-coloring protocol, explain what goes wrong in each modification: (a) Raquel uses the same permutation π in every round. (b) Brito asks for the color of a single random vertex instead of an edge. (c) The commitment is not binding (Raquel can change colors after seeing the challenge).

16 · Select-ALL drill (Exam 2 format)

Notes §16

★ 16.1 Karger / watermelon ⚑ I60, I17

Source: Exam 2 Q1 format (5 points)

Consider one run of the contraction (watermelon) algorithm on a connected multigraph with n vertices.

Select ALL that apply.

- ○ It always returns a cut.
- ○ It returns every fixed minimum cut with probability at least $2/(n(n-1))$.
- ○ It is a Las Vegas algorithm.
- ○ Running it $n^2 \ln n$ times and keeping the smallest cut returns a min cut with probability $\geq 1 - 1/n$.

★ 16.2 Linearity and indicators ⚑ I60, I13, I62

Source: Exam 2 Q1 format (5 points)

Each vertex of G (m edges) independently joins A or B by a fair coin; X = number of crossing edges.

Select ALL that apply.

- ○ $E[X] = m/2$.
- ○ The edge indicators are mutually independent.
- ○ $\Pr[X \geq m/2] > 0$.
- ○ $\Pr[X = m/2] > 0$ for every graph.

★ 16.3 MAX-SAT guarantees ⚑ I60, I28, I31

Source: Exam 2 Q1 format (6 points)

Which statements are always true?

Select ALL that apply.

- ○ Fair coins satisfy at least half the clauses of any CNF formula in expectation.
- ○ Fair coins satisfy at least 7/8 of the clauses of any formula whose clauses have at most 3 literals, in expectation.
- ○ LP rounding satisfies at least $(1 - 1/e)\cdot\mathrm{OPT}$ clauses in every run.
- ○ The better of the fair-coin and LP-rounding assignments satisfies at least $\frac{3}{4}\cdot\mathrm{OPT}$ clauses in expectation.

★ 16.4 Reduction direction ⚑ I60, I33, I40

Source: Exam 2 Q1 format (5 points)

Suppose problem A reduces to problem B in polynomial time.

Select ALL that apply.

- ○ If A is NP-hard, then B is NP-hard.
- ○ If B is NP-hard, then A is NP-hard.
- ○ If $B \in P$, then $A \in P$.
- ○ If $A \in P$, then $B \in P$.

★ 16.5 Classes ⚑ I60, I38, I37

Source: Exam 2 Q1 format (6 points), course definitions

Using the course's definitions (NP = search problems with a poly-time verifier):

Select ALL that apply.

- ○ P ⊆ NP.
- ○ Every NP-hard problem is in NP.
- ○ 2-SAT is in P.
- ○ Max-SAT (find an assignment maximizing satisfied clauses) is NP-complete.

★ 16.6 What a valid reduction needs (select one) ⚑ I35, I36

Source: Exam 2 Q1(b) format (5 points, single answer)

Which condition is NOT required for f, g to form a reduction from A to B in this course?

Select ONE.

- ○ I has a solution ⇔ f(I) has a solution.
- ○ f and g run in polynomial time.
- ○ f is injective.
- ○ g maps a solution of f(I) to a solution of I.

★ 16.7 Independent set / vertex cover / clique ⚑ I60, I54

Source: Exam 2 Q1 format (6 points)

G has n vertices.

Select ALL that apply.

- ○ S is independent in G iff V\S is a vertex cover of G.
- ○ S is independent in G iff S is a clique in the complement graph $\hat{G}$.
- ○ Complementing a 2-approximate vertex cover always gives a 2-approximate independent set.
- ○ max |independent set| + min |vertex cover| = n.

★ 16.8 Repetition protocols ⚑ I60, I17

Source: Exam 2 Q1 format (6 points); HW5 #22 setting

A one-sided-error test catches a cheater with probability ≥ 1/m per round, independently over rounds.

Select ALL that apply.

- ○ After N rounds a cheater passes with probability ≤ $e^{-N/m}$.
- ○ N = m ln(1/ε) rounds make cheating success ≤ ε.
- ○ An honest prover can be rejected with small probability.
- ○ Reusing the same random permutation in every round keeps the protocol zero-knowledge.

Mock Exam A: 70 minutes, two pages of notes, no calculator

Notes §17 · exact Exam 2 layout: Q1 40 (a 5, b 5, c 6, d 10, e 6, f 8) · Q2 15 · Q3 20 · Q4 25

★ **M.1** **Mock A §1: short answer (40 points)** ⚑ I60, I62, I56, I10, I50

Source: Mock A, exact Exam 2 layout: (a) 5 select-all, (b) 5 select-one, (c) 6 select-all, (d) 10 calculation, (e) 6 short proof, (f) 8 bounds

(a) (5 points) One run of the watermelon (contraction) algorithm on a connected multigraph with n vertices. Select ALL statements that are always true.

- ○ It is a Las Vegas algorithm.
- ○ It returns a fixed min cut with probability at least $2/(n(n-1))$.
- ○ In the analysis, the probability that the k-th contraction avoids the min cut (given the past was good) is non-increasing in k (as a bound).
- ○ Keeping parallel edges is unnecessary for the analysis.

(b) (5 points) A is NP-hard, A reduces to B, and $B \in P$. What can we conclude?

- ○ Nothing: reductions say nothing about P.
- ○ P = NP.
- ○ B is NP-complete.
- ○ A is in NP.

(c) (6 points) Select ALL statements that are always true.

- ○ If $E[X] = \mu$, then $\Pr[X \geq \mu] > 0$.
- ○ Linearity of expectation requires independence.
- ○ Markov's inequality applies to every random variable with finite mean.
- ○ The derandomized (conditional expectations) max-cut algorithm outputs a cut with $\geq m/2$ edges.

(d) (10 points) $\varphi = (a \vee b \vee \neg c) \wedge (\neg a \vee c) \wedge (\neg b \vee \neg c)$. Build the 3-SAT → Independent Set graph G: give $|V|$, the number of clause edges and complement edges, and k. Exhibit an independent set of size k and the assignment it gives.

(e) (6 points) Prove Markov's inequality: $X \geq 0$, $a > 0 \Rightarrow \Pr[X \geq a] \leq E[X]/a$.

(f) (8 points) A MAX-SAT formula has m = 48 clauses. You compute that a uniformly random assignment satisfies 40 clauses in expectation, and the LP relaxation has optimal value 44. What are the strongest lower and upper bounds on OPT that follow? Justify briefly.

★ **M.2** **Mock A §2: counting cuts (15 points)** ⚑ I63, I66, I65

Source: Mock A: re-skin of HW5 #19 with a tight-example part (Exam 2 Q2 shape)

a. (5) Using the fact that one run of the contraction algorithm returns any fixed min cut with probability $\geq 2/(n(n-1))$, prove that a graph has at most $n(n-1)/2$ min cuts.

b. (8) Run the contraction algorithm until only r supervertices remain. Show a fixed min cut survives with probability $\geq r(r-1)/(n(n-1))$. *Hint: each supervertex has degree $\geq \lambda$.*

c. (2) Construct a family of graphs with exactly $n(n-1)/2$ min cuts.

★ **M.3** **Mock A §3: randomized rounding for set cover (20 points)** ⚑ I64, I67, I61

Source: Mock A: predicted re-skin of HW5 #21 (Exam 2 Q4 shape)

$U = \{1, \ldots, n\}$, sets $S_1, \ldots, S_m$ covering U. Pick the fewest sets covering U.

a. (4) Write an ILP and its LP relaxation.
b. (6) Round: pick each S_j independently with probability y^*_j. Prove that every element is covered with probability $\geq 1 - 1/e$. *Hint: AM–GM.*
c. (5) Repeat $t = \lceil \ln(4n) \rceil$ times and take the union. Prove that the result fails to be a cover with probability $\leq 1/4$.
d. (5) Prove $E[\#\text{ sets}] \leq t \cdot OPT$, and conclude an $O(\log n)$-approximation that succeeds with probability $\geq 1/2$.

★ **M.4** **Mock A §4: a reduction (25 points)** ⚑ I06, I47, I45, I64

Source: Mock A: HITTING SET is NP-complete (ties HW4 #18 / HW5 #21 to the Nov 12 reductions)

HITTING-SET (search): given subsets $S_1, \ldots, S_m \subseteq [n]$ and k, find $H \subseteq [n]$ with $|H| \leq k$ and $H \cap S_j \neq \emptyset$ for all j, or report none.

a. (4) Show HITTING-SET $\in$ NP.
b. (13) Reduce VERTEX COVER to HITTING-SET and prove the $\Leftrightarrow$.
c. (4) Argue polynomial time and explain how to recover a vertex cover from a hitting set.
d. (4) HW4 gave a b-approximation for hitting set, where $b = \max |S_j|$. What does it give for vertex cover through your reduction? Does this contradict §11.4 (reductions don't preserve approximation)?

Mock Exam B: 70 minutes, two pages of notes, no calculator

Notes §17 · same layout, different questions

★ **N.1** **Mock B §1: short answer (40 points)** ⚑ I60, I33, I43, I21, I62

Source: Mock B, exact Exam 2 layout

(a) (5 points) Select ALL statements that are always true.

- ○ Randomized Quicksort is a Las Vegas algorithm.
- ○ A Monte Carlo algorithm may return a wrong answer.
- ○ If a Monte Carlo algorithm succeeds with probability $\geq 1/n^3$, then $O(n^3 \log n)$ independent repetitions push failure to $\leq 1/n$.
- ○ If success probability is 2^{-n}, polynomially many repetitions give success $1 - o(1)$.

(b) (5 points) To prove a new problem B is NP-hard using a known NP-hard problem A, you must show:

- ○ B reduces to A.
- ○ A reduces to B.
- ○ Both directions.
- ○ $B \in NP$.

(c) (6 points) Select ALL statements that are always true.

- ○ Fair coins give a 7/8-approximation in expectation for MAX-Exact-3SAT.
- ○ For a maximization ILP, the LP relaxation's optimum is $\leq$ the ILP optimum.
- ○ In randomized rounding for MAX-SAT, a negated literal $\neg x_i$ is true with probability $1 - \hat{y}_i$.
- ○ Integer linear programming is NP-hard.

(d) (10 points) $f = (x1 \vee x2) \wedge (\neg x1 \vee x3) \wedge (\neg x3 \vee \neg x2) \wedge (x2 \vee x3) \wedge (\neg x2 \vee x1)$. Draw the implication graph (list the edges), find its SCCs, decide satisfiability, and give a satisfying assignment if one exists.

(e) (6 points) Prove: if X is a discrete random variable with $E[X] = \mu$, then $Pr[X \geq \mu] > 0$.

(f) (8 points) The derandomized max-cut algorithm returns a cut of size 18 on a graph with $m = 30$ edges. What are the strongest lower and upper bounds on the maximum cut that follow? Justify briefly.

★ **N.2** **Mock B §2: MAX-Exact-4SAT (15 points)** ⚑ I68, I65

Source: Mock B: re-skin of HW5 #20 (k = 4)

a. (4) Show a random assignment satisfies 15/16 of the clauses of a MAX-Exact-4SAT formula in expectation.
b. (7) Show that if some assignment satisfies fewer than 15/16 of the clauses, another satisfies more.
c. (4) Construct a formula where every assignment satisfies exactly 15/16 of the clauses.

★ **N.3** **Mock B §3: a challenge protocol (20 points)** ⚑ I63, I17

Source: Mock B: re-skin of HW5 #22 + boosting

A prover claims G (m edges) is bipartite and runs N rounds of the random-recoloring protocol with 2 colors; the verifier checks one uniformly random edge per round.

a. (5) Distribution of the revealed colors for an honest prover, and why that reveals nothing.
b. (6) Bound the cheating probability after N rounds when G is not bipartite.
c. (5) Rounds needed for cheating probability $\leq 10^{-6}$ when m = 40.
d. (4) Classify the test: Las Vegas or Monte Carlo? One-sided or two-sided error?

