COS 445 — Final

Deadline Extended to Monday, May 12th at 11:00am.
Please read Ed # 640 for further deadline details.

- All problems on this exam are *no collaboration* problems.
- You *may not* discuss any aspect of any problems with *anyone except for the course staff*.
- You *may not* consult any external resources, the Internet, LLMs, etc.
- You *may* consult the course lecture notes on Ed, linked lecture videos on Ed, past PSets and solutions, any of the five course readings, past Ed discussion, or any notes directly linked on the course webpage (e.g. the cheatsheet, or notes on linear programming).
- You *may* discuss the test with the course staff, but **we will only answer clarification questions and will not give any guidance or hints**. You should feel free to ask any questions and let us judge whether or not to answer, but just know that we may choose to politely decline to answer. We may choose to answer questions with a response of "I'm sorry, but I'm not comfortable answering that question," or "it is within the scope of the exam for you to answer that question yourself" (or some variant of these).
- If you choose to ask a question on Ed, *ask it privately*. We will maintain a pinned FAQ for questions that are asked multiple times (please also reference this FAQ).
- Please upload each problem as a separate file via codePost, as usual. See Ed post # 640 about the Final Auxiliary, where you can optionally upload an honor pledge and scratchwork.
- You *may not* use late days on the exam. You must upload your solution by May 12th at 11:00am. If you are working down to the wire, upload your partial progress in advance. **There is no grace period for the exam**. In case of a true emergency, email both smweinberg@princeton.edu and mbraverm@princeton.edu your solutions asap.
  - To create some wiggle room between "you missed the extended deadline at 11:00am" and "we cannot grade your exam at all because it's after 11:30am", we will deduct one point on each submission per five minutes late (rounded down). That is, 11:00am - 11:04am will receive a zero-point deduction, 11:05am - 11:09am will receive a one-point deduction, etc.
- If you miss 11:30am on May 12th, **even by a minute**, university policy prohibits us from grading your exam without explicit permission from your dean. Please make sure you have something submitted by the deadline, and take into account that the server may be overloaded or sluggish near the end. For example, you may wish to treat 10:30am on May 12th as a "pencils down", to leave yourself enough time to safely upload.
- There are *no exceptions, extensions, etc.* to the exam policy (again, in case of a *truly exceptional* circumstance, you should reach out to your residential dean).

Problem 1: COS 445 Speedrun (70 points)

For each of the 7 problems below: **unless otherwise specified**[1] you do not need to show any work and can just state the answer. However, if you simply state an incorrect answer with no justification, we cannot award partial credit. You are encouraged to provide a **very brief** outline/justification in order to receive partial credit in the event of a tiny mistake. For example, we will award very significant partial credit if you clearly execute the correct outline, but make a mistake in implementation.

Part a: Stable Matching (10 points)

Find the matching output by student-proposing deferred acceptance in the following example. A reminder of the Deferred Acceptance algorithm is the Lecture Stable Matchings I.

Alice: Princeton $\succ$ Yale $\succ$ Harvard
Bob: Princeton $\succ$ Yale $\succ$ Harvard
Charlie: Princeton $\succ$ Harvard $\succ$ Yale

Princeton: Alice $\succ$ Bob $\succ$ Charlie
Harvard: Alice $\succ$ Charlie $\succ$ Bob
Yale: Bob $\succ$ Charlie $\succ$ Alice

Part b: Voting (10 points)

Definition 1 (Copeland Rule)**.** *For every pair of candidates a, b, give one point to whichever candidate a majority of voters prefer, tie-breaking in favor of the lexicographically-first candidate. Output the candidate with the most points, again tie-breaking in favor of the lexicographically-first candidate.*

Find the candidate (the candidates are the foods — Alice, Bob, and Charlie are voting to pick a restaurant) **output by the Copeland rule** in the following example.

Alice: Tacos $\succ$ Seafood $\succ$ Pasta
Bob: Tacos $\succ$ Seafood $\succ$ Pasta
Charlie: Pasta $\succ$ Seafood $\succ$ Tacos

Part c: Extensive Form Games (10 points)

Consider the extensive form game in Figure 1.

There are two players (named 1 and 2) and three rounds. First player 1 plays, then player 2, then player 1 again. The numbers on the leaves denote the payoffs to the first and second players, respectively (as labeled at the internal nodes of the tree). The labels on the edges denote the names of the actions they can play at that turn.

(i) **Find a subgame-perfect Nash equilibrium for this game.**

(ii) **Find a pure Nash equilibrium such that both players receive strictly higher payoff than in the subgame-perfect Nash equilibrium from (i).**

[1] If otherwise specified, you should follow the otherwise specifications.

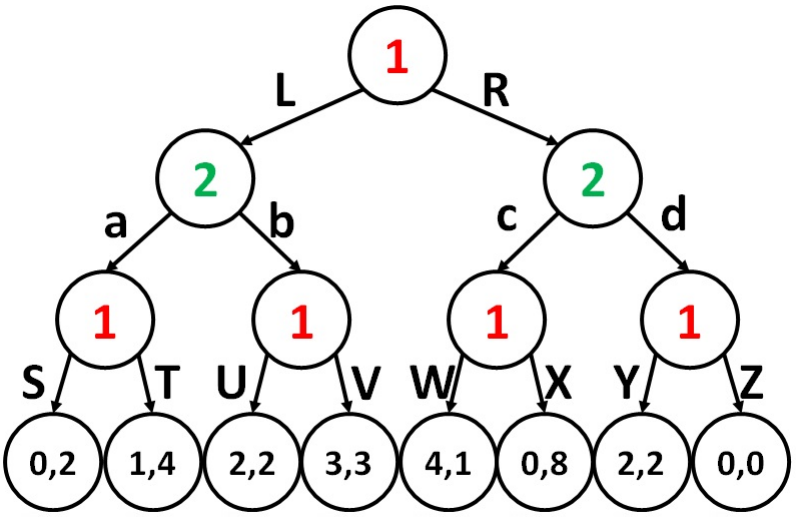

Figure 1: An extensive form game.