★ **N.4** **Mock B §4: Set Cover is NP-hard, then approximate it (25 points)**

⚑ I47, I45, I64

Source: Mock B: VC → SET COVER (KT 8.6) + link to randomized rounding

SET-COVER (search): universe U, subsets S_1, …, S_m, integer k. Find ≤ k sets whose union is U, or report none.

a. (4) Show SET-COVER ∈ NP.
b. (12) Reduce VERTEX COVER to SET-COVER and prove the ⇔.
c. (4) Polynomial time and recovery.
d. (5) Given that SET-COVER is NP-hard, why is the O(log n)-approximation of problem 15.10 still useful? What would an exact poly-time algorithm imply?

Solutions

In problem order. Cover the page and check one at a time.

3 · Probability toolkit, Las Vegas vs Monte Carlo

3.1 · Las Vegas or Monte Carlo?

a. **Las Vegas.** Every pivot sequence sorts correctly; only the number of comparisons is random (average $O(n \log n)$, worst n^2).
b. **Monte Carlo** with one-sided error: a prime is always classified correctly, but a composite can pass (remainder 1) and be wrongly called prime.
c. **Monte Carlo.** It always returns *a* cut in $n - 2$ contractions, but contracting a min-cut edge early destroys that cut. Success probability $\geq 2/(n(n-1))$.
d. **Las Vegas** (MU §6.2.1): the output is checked (count crossing edges in poly time), so it is always correct; the number of rounds is geometric with mean $1/p \leq m/2 + 1$.
e. **Monte Carlo**, and in fact only an expectation guarantee: $E[\#\text{satisfied}] \geq m/2$, but one run can be worse.

3.2 · Bernoulli, binomial, geometric by hand

a. $X \sim \text{Bin}(4, 1/4)$: $\Pr[X = 2] = C(4,2)(1/4)^2(3/4)^2 = \mathbf{27/128}$. $\Pr[X = 0] = (3/4)^4 = \mathbf{81/256}$. $E[X] = np = \mathbf{1}$ (linearity over 4 indicators, each with expectation 1/4).
b. $Y \sim \text{Geo}(1/4)$: $\Pr[Y = 3] = (3/4)^2(1/4) = \mathbf{9/64}$. $\Pr[Y > 3]$ = all three first trials fail $= (3/4)^3 = \mathbf{27/64}$. $E[Y] = 1/p = \mathbf{4}$ (MU Lemma 2.9 / memoryless argument).
c. Y, the geometric one: “repeat until success”. This is exactly why $\Pr[\text{success within N runs}] = \Pr[Y \leq N] = 1 - (1 - p)^N$.

3.3 · Expected number of heads, and the binomial mean

Write $X = X_1 + \cdots + X_n$ where $X_i = 1$ if toss i is heads, else 0. Each X_i is Bernoulli, so $E[X_i] = 1 \cdot \Pr[X_i = 1] + 0 = 1/2$. By linearity of expectation (no independence needed), $E[X] = \Sigma\, E[X_i] = \mathbf{n/2}$.

For Bin(n, p) the same decomposition gives $E[X_i] = p$, so $E[X] = \mathbf{np}$. Compare MU §2.2, where the direct computation from the pmf takes a page.

3.4 · What Markov does and doesn't tell you

Markov: $P(X \geq 24) \leq E[X]/24 = 6/24 = \mathbf{1/4}$.

For $a = 4$ the bound is $6/4 = 3/2 > 1$, which says nothing (every probability is ≤ 1). Markov is only useful for thresholds well above the mean. It can never show that the variable is unlikely to be *near or below* its mean.

3.5 · Prove Markov's inequality

Let I be the indicator of the event $\{X \geq a\}$. Then $I \leq X/a$ pointwise: if $X \geq a$ then $I = 1 \leq X/a$; otherwise $I = 0 \leq X/a$ because $X \geq 0$. Taking expectations (expectation preserves $\leq$), $\Pr[X \geq a] = E[I] \leq E[X/a] = E[X]/a$. ∎

Where $X \geq 0$ was used: the case $I = 0$. Without it, X/a could be negative and the pointwise inequality fails.

3.6 · Quicksort and Markov

a. Markov: $\leq (n \log n)/(10\, n \log n) =$ **1/10**.
b. $\leq (2\, n \ln n)/(10\, n \ln n) =$ **1/5**.
c. No. The worst case is still n^2. Markov only says the probability of a run 10× slower than average is at most 1/10. That's the trade-off the professor stressed: “you can say something meaningful but not exact.”

3.7 · True/False: probability tools

a. **False.** Linearity holds for any random variables, even dependent ones (Lec 16/17: “independence is not required”). The max-cut edge indicators are dependent, and linearity still applies.
b. **False.** It needs $X \geq 0$. Counterexample: $X = -1$ or $+1$ with probability ½ each, $a = 1$: $\Pr[X \geq 1] = ½$, but $E[X]/a = 0$.
c. **True.** $E[I] = 1 \cdot \Pr[I=1] + 0 \cdot \Pr[I=0]$. This is the step behind every “$E[X_e] = \Pr[e \text{ crosses}]$”.
d. **False.** Lec 16 typed notes: “The guarantees are always taken over the algorithm's random choices, not over a presumed random input distribution.” The input is fixed and worst case.
e. **True.** Definition on the Lec 16 typed notes (Karger is the example).
f. **True.** MU Lemma 2.9: $E[X] = \Sigma_i \Pr[X \geq i] = \Sigma_i (1-p)^{i-1} = 1/p$.

3.8 · Jensen's inequality (textbook only)

$E[X] = 50$, so $(E[X])^2 = 2500$. $E[X^2] = (1/99)\Sigma i^2 = 9950/3 \approx 3316.67 > 2500$. In general $E[X^2] - (E[X])^2 = E[(X - E[X])^2] \geq 0$. **Jensen:** for convex f, $E[f(X)] \geq f(E[X])$. ($f(x) = x^2$ is convex.) The trap: $E[X^2] \neq (E[X])^2$, whereas linearity only covers sums and constant multiples.

3.9 · Memoryless + coupon collector (textbook only)

a. $\Pr[X = n + k \mid X > k] = (1-p)^{n+k-1}p \,/\, (1-p)^k = (1-p)^{n-1}p = \Pr[X = n]$. Past failures don't change the future.
b. Let X_i be the boxes bought while holding exactly $i - 1$ types. A new type appears with probability $p_i = (n - i + 1)/n$, so $X_i \sim \text{Geo}(p_i)$ and $E[X_i] = n/(n - i + 1)$. Linearity: $E[X] =$

$\Sigma\ n/(n-i+1) = n \cdot H(n) = n \ln n + \Theta(n)$. $n = 5$: $137/12 \approx 11.42$. $n = 10$: ≈ 29.29 (compare $n \ln n = 23.03$).

4 · Karger's min-cut (watermelon) and boosting

4.1 · Karger on the worked-example graph

a. $(m - |\Delta|)/m = 6/8 = \mathbf{3/4}$. The proof uses $|\Delta|/|E| \le 2/|V|$, giving $\ge 1 - 2/6 = 2/3$. The true value is larger, as it should be (the bound is worst case).
b. $\deg(1)=2$, $\deg(2)=3$, $\deg(3)=3$, $\deg(4)=3$, $\deg(5)=3$, $\deg(6)=2$, all $\ge 2 = |\Delta|$. (Each vertex alone is a cut, so its degree is at least the min-cut size.)
c. $2/(n(n-1)) = 2/30 = \mathbf{1/15} \approx 0.0667$. The exact $6/35 \approx 0.1714$ is above it. Consistent.
d. This graph has 3 min cuts ($\{1\}|$rest, $\{6\}|$rest, $\{1,2,3\}|\{4,5,6\}$). Returning any of them counts as success, so the probability adds over the disjoint events. The theorem is about one fixed Δ.

4.2 · Reproduce the Karger bound (full proof)

Preservation. If the algorithm never contracts an edge of Δ, the final two supervertices are exactly the two sides of Δ, so it returns Δ. (Contracting an edge not in Δ keeps Δ a min cut of the contracted graph: every edge of Δ survives and contraction creates no smaller cut.)

Step probabilities. Let $e_1, \ldots, e_{n-2}$ be the contracted edges. Before step k (given $e_1, \ldots, e_{k-1} \notin \Delta$) the graph has $n - k + 1$ vertices, all of Δ, and E_k edges, so $\Pr[e_k \notin \Delta \mid \text{past good}] = 1 - |\Delta|/E_k$.

Degree bound. In the current graph each vertex v alone is a cut of size $\deg(v)$, so $\deg(v) \ge |\Delta|$. Summing, $2E_k = \Sigma \deg(v) \ge |\Delta|(n - k + 1)$, i.e. $|\Delta|/E_k \le 2/(n - k + 1)$.

Chain rule and telescoping. $\Pr[\text{success}] = \prod_{k=1}^{n-2} \Pr[e_k \notin \Delta \mid \text{past good}] \ge \prod (1 - 2/(n-k+1)) = \prod (n-k-1)/(n-k+1) = (n-2)/n \cdot (n-3)/(n-1) \cdot (n-4)/(n-2) \cdots 2/4 \cdot 1/3 = 2 \cdot 1/(n(n-1))$. ∎

4.3 · Why keep parallel edges?

(1) **The output must be a real cut of G.** The edges between the final two supervertices are exactly the original edges crossing that partition. Merging parallel copies would undercount the cut's size, and we'd compare wrong sizes across runs.

(2) **The probability analysis counts them.** "Uniformly at random" is over current edges with multiplicity, and the degree bound $2|E| = \Sigma \deg(v) \ge |\Delta| \cdot |V|$ needs $\deg(v)$ to count every surviving original edge. With parallel copies merged, a min-cut edge could be chosen with higher probability than the bound allows.

4.4 · How many runs for n = 10?

a. $p = 2/90 = \mathbf{1/45}$.
b. $1 - (1 - p)^N$: $N = 45$ gives $\geq \mathbf{0.6362}$ (about $1 - 1/e$, since $N = 1/p$); $N = 100$ gives $\geq \mathbf{0.8943}$.
c. Need $e^{-pN} \leq 0.01$, i.e. $N \geq \ln(100)/p = 45 \cdot \ln 100$, so $N = \mathbf{208}$ suffices. (The smallest N with $(1 - p)^N \leq 0.01$ is 205; either answer is fine if justified.) By hand: $\ln 100 \approx 4.6$, and $45 \times 4.6 \approx 207$.

4.5 · n = 1000: the lecture's numbers

a. $(1 - p)^N \leq e^{-pN} = e^{-(2/(n(n-1)))\cdot\alpha\, n(n-1)\ln n} = e^{-2\alpha \ln n} = n^{-2\alpha}$. For $\alpha = 1$ that is 1.000×10^{-6}: success $\geq 1 -$ one in a million.
b. Same computation with half as many runs: $pN = \alpha \ln n$, so the failure bound is $n^{-\alpha}$. The constant in N just moves into the exponent.
c. About $n(n-1) \ln n = 6{,}900{,}848$ runs (board version, $\alpha = 1$), each $O(n^2)$-ish. That's $O(n^2 \log n)$ runs: polynomial.

4.6 · Slide exercise: from $1/n^2$ to failure $1/n^3$

Failure after N trials $\leq (1 - 1/n^2)^N \leq e^{-N/n^2}$. We need $e^{-N/n^2} \leq n^{-3} \Leftrightarrow N/n^2 \geq 3 \ln n \Leftrightarrow \mathbf{N \geq 3n^2 \ln n}$. So $O(n^2 \log n)$ repetitions. (E.g. $n = 10$: 691 runs; $n = 100$: 138,156.) This matches the typed notes' general rule $O(p^{-1} \log(1/\delta))$ with $p = n^{-2}$, $\delta = n^{-3}$.

4.7 · “Run until it outputs a min cut”?

We can't recognize success: given a cut, we can't check that it's *minimum* without already knowing the min-cut size, which is the thing we're computing. Instead, run N independent times and **keep the smallest cut found**. If any run produces a min cut, the kept one is a min cut, so $\Pr[\text{success}] \geq 1 - (1 - p)^N$. What makes it work: the *size* of a cut is easy to compute, so we can compare candidates by objective value even though we can't certify optimality (Lec 17 typed notes: “We need not know which trial succeeded if we can select the best candidate by its objective value”).

4.8 · True/False: Karger and boosting

a. **False.** Monte Carlo: it may return a non-minimum cut. Its running time is fixed ($n - 2$ contractions).
b. **True.** $p^{-1} = n^5$ is polynomial; $N = n^5 \cdot c \cdot \ln n$ runs give failure $\leq n^{-c}$ (the boosting principle).
c. **False.** You'd need about 2^n runs to make $(1 - 2^{-n})^N$ small. That isn't polynomial.

d. **False.** Failure probability ≤ $1/n^2$, not 0: it's "with high probability", not "always".
e. **True.** The bound $1 - 2/(n-k+1)$ shrinks as fewer vertices remain. The professor: "at the end you need to get really lucky."
f. **True.** The lecture's slow method: run max-flow for every pair (s, t) and keep the smallest min s–t cut, since the global min cut separates some pair.

4.9 · The cycle shows the bound is tight

a. $C(5,2) = 10$ min cuts. Full enumeration gives each probability exactly **1/10** $= 2/(5\cdot4) = 1/10$, so the theorem is tight here. The 10 probabilities sum to 1.
b. For distinct min cuts $\Delta_1, \ldots, \Delta_t$, the events "the run returns Δ_i" are disjoint, and each has probability $\geq 2/(n(n-1))$. Their total is ≤ 1, so $t \leq n(n-1)/2$.

5 · Probabilistic method and derandomization

5.1 · If E[X] = μ then both sides have positive probability

Suppose $\Pr[X \geq \mu] = 0$. Then every value with positive probability is $< \mu$, so $\mu = E[X] = \Sigma_{x<\mu}\, x\cdot\Pr[X = x] < \Sigma_{x<\mu}\, \mu\cdot\Pr[X = x] = \mu$, a contradiction. (The strict inequality holds because the probabilities sum to 1, so some term has positive probability.) The case $\Pr[X \leq \mu] = 0$ is symmetric: all mass is $> \mu$, giving $\mu > \mu$. ∎

Use: "some outcome achieves at least the average", so an object with value ≥ μ exists.

5.2 · Expected size of a random cut

a. Edge uv: condition on u's side; v independently lands on the other side with probability ½. (Equivalently, 2 of the 4 equally likely outcomes split u and v.)
b. Let X_e be the indicator of e crossing, so $|C(A,B)| = \Sigma_e X_e$. By linearity (the X_e are *not* independent, and that doesn't matter), $E = \Sigma_e \Pr[X_e = 1] = m/2$.
c. By the probabilistic method (5.1), $\Pr[|C| \geq m/2] > 0$, so such a cut exists. Since $OPT \leq m$, a random cut has expected size $m/2 \geq OPT/2$.

5.3 · From expectation to probability (max-cut)

a. Split E[X] at m/2: $m/2 = \Sigma_{i<m/2}\, i\,\Pr[X=i] + \Sigma_{i\geq m/2}\, i\,\Pr[X=i] \leq (m/2 - 1)(1 - p) + m\cdot p$, since the small values are $\leq m/2 - 1$ and the large ones $\leq m$. Rearranging: $m/2 \leq m/2 - 1 + p(m/2 + 1)$, so **$p \geq 1/(m/2 + 1)$**.
b. For odd m the largest integer below m/2 is $(m-1)/2$, giving $m/2 \leq (m-1)/2\cdot(1 - p) + mp \Rightarrow p \geq (1/2)/((m+1)/2) =$ **$1/(m + 1)$**. $m = 8$: $p \geq 1/5$. $m = 9$: $p \geq 1/10$. Both are inverse-polynomial, which is all that boosting needs.

c. Repeat: sample a random cut, count crossing edges, stop when ≥ m/2. The output is always correct, so it's **Las Vegas**. The number of samples is geometric with mean 1/p ≤ m/2 + 1 (m = 8: ≤ 5; m = 9: ≤ 10). Alternatively, run a fixed number N of samples and keep the best (Monte Carlo, failure ≤ $(1 - p)^N$).

5.4 · Derandomize max-cut by hand

vertex	placed nbrs in A / B	E[cut | → A]	E[cut | → B]	choose
v1	0 / 0	9/2	9/2	A
v2	1 / 0	4	5	B
v3	1 / 1	5	5	A
v4	0 / 1	11/2	9/2	A
v5	2 / 0	9/2	13/2	B
v6	2 / 1	6	7	B

Final cut size **7** ≥ m/2 = 9/2 (here it is even optimal: brute force gives max cut 7). Shortcut: E[cut | ·] = (crossing edges among placed vertices) + ½·(all other edges), so you only compare the new vertex's placed neighbors: put it opposite the majority.

5.5 · Slide exercise: the majority rule directly

Charge each edge to its *later* endpoint in the processing order, so every edge is charged exactly once. When vertex v is placed, let d_v be the number of its already-placed neighbors. Putting v on the side opposite the majority of them makes at least $\lceil d_v/2 \rceil \geq d_v/2$ of these edges cross, and later placements never change whether these edges cross. Summing over v: crossing edges ≥ $\Sigma\, d_v/2 = m/2$, because $\Sigma_v\, d_v = m$. ∎

5.6 · Slide exercise: the indicators are dependent

In a triangle, if two edges cross then the third cannot: two vertices are on one side and the third on the other, so exactly 0 or 2 edges cross. So Pr[all three cross] = 0 ≠ $(½)^3$. The exact distribution is Pr[X=0] = 1/4, Pr[X=2] = 3/4, with E[X] = 3/2 = 3·½, as linearity predicts. Linearity adds expectations of any random variables; independence would be needed only for products.

5.7 · MAX-SAT existence (probabilistic method)

Random assignment: a clause with $k_j \geq k$ distinct literals fails only if all its literals are false, with probability $2^{-k_j} \leq 2^{-k}$. By linearity, E[#satisfied] $\geq m(1 - 2^{-k})$, and by the expectation argument some assignment achieves at least that. For m = 40, k = 3: $\geq$ **35** clauses.

5.8 · Derandomize MAX-SAT by hand

Start: $E = \Sigma\ (1 - 2^{-k_j}) =$ **29/8**.

variable	E[· | T]	E[· | F]	choose
x1	4	13/4	T
x2	7/2	9/2	F
x3	5	4	T

Final assignment satisfies **5** of 5 clauses ($\geq$ 29/8; brute-force optimum 5). Each conditional expectation is computed clause by clause: a satisfied clause counts 1, a falsified one 0, and a clause with f free literals and no true literal counts $1 - 2^{-f}$.

5.9 · True/False: probabilistic method

a. **False.** Existence only. Finding needs an extra step: the probability bound plus boosting (Las Vegas), or derandomization.
b. **False.** Deterministic for a fixed order, but different orders can give different cuts, all with $\geq$ m/2 edges (Nov 3 recap).
c. **False.** Max-Cut is NP-hard (MU §6.2.1; the professor: “finding the largest cut can take us a long time”). Min-cut is in P.
d. **True.** By symmetry: swapping A and B doesn't change the cut size.
e. **True.** The number of samples is geometric with mean $1/p \leq m/2 + 1$ (MU §6.2.1).
f. **True.** It finds $\geq m/2 \geq$ OPT/2 edges, deterministically.

5.10 · Union bound: 2-coloring K_{1000} (textbook only)

A fixed k-clique is monochromatic with probability $2 \cdot 2^{-C(k,2)}$ (all edges red or all blue). There are C(n,k) cliques, so by the union bound Pr[some monochromatic K_k] $\leq C(n,k) \cdot 2^{-C(k,2)+1}$. If this is < 1, the complement event has positive probability, so a good coloring exists. For n = 1000, k = 20: $C(1000,20) \cdot 2^{-189} = 4.327 \times 10^{-16} < 1$. ✓

5.11 · Sample and modify: independent sets (textbook only)

d = 2m/n = 6. E[surviving vertices] = n/d and E[surviving edges] = $(nd/2)(1/d)^2 = n/(2d)$. Deleting one endpoint per edge leaves $\geq X - Y$ vertices, so E[IS size] $\geq n/d - n/(2d) = n/(2d)$

= $n^2/(4m)$. Here $n^2/(4m) = 25/3 \approx 8.33$. Since a set size is an integer, some independent set has ≥ **9** vertices.

6 · Randomized rounding: MAX-SAT

6.1 · Check the ILP clause constraint

Constraint: $z \le y_1 + (1 - y_2) + y_3$.

Satisfying, e.g. x = (T, T, F): RHS = 1 + 0 + 0 = 1, so z can be 1 and the maximization sets it to 1. The clause is counted.

Unsatisfying, the unique one x = (F, T, F): RHS = 0 + 0 + 0 = 0, so z = 0. The clause is not counted.

6.2 · Write the ILP and the LP relaxation

a. max $z_1 + z_2 + z_3$ s.t. $z_1 \le y_1 + (1 - y_2)$, $z_2 \le y_2$, $z_3 \le (1 - y_1) + (1 - y_2)$, with $y, z \in \{0,1\}$ (ILP), or $0 \le y, z \le 1$ (LP).
b. OPT = **2**: no assignment satisfies all three. LP* = **2.5**.
c. Every integer solution is LP-feasible and the LP maximizes over a larger set (relaxation).
d. Fair coins: ¾ + ½ + ¾ = **2**. LP rounding with $\hat{y} = (½, ½)$: also **2.0000**. Both are ≥ $(1 - 1/e)$·LP* and ≥ ½·OPT, as guaranteed.

6.3 · One clause, fractional values

a. Fails only if all three are false: 0.5·0.7·0.8 = 0.2800, so Pr = **0.7200**.
b. $1 - (1 - 1/3)^3 = 0.7037 = \beta_3 \cdot 1$ (here $\Sigma\hat{y} = \hat{z} = 1$, so the AM–GM bound equals $\beta_3\hat{z}$). $(1 - 1/e)\cdot 1 = 0.6321$. Order: 0.7200 ≥ 0.7037 ≥ 0.6321.
c. $\hat{z} \le (1 - 0.9) + (1 - 0.6) = 0.5000$. Literal probabilities: Pr[¬x_1 true] = 0.1, Pr[¬x_2 true] = 0.4. Pr[sat] = $1 - 0.9\cdot 0.6$ = **0.4600** ≥ $\beta_2\hat{z} = ¾\cdot 0.5 = 0.3750$. ✓ (For negated literals, use $1 - \hat{y}_i$ as the literal's probability.)

6.4 · Slide exercise: concavity without derivatives

$g(z) = 1 - e^{-z}$ is concave (e^{-z} is convex, and the negative of a convex function is concave). Any $z \in [0,1]$ is the convex combination $z = (1 - z)\cdot 0 + z\cdot 1$, so concavity gives $g(z) \ge (1 - z)g(0) + z\cdot g(1) = (1 - z)\cdot 0 + z(1 - 1/e) = (1 - 1/e)z$. ∎ (The curve lies above its chord from (0, 0) to $(1, 1 - 1/e)$.)

6.5 · Slide exercise: why 7/8 needs exactly 3 literals

The proof uses Pr[clause fails] = $(\frac{1}{2})^3$ for *every* clause. A unit clause fails with probability ½ and a 2-clause with ¼, so the expected fraction can drop to ½. E.g. the formula $(x) \wedge (\neg x)$: fair coins satisfy exactly 1 of 2 clauses. Only the general bound $E \geq m/2$ survives. (Also needed: the three literals are distinct variables, so the events are independent.)

6.6 · Fill the comparison table

k	α_k	β_k	max	$\alpha_k + \beta_k$
1	1/2 = 0.5000	1 = 1.0000	1.0000	1.5000
2	3/4 = 0.7500	3/4 = 0.7500	0.7500	1.5000
3	7/8 = 0.8750	19/27 = 0.7037	0.8750	1.5787
4	15/16 = 0.9375	175/256 = 0.6836	0.9375	1.6211

The better factor is always ≥ ¾ and the sum is always ≥ 3/2 (minimum 1.5000, at k = 1 and k = 2). Fair coins win for k ≥ 3, the LP wins for k = 1, and they tie at k = 2.

6.7 · The ¾-approximation, proved correctly

$\max(M_1, M_2) \geq (M_1 + M_2)/2$ pointwise, so $E[\max] \geq \frac{1}{2}(E[M_1] + E[M_2])$. For a clause C_j of size k: $\Pr_{fair}[C_j \text{ sat}] = \alpha_k \geq \alpha_k \hat{z}_j$ (since $\hat{z}_j \leq 1$) and $\Pr_{LP}[C_j \text{ sat}] \geq \beta_k \hat{z}_j$. Summing over clauses with linearity: $\frac{1}{2}(E[M_1] + E[M_2]) \geq \sum_j \frac{1}{2}(\alpha_k + \beta_k)\hat{z}_j \geq \frac{3}{4} \sum_j \hat{z}_j = \frac{3}{4} \cdot LP^* \geq \frac{3}{4} \cdot OPT$. ∎

Why not the video's version: “for each clause take whichever algorithm is better” describes a different, imaginary per-clause choice. The algorithm picks one whole assignment, so you must average.

6.8 · Prove the rounding lemma

Let C_j have k literals with satisfaction probabilities $p_1, \ldots, p_k$ ($p_r = \hat{y}_i$ for x_i, $1 - \hat{y}_i$ for $\neg x_i$). The LP constraint says $\Sigma p_r \geq \hat{z}_j$. The variables are distinct and independent, so $\Pr[\text{sat}] = 1 - \prod(1 - p_r)$. AM–GM on the k numbers $1 - p_r$: $\prod(1 - p_r) \leq ((1/k)\Sigma(1 - p_r))^k = (1 - \Sigma p_r/k)^k \leq (1 - \hat{z}_j/k)^k$. So $\Pr[\text{sat}] \geq 1 - (1 - \hat{z}_j/k)^k =: f(\hat{z}_j)$. f is concave on [0,1] with f(0) = 0, so $f(z) \geq f(1) \cdot z = \beta_k z$. Finally $(1 - 1/k)^k \leq e^{-1}$ (from $1 - x \leq e^{-x}$), so $\beta_k \geq 1 - 1/e$.

Linearity: $E[\#\text{sat}] = \sum_j \Pr[C_j \text{ sat}] \geq (1 - 1/e)\Sigma_j \hat{z}_j = (1 - 1/e)LP^* \geq (1 - 1/e)OPT$. ∎

6.9 · Rounding on formula R

a. The LP is a relaxation, so its value can exceed the best integer value (5.66667 > 5).
b. Fair: 37/8 ≥ m/2 = 3.5. LP rounding: 4.8148 ≥ $(1 - 1/e) \cdot LP^*$ = 3.5820 ≥ $(1 - 1/e) \cdot OPT$.
c. E[max] ≥ average = 4.7199 ≥ ¾·LP* = 4.2500.

6.10 · Why not round deterministically?

Take the clause $(x_1 \vee \cdots \vee x_k)$, $k \geq 3$, with $\hat{y}_i = 1/k$ for all i. Then $\Sigma\hat{y}_i = 1$, so $\hat{z} = 1$ is feasible. Every $\hat{y}_i < ½$, so threshold rounding sets all of them false and the clause fails. Randomized rounding fails only with probability $(1 - 1/k)^k \leq 1/e$. The coins exploit the fractional mass spread across the literals; the threshold throws it away.

6.11 · True/False: rounding and MAX-SAT

a. **False.** The guarantee is in expectation; a single run can be lower (Lec 18 typed notes).
b. **False.** It's at least: the relaxation has more feasible points (LP* ≥ OPT).
c. **False.** Only the y_i are rounded. The resulting assignment determines which clauses are satisfied.
d. **True.** Håstad: doing better than 7/8 is NP-hard, and 0.88 > 0.875.
e. **False.** 7/8 needs exactly 3 distinct literals per clause; unit clauses are satisfied with probability only ½.
f. **True.** LP4: “Linear programming is solvable in polynomial time… integer linear programming is NP-hard”, shown via Max-SAT → ILP.

6.12 · Strongest bounds on OPT (Q1(f) drill)

a. **8 ≤ OPT ≤ 19.** Min problem: the LP relaxation gives OPT ≥ 7.4, and OPT is an integer, so OPT ≥ 8. The cover found is feasible, so OPT ≤ 19.
b. **27 ≤ OPT ≤ 30.** Exactly-3 clauses: E[fair coins] = 7·30/8 = 26.25, so some assignment satisfies ≥ 27 (integer). The run's 24 is a weaker lower bound. Upper: OPT ≤ m = 30. The expected guarantee says nothing about this single run, so “24 ≥ (1 − 1/e)OPT” is **not** valid.
c. **7 ≤ OPT ≤ 14.** Upper: the returned cover. Lower: 14 ≤ 2·OPT. The independent set gives only OPT ≤ 20 − 5 = 15 (its complement is a cover, §11.3), which is weaker than 14.

9 · P, NP, NP-hard, reductions

9.1 · Definitions, in the course's words

- **Search problem:** given an input I and a candidate solution S, a verifier checks in polynomial time whether S is a solution for I.
- **NP:** the class of all search problems. **P:** the search problems that can be *solved* in polynomial time (P ⊆ NP).
- **NP-hard:** every problem in NP reduces to it (f and g poly time). **NP-complete:** NP ∩ NP-hard.

No. NP-hard problems “don't have to be search problems”. Max-SAT or ILP as optimization problems are NP-hard but not in NP under this definition, because you can't verify optimality

in poly time.

9.2 · In NP or not (course definition)?

a. In NP (checking a path's length against a shortest-path computation is poly) and **in P**.
b. In NP: count the vertices and check no edge inside. Poly-time solvability is unknown; it's NP-complete.
c. **Not known to be in NP** (the lecture: “not a search problem” as seen in class): verifying optimality would need the DP, which is pseudo-polynomial (time depends on B, exponential in its bit length). Also not known to be in P. Knapsack-*search* (value ≥ g) is in NP (problem 9.10).
d. Not in NP as an optimization problem (you can't check that the count is maximum), but NP-hard (SAT → Max-SAT).
e. In NP and **in P** (SCCs of the implication graph).

9.3 · Direction of the reduction

(b) A → B. Then a poly-time algorithm for B would solve A, and hence everything in NP. B → A would only say B is no harder than A.

Shortest path → SAT holds (shortest path ∈ NP, Cook–Levin). SAT → shortest path is not known: it would put SAT in P and give P = NP. Reductions are not symmetric.

9.4 · What follows from A → B?

a. B is NP-hard.
b. A ∈ P (solve A by f, then B's algorithm, then g).
c. Nothing about B. Easy problems reduce to hard ones all the time.
d. Nothing about A. Everything in NP reduces to an NP-hard B, easy problems included.
e. B is NP-hard (from A) and in NP, so **B is NP-complete**.

9.5 · The dramatic case

B ∈ P ⇒ A ∈ P (reductions carry easiness backward). A is NP-hard, so every NP problem reduces to A and is in P. **P = NP.** (So we don't expect such a reduction to exist, but that's what it would imply.)

9.6 · Why is this “reduction” invalid?

Two requirements fail. (1) **The ⇔ fails**: f(I) always has a solution, even when I has none. (2) **No useful g**: the answer (the edge 1–2) carries no information about I, so no poly-time g can

turn it into a solution of I. f and g must be designed together so that a solution of f(I) can be translated back.

9.7 · True/False: P, NP, reductions

a. **False.** No restriction beyond: I solvable ⇔ f(I) solvable, g recovers a solution, f and g poly time (Nov 10 Q&A).
b. **False.** Unknown. We don't know how to; doing so would prove P = NP (Nov 10 Q&A: “We don't know yet”).
c. **True.** A problem we can solve in poly time can be verified by solving it (KT 8.10).
d. **False.** NP-hard problems “don't have to be search problems” (e.g. optimization Max-SAT, ILP).
e. **False.** “They very much depend on A and B.”
f. **True.** Everything in NP reduces to it (KT 8.12; Nov 10 lecture).
g. **True.** Compose: $f_{BC} \circ f_{AB}$ and $g_{AB} \circ g_{BC}$ are poly time (KT 8.9). Polynomials compose to polynomials.

9.8 · Why polynomial time composes

f outputs at most $O(n^2)$ symbols (it can't write more than its running time), so B's algorithm runs on $N = O(n^2)$: $O((n^2)^3) = O(n^6)$, plus $O(n^2)$ for f. Polynomial. Polynomials are closed under composition and sums, which is the professor's “sequence and nest efficient steps and stay efficient” idea. That's what makes “solve A via B” efficient whenever B is.

9.9 · Decision vs. search (self-reducibility)

a. Binary search on $k \in [0, n]$ (KT §8.1).
b. Let α be the max size. For each vertex v in turn: if $G - v$ still has an independent set of size α, delete v permanently (some optimum avoids it); otherwise keep it. At the end every remaining vertex is needed, so the remaining graph is an independent set of size α. That's n more calls. (Derived argument; the lecture left this as an exercise.)

9.10 · Knapsack-search and binary search

a. Given S, check $\Sigma_{i \in S} w_i \le B$ and $\Sigma_{i \in S} v_i \ge g$: two sums of at most n numbers, $O(n \log W)$ bit operations, polynomial in the input size. It has the search form (a solution or NO), so it's in NP.
b. Binary search on $g \in [0, V]$: if the box finds a subset for goal g, the optimum is $\ge g$; if it says NO, the optimum is $< g$. That takes $\lceil \log_2(V + 1) \rceil$ calls. The box's answer at the largest feasible g is an optimal subset. V is written in $O(n \log V_max)$ bits, so log V is at most the input size, and the total time is polynomial.

c. “≤ g” would be trivial: the empty set always has value 0 ≤ g. We want to push the value *up*.
d. V = 20: binary search makes 5 calls (≤ $\lceil \log_2 21 \rceil = 5$), and the optimum is 13, using items {2, 4} (weights 4 + 5 = 9, values 5 + 8 = 13).
e. Not known to be in NP: we know no polynomial-time way to verify that a subset is *optimal*, since our only optimum-finder is the O(nB) DP, which is pseudo-polynomial. Not known to be in P either. (Proving it's *not* in NP would prove P ≠ NP.)

9.11 · Select all: search problems, P and NP

Correct: 1, 2, 5, 6

1. ✔ Verify T is a spanning tree (BFS/DFS) and compare its weight with Kruskal's/Prim's MST: O(m log n).
2. ✔ Kruskal/Prim find it in polynomial time.
3. ✘ Not known: verifying optimality needs the pseudo-polynomial DP.
4. ✘ Nondeterministic polynomial time. P ⊆ NP.
5. ✔ Each clause has at most 2n literals, so O(n) per clause, m clauses (NP1 video).
6. ✔ If one NP-complete problem were in P, every NP problem would be (all reduce to it), so P = NP. Contrapositive.
7. ✘ Wrong direction: that only shows B is no harder than SAT. Reduce a known NP-complete problem TO B.

10 · SAT, Max-SAT, 2-SAT, 3-SAT

10.1 · Why is Max-SAT hard? (predicted exam question)

Reduce SAT → Max-SAT (SAT is NP-hard by Cook–Levin).

f: given a CNF formula φ with m clauses, pass the same φ to Max-SAT. Identity: poly time.

⇔ and g: run Max-SAT to get an assignment a maximizing the number of satisfied clauses. Output a if it satisfies all m clauses, otherwise “no”. If φ is satisfiable, the maximum is m, so a satisfies everything and we return a satisfying assignment. If φ isn't satisfiable, no assignment satisfies all m, so a doesn't and we correctly say “no”. Checking the count is linear time. So a poly-time Max-SAT algorithm would solve SAT, and Max-SAT is NP-hard.

(Max-SAT is not in NP as an optimization problem, so it is NP-hard, not NP-complete.)

10.2 · ILP is NP-hard

f: variables y_i (one per Boolean variable) and z_j (one per clause), all in {0,1}; maximize Σz_j subject to $z_j \le \Sigma_{i \in C_j^+} y_i + \Sigma_{i \in C_j^-} (1 - y_i)$. Size O(n + m) variables and m constraints: poly.

Correctness: for 0/1 y, the RHS is 0 exactly when every literal of C_j is false, which forces $z_j = 0$. Otherwise z_j can be 1, and maximization sets it. So the ILP optimum equals the max number of satisfiable clauses, achieved at y = the assignment. **g:** read x_i = T iff $y_i = 1$. Chain: SAT → Max-SAT → ILP, so ILP is NP-hard.

10.3 · 2-SAT on the board example

a. (x1∨x2): ¬x1→x2, ¬x2→x1. (x1∨¬x3): ¬x1→¬x3, x3→x1. (¬x2∨¬x4): x2→¬x4, x4→¬x2. (¬x3∨x4): x3→x4, ¬x4→¬x3.
b. No cycles, so all 8 SCCs are single vertices. No x shares an SCC with ¬x, so f is **satisfiable**.
c. Assigning in reverse topological order (x = T iff x's SCC comes after ¬x's) gives **x1 = T, x2 = T, x3 = F, x4 = F**. Check: (T∨T), (T∨T), (F∨T), (T∨F). All true.

10.4 · An unsatisfiable 2-SAT formula

Edges: ¬x1→x2, ¬x2→x1, ¬x1→¬x2, x2→x1, x1→x2, ¬x2→¬x1, x1→¬x2, x2→¬x1. Then x1 → x2 → ¬x1 and ¬x1 → x2 → x1, so x1 and ¬x1 are in the same SCC. In fact all four literals form 1 SCC. Same SCC ⇒ same truth value, but x1 and ¬x1 can't be equal. Unsatisfiable.

10.5 · Why SCCs decide 2-SAT

(a) The clause is false only when a = b = F. Exactly then ¬a → b fails (T → F) and ¬b → a fails. In all other cases both implications hold.

(b) Each edge u → v is an implication that a satisfying assignment must respect: if u is true then v is true. If u and v are in the same SCC, there are paths u ⇝ v and v ⇝ u, so u true ⇒ v true and v true ⇒ u true. Equal values.

(c) x and ¬x always have opposite values, so they can't be in one SCC of a satisfiable formula. (The converse, that otherwise it's satisfiable via the topological-order assignment, is the video's lemma.)

10.6 · Clause splitting: SAT → 3-SAT

(a) $C' = (x1 \vee x2 \vee y1) \wedge (\neg y1 \vee x3 \vee y2) \wedge (\neg y2 \vee x4 \vee y3) \wedge (\neg y3 \vee x5 \vee y4) \wedge (\neg y4 \vee x6 \vee x7)$.

(b) ⇒: let x_i be a true literal. Its clause is satisfied. Set $y_1, \ldots, y_j$ = T for every y appearing before x_i's clause and the rest F. Each clause to the left contains some y_t = T, and each clause to the right contains some $\neg y_t$ with y_t = F. ⇐ (contrapositive): if all x_i are false, clause 1 forces y_1 = T, then clause 2 forces y_2 = T, …, then the last clause $(\neg y_{k-3} \vee x_{k-1} \vee x_k)$ is false. So C′ is unsatisfiable.

(c) $k - 2 = 8$ clauses and $k - 3 = 7$ new variables. Each clause of length k becomes O(k) clauses, so the total size is linear in the formula. g: restrict the 3-SAT assignment to the original variables.

10.7 · Why doesn't the trick give 2-SAT?

Each middle clause needs three slots: the connector $\neg y_{t-1}$ from the previous clause, one original literal, and the connector y_t to the next. With two slots you can't both chain the clauses and carry an original literal, so the equivalence breaks. That's consistent with 2-SAT $\in$ P (unless P = NP, no such reduction exists).

10.8 · 3-SAT is NP-complete

In NP: the candidate solution is an assignment; the verifier evaluates each clause in linear time. **NP-hard:** SAT → 3-SAT by clause splitting (10.6). Clauses with ≤ 3 literals are copied unchanged, longer ones are split, and the original variables' values are kept.

10.9 · k-SAT for k ≥ 3

Reduce 3-SAT → k-SAT with f = identity: every 3-SAT formula already has ≤ 3 ≤ k literals per clause, so it is a k-SAT input with exactly the same satisfying assignments (g = identity). Polynomial trivially. (Reducing to a more general problem is often this easy. Compare VC → Set Cover.)

10.10 · Exact-4-SAT is NP-complete (DPV 8.8)

In NP. Given an assignment, check each clause in O(1) time: O(m).

Reduce 3-SAT → Exact-4-SAT. Let φ be a 3-SAT formula (every clause has 1 to 3 literals; remove repeated literals, and drop clauses containing both x and ¬x since they're always true).

f: for each clause C with $j < 4$ literals, create $4 - j$ *fresh* variables z and replace C by the 2^{4-j} clauses "C ∨ (one sign pattern of the z's)". For example, a 3-literal clause $(a \vee b \vee c)$ becomes $(a \vee b \vee c \vee z) \wedge (a \vee b \vee c \vee \neg z)$. Clause counts per original clause: 1 → 8, 2 → 4, 3 → 2 (literals → clauses). So there are ≤ 8m clauses and ≤ 3m new variables: polynomial (linear).

(⇒) If an assignment satisfies φ, keep it and set the z's arbitrarily. Every new clause contains an original clause C, which is true.

(⇐) Take an assignment satisfying f(φ) and suppose some original C is false under it. Among C's padded copies, pick the one whose z-literals are all false under this assignment (choose

each sign opposite to that z's value). That copy has every literal false: contradiction. So every C is true, and the assignment restricted to the original variables satisfies φ.

g: drop the z's. Linear. Together with “in NP”, Exact-4-SAT is NP-complete.

Shape lesson (§12.2): “same problem, extra size condition” ⇒ reduce from the original and pad.

10.11 · 2-SAT: eliminate unit clauses (GR2 example)

a. Unit ($\neg x_1$): set x_1 = F. Delete clauses containing $\neg x_1$ (the unit clause itself) and delete the false literal x_1 from ($x_1 \vee x_4$): $f' = (x_3 \vee \neg x_2) \wedge (x_4) \wedge (\neg x_4 \vee x_2) \wedge (\neg x_3 \vee x_4)$. Unit ($x_4$): set x_4 = T, delete (x_4) and ($\neg x_3 \vee x_4$), shrink ($\neg x_4 \vee x_2$) to (x_2): $(x_3 \vee \neg x_2) \wedge (x_2)$. Unit ($x_2$): set x_2 = T, giving (x_3). Unit (x_3): set x_3 = T, giving the empty formula.
b. Yes: x_1 = F, x_2 = T, x_3 = T, x_4 = T (the only satisfying assignment).
c. The unit clause (a) can only be satisfied by making a true, so every satisfying assignment does that. Clauses containing a are then already satisfied (safe to delete), and the literal ¬a is false (safe to delete from its clauses). Nothing else changes.
d. An empty clause can't be satisfied, so output NO. (This case is my addition; the video only shows the satisfiable path.) Otherwise, after elimination every clause has exactly 2 literals and the SCC algorithm takes over.

10.12 · Stingy SAT is NP-complete (DPV 8.3)

In NP: check that the assignment satisfies φ (O(nm)) and count the true variables (O(n)).

SAT → Stingy SAT: f(φ) = (φ, n) where n is the number of variables. **⇔:** every assignment has ≤ n true variables, so φ has a satisfying assignment ⇔ (φ, n) has a stingy one. **g** = identity. f and g are linear.

Shape lesson (§12.2): a generalization of a known hard problem (extra parameter k) ⇒ set the parameter so it becomes the special case.

10.13 · Why the 2-SAT assignment rule works (key fact)

a. Write the path as $\alpha = \gamma_0 \to \gamma_1 \to \cdots \to \gamma_\ell = \beta$. Each edge $\gamma_i \to \gamma_{i+1}$ comes from the clause ($\neg\gamma_i \vee \gamma_{i+1}$), which also contributes the edge $\neg\gamma_{i+1} \to \neg\gamma_i$. Following these edges backwards gives $\neg\beta = \neg\gamma_\ell \to \cdots \to \neg\gamma_0 = \neg\alpha$.
b. For $\alpha, \beta \in S$ there are paths both ways, so by (a) there are paths $\neg\beta \rightsquigarrow \neg\alpha$ and $\neg\alpha \rightsquigarrow \neg\beta$: all of $\bar{S}$ lies in one SCC. Applying the same argument to that SCC shows it can't contain anything outside $\bar{S}$, so $\bar{S}$ is exactly an SCC. If an edge $u \to \neg\alpha$ entered $\bar{S}$ from outside ($u \notin \bar{S}$, $\alpha \in S$), then by (a) the edge $\alpha \to \neg u$ would exist, with $\neg u \notin S$: an edge *leaving* S. A sink has none, so $\bar{S}$ has no incoming edges: it is a source.

c. A clause with a literal in S is satisfied, because that literal is true. A clause ($\ell \vee b$) with $\ell \in \bar{S}$ gives the edge $\neg\ell \to b$ with $\neg\ell \in S$. S is a sink, so $b \in S$, and b is true. $S \neq \bar{S}$ because no x shares an SCC with ¬x, so no literal gets both values. Removing $S \cup \bar{S}$ (closed under negation) leaves exactly the implication graph of the remaining clauses, with the same SCCs, so we can repeat. Total time O(n + m).

11 · Graph reductions: IS, Clique, VC, matching

11.1 · Build the 3-SAT → IS graph

a. Vertices (one per literal occurrence): x@C1, ¬x@C2, y@C2, ¬y@C3, z@C3, x@C3, so 6 vertices. Clause edges: ¬x@C2–y@C2; ¬y@C3–z@C3, ¬y@C3–x@C3, z@C3–x@C3. Complement edges: x@C1–¬x@C2, ¬x@C2–x@C3, y@C2–¬y@C3. Total 7 edges. k = m = **3**.
b. S = {x@C1, y@C2, z@C3} is independent and |S| = 3 ⇒ x = T, y = T, z = T. Check: (T), (F ∨ T), (F ∨ T ∨ T). All satisfied.
c. No: x@C1 and ¬x@C2 are joined by a complement edge (a contradiction: x and ¬x both "true").

11.2 · Prove Independent Set is NP-hard (full write-up)

In NP. Given S, check |S| ≥ k and that no edge joins two vertices of S: O(|V| + |E|).

Reduce 3-SAT → IS. Transformation f. For a formula φ with m clauses (≤ 3 literals each), make one vertex per literal occurrence. Join all vertices of the same clause (clause cliques), and join every pair of complementary literals x, ¬x, in any clauses. Set k = m. Size: ≤ 3m vertices, ≤ 3m clause edges, ≤ $(3m)^2$ complement edges. Polynomial.

(⇒) If φ is satisfiable, pick one true literal per clause. The m picked vertices are in different clauses (no clique edge), and no two are complementary because both would have to be true (no complement edge). So we get an independent set of size m.

(⇐) If G has an independent set S of size m, it has exactly one vertex per clause (the cliques allow at most one) and never both x and ¬x. Set those literals true and the remaining variables arbitrarily. This is consistent and satisfies every clause.

Recovery g. Given S, read off the assignment as above in linear time. If IS reports no set of size m, φ is unsatisfiable by (⇒).

11.3 · Independent set ⇔ clique in the complement

(a) For u, v ∈ S: $uv \notin E \Leftrightarrow uv \in \hat{E}$. "No pair in S is an edge of G" is the same as "every pair in S is an edge of $\hat{G}$".

(b) In NP: check all pairs of K are edges ($O(k^2)$). NP-hard: IS → Clique with $f(G, k) = (\hat{G}, k)$, which takes $O(n^2)$ to complement. By (a), G has an independent set of size k ⇔ $\hat{G}$ has a clique of size k, and g returns the same set.

(c) Yes: complementing twice gives back G, so Clique → IS is the same map. (Both are NP-complete, so reductions exist both ways.)

11.4 · Independent set ⇔ complement is a vertex cover

(a) ⇒: for any edge uv, u and v are not both in S, so one of them is in $V \setminus S$. ⇐: if $u, v \in S$ were adjacent, neither endpoint would be in $V \setminus S$, contradicting the cover.

(b) IS → VC: $(G, k) \mapsto (G, n - k)$. G has an independent set of size $\geq k$ ⇔ a vertex cover of size $\leq n - k$; g = complement. VC → IS: $(G, k) \mapsto (G, n - k)$, symmetric.

(c) Yes, that's the ⇐ direction of (a).

11.5 · Do reductions preserve approximation?

No. Take a perfect matching on n vertices (n/2 disjoint edges). Min VC = n/2, so a 2-approximation may legally return all n vertices. Its complement is ∅, but the max IS is n/2. The ratio is 0: no constant factor survives. (In the star, the min VC is just the center, and the max IS is everything else.) Exact reductions preserve optimal solutions, not approximation ratios.

11.6 · Bipartite matching via max flow

(a) Add s and t; s → u for all $u \in V_1$; direct each edge u → v (V_1 to V_2); v → t for all $v \in V_2$; all capacities 1. Poly time.

(b) Suppose two edges of M share $u \in V_1$. Then the flow out of u is ≥ 2, but the only edge into u is s → u, with capacity 1, so the flow in is ≤ 1. This contradicts conservation. The same argument at $v \in V_2$ (only one edge out, v → t) rules out sharing a V_2 endpoint. Conversely, each matching edge gives a unit path s → u → v → t, so max flow value = max matching size. (Uses that max-flow algorithms return integral flows when capacities are integers.)

(c) Matching → Max-Flow, and Max-Flow ∈ P, so bipartite matching ∈ P. Reductions to easy problems prove easiness.

11.7 · Lecture Q&A on the IS construction

(a) Fine: the clause is satisfied more than once, but the independent set takes exactly one vertex per clause, because the clause clique allows at most one. (b) Prevented by the

complement edge y–¬y: such an S isn't independent. (c) The cliques cap |S| at one per clause, so m is the maximum possible, and reaching m forces one true literal in every clause.

11.8 · Vertex Cover → Set Cover (textbook only)

f: U = E (the edges); for each vertex i, S_i = the edges incident to i; same k. A vertex cover $\{i_1, \ldots, i_\ell\}$ touches every edge ⇔ $S_{i_1} \cup \cdots \cup S_{i_\ell} = E$. So covers of size ≤ k correspond exactly. Poly time; g maps the chosen sets back to their vertices. Since VC is NP-hard, Set Cover is NP-hard (and in NP: check the union).

11.9 · True/False: graph reductions

a. **False.** Reductions in the other direction don't have to be inverses (Nov 12 Q&A). The 3-SAT → IS graphs are very special (small cliques plus links).
b. **True.** Complements: sizes add to n, so maximizing |S| minimizes |V\S|.
c. **False.** Perfect-matching example: the 2-approximate cover can be all of V, leaving ∅.
d. **True.** That's the ⇐ direction.
e. **False.** It shows matching is in P (reduction TO an easy problem). Hardness would need a reduction FROM a hard problem.
f. **True.** Both via the complement graph $\hat{G}$.

12 · Writing reductions: template drills

12.1 · Vertex Cover is NP-complete (full template)

1. In NP: given M, check |M| ≤ k and that every edge has an endpoint in M: O(|V| + |E|).

2. Known hard problem: Independent Set (NP-hard via 3-SAT). Reduce IS → VC.

3. f: (G, k) ↦ (G, n − k). O(1) extra work beyond copying G.

4. ⇔: (⇒) if S is independent with |S| ≥ k, then V\S covers every edge (an edge inside S would contradict independence), and |V\S| ≤ n − k. (⇐) If M is a cover with |M| ≤ n − k, then V\M is independent (an edge inside it would be uncovered), and |V\M| ≥ k.

5. Poly + recovery: f and g (g(M) = V\M) are linear time. If VC answers “none”, IS has none either, by (⇒).

12.2 · HALF-IS: independent set of exactly half the vertices

In NP: check size n′/2 and independence.

Reduce IS → HALF-IS. Input (G, k), n = |V|. Assume 2 ≤ k ≤ n (k = 1 is trivial).

f: If $2k \le n$, add $n - 2k$ *isolated* vertices: $n' = 2n - 2k$, target $n'/2 = n - k = k + (n - 2k)$. If $2k > n$, add $t = 2k - n$ *universal* vertices, adjacent to every vertex and to each other: $n' = 2k$, target k. Linear/quadratic size.

⇔ (case $2k \le n$): (⇒) an independent set of size k in G plus all $n - 2k$ isolated vertices is independent, of size $n - k$. (⇐) An independent set of size $n - k$ in G' uses at most $n - 2k$ new vertices, so it has $\ge k$ vertices of G, which are independent in G.

⇔ (case $2k > n$): (⇒) an independent set of size k in G is still independent in G' (no new edges among old vertices). (⇐) Any independent set containing a universal vertex has size $1 < k$ ($k \ge 2$), so an independent set of size k in G' lies entirely in G.

g: drop the added vertices. Poly time throughout.

12.3 · Set Packing is NP-hard

f: $U = E$; for each vertex v, S_v = the set of edges incident to v; same k. Poly.

⇔: $S_u \cap S_v \neq \emptyset \Leftrightarrow$ the edge uv exists (an edge is in both incidence sets exactly when it joins u and v). So a family $\{S_v : v \in T\}$ is pairwise disjoint ⇔ T is independent, and sizes match. **g:** T = the indices of the chosen sets. In NP: check pairwise disjointness. So Set Packing is NP-complete.

12.4 · Which problem should I reduce from?

a. Independent Set (a packing problem with conflicts as edges): identity-like map.
b. Vertex Cover (covering edges with vertices).
c. Clique (or IS on $\hat{G}$).
d. 3-SAT (identity, since $3 \le 4$), or SAT with clause splitting.
e. SAT → "max" version with the same instance (Max-SAT style: satisfiable ⇔ the optimum satisfies all).

12.5 · Spot the broken reduction

Student 1 reduced 2-SAT → IS. That only says IS is at least as hard as 2-SAT, which tells us nothing about 2-SAT (wrong direction). Hardness needs a known hard problem → 2-SAT, which would imply P = NP since 2-SAT ∈ P.

Student 2 proved only (⇒). Without (⇐), a clique of size m in $\hat{G}$ might not come from any satisfying assignment, so the black box's answer can't be trusted for "no" instances. Both directions are required.

15 · HW5 decoded: originals (hints only) + exam-style re-skins

15.1 · HW5 #19: Min cuts (original wording, full solution)

Stuck? Hint path (try one hint, then go back to the problem)

1. Part 1: Karger's theorem is about *each* fixed min cut. Can two different cuts be returned by the same run?
2. Part 2: redo the Karger product for a fixed cut of size $\leq \alpha\lambda$. The per-step bound becomes $2\alpha/t$. Stop when 2α supervertices remain, then output a random cut of what's left.

Full solution (what to write)

1. At most n(n−1)/2 min cuts. Let λ be the min-cut size and let $\Delta_1, \ldots, \Delta_t$ be all the distinct min cuts. **Fact (lecture, §4.2):** for any *fixed* min cut Δ, one run of the contraction algorithm outputs Δ with probability $\geq 2/(n(n-1))$. (Reason: with j supervertices left and Δ intact, every supervertex has degree $\geq \lambda$, so there are $\geq j\lambda/2$ edges and Pr[contract an edge of Δ] $\leq \lambda/(j\lambda/2) = 2/j$. Multiply $\prod_{j=3}^{n}(1 - 2/j) = 2/(n(n-1))$.)

Let A_i = "the run outputs Δ_i". A run outputs exactly one cut, so the events A_i are **disjoint**. Therefore $1 \geq \Pr[A_1 \cup \cdots \cup A_t] = \Sigma_i \Pr[A_i] \geq t \cdot 2/(n(n-1))$, so $t \leq n(n-1)/2 = O(n^2)$. (Tight: the cycle C_n has exactly this many, problem 15.5.)

2. α-small cuts. First assume 2α is an integer and set $r = 2\alpha$ (so $r \geq 3$, because $\alpha > 1$). Fix a cut $C = (S, T)$ with $|C| \leq \alpha\lambda$. **Modified algorithm:** contract uniformly random edges until r supervertices remain; then output a uniformly random partition of those r supervertices into two non-empty sides.

Contraction phase. Suppose j supervertices remain and no edge of C has been contracted yet. Each supervertex X is one side of a cut $(X, V \setminus X)$ of the original G, so $\deg(X) \geq \lambda$, and the multigraph has $\geq j\lambda/2$ edges. The next edge is in C with probability $\leq \alpha\lambda/(j\lambda/2) = 2\alpha/j$. So Pr[C survives until r remain] $\geq \prod_{j=r+1}^{n}(1 - r/j) = \prod_{j=r+1}^{n}(j - r)/j = (n - r)!\cdot r!/n! = 1/C(n, r)$.

Final phase. If C survived, each side of C is a union of supervertices, so C is one of the $2^{r-1} - 1$ two-sided partitions of the r supervertices. It is chosen with probability $1/(2^{r-1} - 1) \geq 2^{-(r-1)}$.

Counting. $\Pr[\text{output} = C] \geq 1/(C(n, 2\alpha)\cdot 2^{2\alpha-1})$ for every α-small C. The events "output = C" are disjoint over different C, so #α-small cuts $\leq C(n, 2\alpha)\cdot 2^{2\alpha-1} \leq n^{2\alpha}\cdot 2^{2\alpha-1}/(2\alpha)! = O_\alpha(n^{2\alpha})$.

If 2α is not an integer, stop at $r = \lceil 2\alpha \rceil$. Using $\ln(1 - x) \geq -x/(1 - x)$: $1 - 2\alpha/j \geq \exp(-2\alpha/(j - 2\alpha))$, and $\Sigma_{j=r+1}^{n} 1/(j - 2\alpha) \leq 1 + \ln n$ (first term ≤ 1, the rest $\leq \int ds/(s - 2\alpha)$). So the survival probability is $\geq e^{-2\alpha}\cdot n^{-2\alpha}$, and the count is $\leq e^{2\alpha}\cdot 2^{r-1}\cdot n^{2\alpha} = O_\alpha(n^{2\alpha})$. (Both product bounds are checked numerically in calc.py.)

15.2 · HW5 #20: Linearity of expectation (original wording, full solution)

Stuck? Hint path (try one hint, then go back to the problem)

1. (a) One indicator per clause. A clause with 3 distinct variables fails for exactly one of its 8 settings (§6.2).
2. (b) The expectation is an average over the 2^n equally likely assignments. If one term is below the average and none is above, what happens to the average? (§5.1)
3. (c) Look at the figure in §15.2 (k = 2, 4 clauses) and do the same with k = 3.

Full solution (what to write)

(a) Assume (as "exactly three literals" means) the three literals of each clause are on distinct variables. Let a random assignment set each variable T/F independently with probability ½. For clause C_j let $Y_j = 1$ if C_j is satisfied. C_j is false only if all three literals are false, which has probability $(½)^3 = 1/8$. So $E[Y_j] = 7/8$. The number of satisfied clauses is $W = \Sigma_j Y_j$. By **linearity of expectation** (no independence between clauses needed), $E[W] = 7m/8$ = 87.5% of m.

(b) The random assignment is uniform over all 2^n assignments, so $E[W] = (1/2^n)\cdot\Sigma_a W(a) = 7m/8$: the expectation is the plain average of W over all assignments. Suppose some a_0 has $W(a_0) < 7m/8$, and suppose for contradiction that no assignment has $W(a) > 7m/8$. Then every term is $\le 7m/8$ and one is strictly smaller, so $\Sigma_a W(a) < 2^n\cdot 7m/8$, i.e. $E[W] < 7m/8$. Contradiction. Hence some a_1 has $W(a_1) > 7m/8$.

(c) Take the 8 clauses on x_1, x_2, x_3 with all 2^3 sign patterns: $(x_1 \vee x_2 \vee x_3)$, $(x_1 \vee x_2 \vee \neg x_3)$, …, $(\neg x_1 \vee \neg x_2 \vee \neg x_3)$. Fix any assignment a. A clause is false under a iff each of its literals is false, which fixes the sign of every literal (use x_i if a_i = F, $\neg x_i$ if a_i = T). So **exactly one** of the 8 clauses is false, and every assignment satisfies exactly 7 of 8 = 87.5%. (Brute-force checked in calc.py.)

15.3 · HW5 #21: Randomized rounding for Hitting Set (original wording, full solution)

Stuck? Hint path (try one hint, then go back to the problem)

1. (a) One 0/1 variable per element i; one covering constraint per set S_j (compare HW4 #18 and Exam 2 Q4).
2. (b)–(c) This is the MAX-SAT rounding lemma (§6.4) with the clause replaced by S_j: a product, then AM–GM, then the constraint, then $(1 - 1/k)^k \le 1/e$.
3. (d) After t rounds, $\Pr[S_j \text{ still missed}] \le e^{-t}$. Count missed sets with linearity; for E[#rounds] use $E[T] = \Sigma \Pr[T > t]$.
4. (e) Each round adds expected cost $\Sigma x^*_i = LP^* \le OPT$.

Full solution (what to write)

(a) ILP. Variable $x_i \in \{0, 1\}$ for each element i = 1, …, n ($x_i = 1$ means $i \in S$).

minimize $\Sigma_i x_i$ subject to $\Sigma_{i \in S_j} x_i \ge 1$ for every j = 1, …, m, and $x_i \in \{0, 1\}$.

Equivalence: if S is a hitting set, its indicator vector satisfies every constraint (S contains some $i \in S_j$) and has objective $|S|$. Conversely a feasible 0/1 vector gives $S = \{i : x_i = 1\}$, which meets every S_j, with $|S|$ equal to the objective. So the optima are equal. The LP relaxation replaces $x_i \in \{0, 1\}$ by $0 \le x_i \le 1$. It is a relaxation of a *minimization*, so $LP^* = \Sigma_i x^*_i \le OPT$.

(b) The x_i are rounded independently. S misses S_j exactly when $x_i = 0$ for every $i \in S_j$, so $P(S \cap S_j = \emptyset) = \prod_{i \in S_j}(1 - x^*_i)$ and $\mathbf{P(S \cap S_j \neq \emptyset) = 1 - \prod_{i \in S_j}(1 - x^*_i)}$.

(c) Let $k = |S_j| \ge 1$. AM–GM on the k non-negative numbers $1 - x^*_i$: $\prod_{i \in S_j}(1 - x^*_i) \le ((1/k)\Sigma_{i \in S_j}(1 - x^*_i))^k = (1 - (1/k)\Sigma_{i \in S_j} x^*_i)^k \le (1 - 1/k)^k \le e^{-1}$. The middle step uses the LP constraint $\Sigma_{i \in S_j} x^*_i \ge 1$; the last uses $1 - y \le e^{-y}$. So $P(S \cap S_j \neq \emptyset) \ge 1 - 1/e \approx 0.632$.

(d) Rounds use fresh coins, so they are independent. For a fixed j, $P(S_j$ missed by all of the first t rounds$) = \prod$ over rounds of P(that round misses $S_j) \le e^{-t}$. Let X_t = number of sets not yet hit after t rounds. By linearity, $E[X_t] = \Sigma_j P(S_j \text{ missed}) \le m \cdot e^{-t}$, which is ≤ 1 for $t = \lceil \ln m \rceil$. Let T = number of rounds until every set is hit. Then $P(T > t) = P(X_t \ge 1) \le E[X_t] \le m\, e^{-t}$ (Markov). With $t_0 = \lceil \ln m \rceil$ (so $m\, e^{-t_0} \le 1$): $E[T] = \Sigma_{t \ge 0} P(T > t) \le t_0 + \Sigma_{s \ge 0} m\, e^{-t_0 - s} \le \lceil \ln m \rceil + 1/(1 - 1/e) \approx \lceil \ln m \rceil + 1.582 = O(\ln m)$.

(e) The output is the union of the T samples, so its size is $\le \Sigma_{r=1}^{T} |S^{(r)}|$. One round has $E|S^{(r)}| = \Sigma_i x^*_i = LP^*$. Whether round r happens ($T \ge r$) depends only on rounds 1, …, r − 1, so it is independent of $S^{(r)}$. Hence $E[\text{cost}] \le \Sigma_{r \ge 1} E[|S^{(r)}| \cdot 1\{T \ge r\}] = \Sigma_{r \ge 1} LP^* \cdot P(T \ge r) = LP^* \cdot E[T] \le (\lceil \ln m \rceil + 1.582) \cdot OPT$. The output is always a valid hitting set and its expected size is within a factor $\ln m + O(1) = O(\ln m)$ of OPT: an $O(\ln m)$-approximation (in expectation).

15.4 · HW5 #22: Zero-knowledge proofs and 3-coloring (original wording, full solution)

Stuck? Hint path (try one hint, then go back to the problem)

1. (a) The two colors are $\pi(c_1)$, $\pi(c_2)$ with $c_1 \neq c_2$ fixed and π uniform over all 6 permutations. List the pairs that can come out and how often.
2. (b) Whatever Raquel commits to in a round is some coloring, and it has a bad edge. What's the chance Brito picks it? Rounds use fresh coins.
3. (c) Boosting algebra (§4.3) with the roles reversed: make $(1 - 1/m)^N \le \varepsilon$ using $1 - x \le e^{-x}$.

Full solution (what to write)

(a) In one round Brito sees: the commitments (perfectly hiding, so they reveal nothing), the edge {u, v} he chose himself, and the two opened colors $(C'(u), C'(v)) = (\pi(c_1), \pi(c_2))$, where $c_1 = C(u) \neq c_2 = C(v)$ because C is proper. For any ordered pair (d_1, d_2) of *different* colors there is exactly one permutation π with $\pi(c_1) = d_1$ and $\pi(c_2) = d_2$ (the third color must go to the remaining one). π is uniform over the 3! = 6 permutations, so **the revealed pair is**

uniform over the 6 ordered pairs of distinct colors, each with probability 1/6. This distribution is the same for every proper coloring C and every edge. Brito could generate it himself (pick an edge, pick two distinct random colors) without talking to Raquel, so he learns nothing except that the endpoints differ, which is exactly what 3-colorability implies. A fresh π each round makes the rounds independent, so nothing builds up over N rounds.

(b) Fix a round. Because commitments are binding, they fix one assignment $C' : V \to \{1, 2, 3\}$ *before* Brito picks the edge. G is not 3-colorable, so some edge e* is monochromatic under C′. Brito's edge is uniform over the m edges and independent of C′, so he picks e* (and rejects) with probability ≥ 1/m. Hence P(pass a round) ≤ 1 − 1/m, even conditioned on everything that happened in earlier rounds (Brito's coins are fresh). Multiplying over rounds: **P(pass all N rounds) ≤ $(1 - 1/m)^N \le e^{-N/m}$**.

(c) It suffices that $e^{-N/m} \le \varepsilon \Leftrightarrow N/m \ge \ln(1/\varepsilon) \Leftrightarrow$ **$N \ge m \ln(1/\varepsilon)$**, so $N = \lceil m \ln(1/\varepsilon) \rceil$ works. (The exact minimum is $\lceil \ln \varepsilon / \ln(1 - 1/m) \rceil$, slightly smaller.) Example: m = 15, $\varepsilon = 10^{-3}$ gives N = 104 (exact minimum 101). An honest prover passes with probability 1: one-sided error.

15.5 · Re-skin of 19.1: the cycle is tight

a. Let $\Delta_1, \ldots, \Delta_t$ be the distinct min cuts and A_i the event "the run returns Δ_i". A run returns exactly one cut, so the A_i are disjoint. So $1 \ge \Pr[A_1 \cup \cdots \cup A_t] = \Sigma \Pr[A_i] \ge t \cdot 2/(n(n-1))$, giving $t \le n(n-1)/2 = O(n^2)$.
b. The cycle C_n: λ = 2, and removing *any* 2 of the n edges splits it into two arcs, a cut of size 2. That's $C(n,2) = n(n-1)/2$ min cuts. (Enumeration for C_5: 10 min cuts, each returned with probability exactly 1/10.)

15.6 · Re-skin of 19.2: α-small cuts of a cycle

a. Every cut of a cycle cuts an **even** number of edges, and any even-size edge set defines a cut (go around the cycle alternating sides). So the α-small cuts number C(n,2) + C(n,4). For n = 10: 45 + 210 = **255**.
b. Bound: $C(10,4) \cdot 2^3 = 1{,}680 \ge 255$ ✓. Since $C(n,4) = \Theta(n^4) = \Theta(n^{2\alpha})$, the exponent 2α can't be improved in general. (For n = 20: 5035 cuts vs bound 38,760.)

15.7 · Re-skin of 19.2: stopping the watermelon early (guided)

a. Every supervertex alone is a cut of G, so it has degree ≥ λ. So there are ≥ tλ/2 edges and Pr[pick a Δ-edge] ≤ λ/(tλ/2) = 2/t.
b. Chain rule over t = n, n−1, …, r+1: $\prod (1 - 2/t) = \prod (t-2)/t$, which telescopes to (r−1)r/((n−1)n). For n = 10, r = 4: **2/15**. (r = 2 recovers 2/(n(n−1)).)
c. Same argument with |C| ≤ αλ: Pr[pick a C-edge] ≤ αλ/(tλ/2) = k/t. $\prod_{t=k+1}^{n} (t-k)/t = k!\,(n-k)!/n! = 1/C(n,k)$.

d. If C survived, it is one of the $2^{k-1} - 1$ cuts of the remaining graph, and a uniform choice picks it with probability $\geq 2^{-(k-1)}$. So Pr[output C] $\geq 1/(C(n,k)\cdot 2^{k-1})$. The events "output C" are disjoint for different C, so the number of α-small cuts is $\leq C(n,k)\cdot 2^{k-1} \leq n^{2\alpha}\cdot 2^{2\alpha-1} = O_\alpha(n^{2\alpha})$. (Non-integer 2α: stop at $\lceil 2\alpha \rceil$ and bound $\prod(1 - 2\alpha/t)$ by $c_\alpha \cdot n^{-2\alpha}$ using $\ln(1 - x) \geq -x - x^2$; the same conclusion holds.)

15.8 · Re-skin of #20: MAX-Exact-4SAT

a. A clause fails only when all 4 literals are false: one of the 16 equally likely settings of its variables. $E[W_j] = 15/16$, and by linearity $E[W] = 15m/16$.
b. E[W] is the average of W(a) over all 2^n equally likely assignments a: $(1/2^n)\Sigma_a W(a) = 15m/16$. If $W(a_0) < 15m/16$ and every $W(a) \leq 15m/16$, the average would be strictly below 15m/16, a contradiction. So some $W(a) > 15m/16$. (Exactness matters: with exactly 4 distinct literals, E is exactly 15m/16.)
c. All 16 clauses on $x_1 \ldots x_4$ with every sign pattern. Each assignment falsifies exactly the one clause whose literals are all false under it, so it satisfies 15 of 16.
d. Some assignment achieves ≥ the mean 75/4, and counts are integers, so ≥ **19**.

15.9 · Re-skin of #20(c): exact-2 clauses

(a) $(x_1 \vee x_2) \wedge (x_1 \vee \neg x_2) \wedge (\neg x_1 \vee x_2) \wedge (\neg x_1 \vee \neg x_2)$: every assignment satisfies 3 of 4 (see the §15.2 figure). It's also the unsatisfiable 2-SAT example from §10.3. (b) A random assignment averages 3m/4, so some assignment reaches at least that. (c) $\geq \lceil 15/4 \rceil =$ **4**.

15.10 · Predicted re-skin of #21: randomized rounding for SET COVER (guided)

a. $\min \Sigma_j y_j$ s.t. $\Sigma_{j:\, e \in S_j} y_j \geq 1$ for every $e \in U$, $y_j \in \{0,1\}$ (ILP), or $0 \leq y_j \leq 1$ (LP). Every cover is LP-feasible with the same value and the LP minimizes over a superset, so LP* ≤ OPT.
b. e is missed only if none of its k sets is picked: $\Pr = \prod_i (1 - y^*_{j_i})$ (independence). AM–GM: $\leq (1 - (\Sigma_i y^*_{j_i})/k)^k \leq (1 - 1/k)^k \leq 1/e$, using the constraint $\Sigma \geq 1$ and $1 - x \leq e^{-x}$.
c. Rounds are independent: Pr[e missed by all t] $\leq (1/e)^t$. Union bound over n elements: $\leq n\cdot e^{-t}$. $t = \lceil \ln(4n) \rceil$ gives ≤ 1/4 (n = 100: t = 6; n = 1000: t = 9).
d. One round picks S_j with probability y^*_j, so E[# sets in a round] $= \Sigma\, y^*_j =$ LP* ≤ OPT. The union of t rounds has at most the sum of the round sizes, so by linearity $E \leq t\cdot LP^* \leq t\cdot OPT$.
e. Markov: Pr[size > 4t·OPT] ≤ 1/4. Union bound with (c): Pr[not a cover, or too big] ≤ 1/2. Repeat to boost (§4.3).

15.11 · Re-skin of #21(d)–(e): expected number of rounds and cost

a. $T > t$ means some element is uncovered after t rounds: $\le n\, e^{-t}$ by 15.10(c). Tail sum: $E[T] = \Sigma_{t\ge 0} \Pr[T > t] \le \Sigma_{t<\lceil \ln n\rceil} 1 + \Sigma_{s\ge 0} n\, e^{-\lceil \ln n\rceil - s} \le \lceil \ln n\rceil + \Sigma_{s\ge 0} e^{-s} = \lceil \ln n\rceil + 1/(1 - 1/e)$. ($1/(1 - 1/e) \approx 1.582$; $n = 100$ gives $E[T] \le 6.58$.)
b. Let C_r be the number of sets picked in round r, so the total is $\Sigma_r C_r \cdot 1\{T \ge r\}$. The event $\{T \ge r\}$ ("not done after $r - 1$ rounds") depends only on rounds $1\ldots r-1$, so it is independent of C_r, and $E[C_r \cdot 1\{T \ge r\}] = LP^* \cdot \Pr[T \ge r]$. Summing: $E[\text{total}] = LP^* \cdot \Sigma_r \Pr[T \ge r] = LP^* \cdot E[T] \le (\ln n + O(1)) \cdot OPT$. That is an $O(\log n)$-approximation in expectation.

15.12 · Tight instance for one round of rounding

$U = \{1\}$, $S_1 = \cdots = S_k = \{1\}$, $y^* = (1/k, \ldots, 1/k)$: $LP^* = 1 = OPT$. Element 1 is covered unless all k coins fail: $\Pr = 1 - (1 - 1/k)^k$ ($k = 3$: 19/27). Threshold rounding picks all k sets (each $y^* = 1/k \ge 1/k$), a cover of size $k = k \cdot OPT$. That's Exam 2 Q4's tight example. The two methods trade a factor k (deterministic, frequency k) against $O(\log n)$ (randomized, any instance).

15.13 · Re-skin of #22: a 2-coloring protocol

a. The endpoints have different colors $c_1 \ne c_2$, and π is uniform over the 2 permutations, so $(\pi(c_1), \pi(c_2))$ is (1,2) or (2,1), each with probability ½, whatever the coloring. Brito learns nothing beyond "the endpoints differ".
b. Any committed coloring of a non-bipartite graph has ≥ 1 monochromatic edge. Brito hits it with probability $\ge 1/m$, so a round is passed with probability $\le 1 - 1/m$. Rounds are independent (fresh π, fresh commitment, fresh challenge), so all N pass with probability $\le (1 - 1/m)^N \le e^{-N/m}$.
c. $e^{-N/m} \le \varepsilon \Leftrightarrow N \ge m \ln(1/\varepsilon)$. For $m = 15$, $\varepsilon = 10^{-3}$: $N =$ **104** (the exact minimum with $(14/15)^N \le 10^{-3}$ is 101).
d. Monte Carlo (fixed N rounds, can wrongly accept) with **one-sided** error: an honest prover is always accepted, and only a cheater can slip through.

15.14 · What breaks the protocol?

(a) Zero knowledge fails: across rounds Brito sees colors under one fixed relabeling, so revealed edges stitch together into Raquel's actual coloring (up to one renaming). (b) Soundness fails: a single color is never "wrong", so a cheater is never caught. (c) Soundness fails: after seeing {u, v}, Raquel just reveals two different colors, so every round passes even when G is not 3-colorable.

16 · Select-ALL drill (Exam 2 format)

16.1 · Karger / watermelon

Correct: 1, 2, 4

1. ✔ Two supervertices remain, which is a partition of V.
2. ✔ The theorem, for each fixed Δ.
3. ✘ Monte Carlo: it may return a non-minimum cut.
4. ✔ $p \geq 2/(n(n-1)) \geq 1/n^2$ gives failure $\leq (1 - 1/n^2)^{n^2 \ln n} \leq e^{-\ln n} = 1/n$.

16.2 · Linearity and indicators

Correct: 1, 3

1. ✔ Linearity, each edge ½.
2. ✘ Counterexample: a triangle never has all 3 edges crossing.
3. ✔ Probabilistic method (MU Lemma 6.2).
4. ✘ Counterexample: a triangle ($m/2 = 1.5$ is not an integer); also a single edge ($X \in \{0, 1\}$).

16.3 · MAX-SAT guarantees

Correct: 1, 4

1. ✔ Each nonempty clause has probability ≥ ½.
2. ✘ Counterexample: $(x) \wedge (\neg x)$, where the expectation is 1 of 2.
3. ✘ Only in expectation.
4. ✔ Averaging argument with $\alpha_k + \beta_k \geq 3/2$.

16.4 · Reduction direction

Correct: 1, 3

1. ✔ Hardness flows forward.
2. ✘ Counterexample: shortest path → SAT.
3. ✔ Easiness flows backward.
4. ✘ Counterexample: shortest path → SAT again.

16.5 · Classes

Correct: 1, 3

1. ✔ Solve, then verify by solving.
2. ✘ NP-hard problems need not be search problems.
3. ✔ SCCs of the implication graph.
4. ✘ NP-hard, but not in NP as an optimization problem, so not NP-complete.

16.6 · What a valid reduction needs (select one)

Correct: 3

1. ✘ Required.
2. ✘ Required.
3. ✔ Not required (Nov 10 Q&A).
4. ✘ Required in the search setting.

16.7 · Independent set / vertex cover / clique

Correct: 1, 2, 4

1. ✔ Both directions.
2. ✔ Edges flipped.
3. ✘ Counterexample: a perfect matching; the cover may be all of V.
4. ✔ Complements: optimal sets correspond.

16.8 · Repetition protocols

Correct: 1, 2

1. ✔ $(1 - 1/m)^N \le e^{-N/m}$.
2. ✔ $e^{-\ln(1/\varepsilon)} = \varepsilon$.
3. ✘ One-sided: honest provers always pass.
4. ✘ The revealed colors would stitch together the coloring (15.14a).

Mock Exam A: 70 minutes, two pages of notes, no calculator

M.1 · Mock A §1: short answer (40 points)

(a) 2 and 3. (1 ✘: Monte Carlo. 4 ✘: the degree argument counts every edge with multiplicity, and the output must be a real cut.)

(b) 2: $B \in P \Rightarrow A \in P$, and A is NP-hard ⇒ everything in NP is in P.

(c) 1 and 4. (2 ✘: never needs independence. 3 ✘: needs $X \ge 0$. Counterexample: $X = \pm 1$, $a = 1$ gives $\Pr = ½ > 0 = E[X]/a$.)

(d) $|V| = 7$ (one per literal occurrence), clause edges 5 (a triangle for C1, one edge each for C2 and C3), complement edges 4 (a–¬a, ¬c–c, b–¬b, c–¬c), total 9, $k = m = 3$. An independent set of size 3: ¬c@C1, ¬a@C2, ¬b@C3 ⇒ a = b = c = F. Check: (F∨F∨T), (T∨F), (T∨T). ✓

(e) $I = 1\{X \ge a\} \le X/a$ pointwise (uses $X \ge 0$). Take expectations: $\Pr[X \ge a] = E[I] \le E[X]/a$.

(f) 40 ≤ OPT ≤ 44. Lower: some assignment satisfies at least the mean 40 (probabilistic method). Upper: the LP relaxation of a maximization satisfies LP* ≥ OPT. (m = 48 and the ½-

guarantee give only weaker bounds: OPT ≤ 48, OPT ≥ 24.)

M.2 · Mock A §2: counting cuts (15 points)

(a) See 15.5(a): the events are disjoint and each has probability ≥ $2/(n(n-1))$. (b) See 15.7(a)–(b): per step ≥ $1 - 2/t$, telescoping to $r(r-1)/(n(n-1))$. (c) The cycles C_n (15.5b).

M.3 · Mock A §3: randomized rounding for set cover (20 points)

Full solution: problem 15.10 (a)–(e).

M.4 · Mock A §4: a reduction (25 points)

a. Given H: check $|H| \le k$ and that each S_j meets H, in $O(\Sigma|S_j|)$.
b. f(G = (V, E), k): universe [n] = V, one set $S_e = \{u, v\}$ for each edge e = uv, same k. (⇒) A vertex cover M with $|M| \le k$ contains an endpoint of every edge, i.e. $M \cap S_e \neq \emptyset$ for all e: a hitting set of size ≤ k. (⇐) A hitting set H with $|H| \le k$ meets every {u, v}, i.e. touches every edge: a vertex cover of size ≤ k.
c. f writes one 2-element set per edge, O(n + m). g = identity (H itself is the cover). If HITTING-SET reports none, G has no cover of size ≤ k, by (⇒).
d. All sets have size b = 2, so we get a **2-approximation for vertex cover**. No contradiction: this reduction maps solutions to solutions with the *same* cost (VC is literally the special case b = 2), so ratios carry over. The §11.4 failure was for IS ↔ VC, where costs transform as $k \mapsto n - k$.

Mock Exam B: 70 minutes, two pages of notes, no calculator

N.1 · Mock B §1: short answer (40 points)

(a) 1, 2, 3. (4 ✘: about 2^n repetitions would be needed.)

(b) 2: FROM the known problem TO the new one.

(c) 1, 3, 4. (2 ✘: LP* ≥ OPT for a maximization.)

(d) Edges (two per clause, ¬a → b and ¬b → a): x1→x3, x2→¬x3, x2→x1, x3→¬x2, ¬x1→x2, ¬x1→¬x2, ¬x2→x1, ¬x2→x3, ¬x3→¬x1, ¬x3→x2. SCCs: {x1, ¬x2, x3} and {¬x1, x2, ¬x3}. No variable shares an SCC with its negation ⇒ **satisfiable**. The assignment rule (x true iff SCC(x) is later in topological order) gives **x1 = T, x2 = F, x3 = T**. Check: (T∨F), (F∨T), (F∨T), (F∨T), (T∨T). ✓

(e) If Pr[X ≥ μ] = 0, all mass is on values < μ, so $\mu = \Sigma\, x \Pr[X = x] < \mu\, \Sigma \Pr[X = x] = \mu$, a contradiction.

(f) 18 ≤ OPT ≤ 30. Lower: a cut of size 18 exists. Upper: no cut exceeds m. The algorithm's guarantee (≥ m/2 = 15) is weaker than the observed 18, and "18 ≥ OPT/2" would only give OPT ≤ 36, weaker than 30. (Same trap as Exam 2 Q1(f): extra information can be useless.)

N.2 · Mock B §2: MAX-Exact-4SAT (15 points)

Full solution: problem 15.8 (a)–(c).

N.3 · Mock B §3: a challenge protocol (20 points)

(a), (b), (d): see 15.13. (c) $N \geq m \ln(10^6) =$ **553** (exact minimum 546).

N.4 · Mock B §4: Set Cover is NP-hard, then approximate it (25 points)

a. Check that the chosen sets number ≤ k and that their union is U: polynomial.
b. f(G, k): U = E; for each vertex v, S_v = the set of edges incident to v; same k. (⇒) A cover M with |M| ≤ k touches every edge, so $\cup_{v \in M} S_v = E$. (⇐) If $\cup_{v \in T} S_v = E$ with |T| ≤ k, every edge has an endpoint in T, so T is a vertex cover.
c. |U| = m, n sets, total size 2m: linear. g maps chosen sets to their vertices. A "none" answer is correct by (⇒).
d. No exact poly-time algorithm is known: one would give P = NP, since VC → SET-COVER and VC is NP-hard. So a polynomial-time algorithm with a *provable* O(log n) factor is the realistic goal: it always returns a valid cover whose expected size is within O(log n)·OPT, and it never needs to know OPT.