Recall that for both parts, a complete strategy lists a pure action for **every** internal node. For example, $\{L, S\}$ is not a complete strategy for player 1, nor is a a complete strategy for player 2. But $\{L, S, U, W, Y\}$ is a complete strategy for player 1, as is $\{a, c\}$ for player 2. A definition of Nash equilibria and subgame perfect Nash equilibria can be found in Lecture Game Theory II.

Part d: Linear Programming (10 points)

Write the dual of the following LP. **You do not need to solve the LP. You only need to write the dual.** A reminder of LP duality is in Lecture Linear Programming.

Maximize $3x + 2y$, such that:

- $4x + 4y \leq 1$.
- $3x + 7y \leq 3$.
- $x, y \geq 0$.

Part e: Welfare-maximizing Auctions (10 points)

There are three bidders, and two ad slots. The first ad slot has a click-through rate of 1, and the second has a click-through rate of $1/2$. The three bidders submit bids of $b_1 = 10, b_2 = 6, b_3 = 4$. The auctioneer is running a VCG auction (to assign each bidder at most one slot, and each slot to at most one bidder).

For each of the three bidders, state the slot they win and their payment (state the bidder's total payment, not their payment per-click).

A reminder of the VCG auction for sponsored search is in Lecture Auction Theory II.

Part f: Price of Anarchy (10 points)

Consider the network in Figure 2. There are two nodes, s and t, and one unit of flow traveling from s to t. There are two directed edges from s to t, one with cost $c(x) = 2$ and the other with cost

$c(x) = 1 + x$. **Compute the Price of Anarchy of this graph.**

A reminder of Price of Anarchy appears in Lecture Price of Anarchy I.

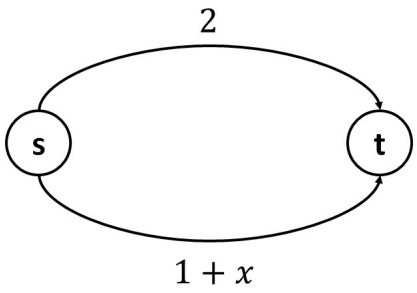

Figure 2: A routing network.

Part g: Time-Inconsistent Planning (10 points)

In the planning graph of Figure 3:

- What is the shortest path from s to t?
- What path is taken by a naive planner with present bias $b = 2$ from s to t?
- What path is taken by a sophisticated planner with present bias $b = 2$ to get from s to t?

A reminder the naive planner and sophisticated planner is in Lecture Behavioral Game Theory II.

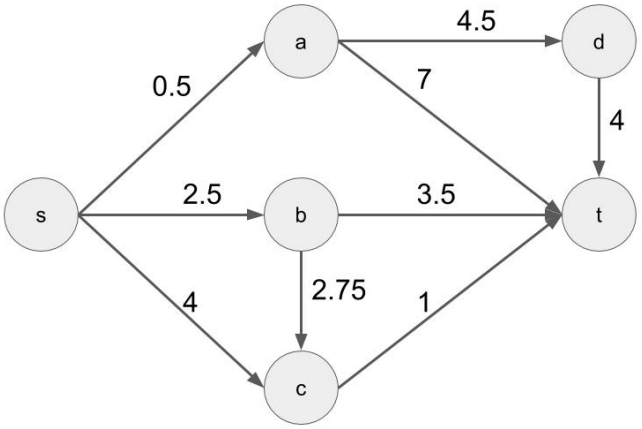

Figure 3: A planning graph.

Problem 2: Don't be Greedy with VCG (50 points)

This problem asks you to consider an extension of VCG, and is essentially asking you to work through in detail the proof given in lecture.

Let ALG be any algorithm that takes as input n valuation functions $b_1, \ldots, b_n$ and selects an outcome in A, $\text{ALG}(\vec{b})$. ALG does not necessarily select the outcome $x^* = \arg\max_{x \in A}\{\sum_i b_i(x)\}$ – it might select a different outcome. Below is a copy/paste of a proof from Lecture Welfare Optimal Auctions defining the VCG mechanism and proving that VCG is truthful. I have noted ~~deletions~~ and **additions** throughout the proof.[2] If ALG happens to be optimal, and select the outcome $x^* = \arg\max_{x \in A}\{\sum_i b_i(x)\}$, we promise that the proof below is correct, and is simply changing notation from lecture. We do not promise that the proof below is correct for any ALG – this problem asks you to decide!

ALG-Assisted VCG

Now, let's try to use our same three step process to design a truthful auction that ~~maximizes welfare~~ **selects the same outcome as ALG** in these extremely general abstract settings.

1. Like in the second-price auction, we'll ask each bidder for their valuation function $v_i : A \to \mathbb{R}_+$. So the strategy space is $\mathbb{R}_+^{|A|}$, because we're asking each bidder to list a non-negative real number $b_i(x)$ for every $x \in A$.

2. We want to ~~maximize welfare~~ **select the same outcome as ALG**, so we'll select the outcome $x^* = \text{ALG}(b_1, \ldots, b_n)$.

3. Again, the tricky step is to figure out what prices we should charge to make the auction incentive compatible. We'll just jump straight to the right answer – like in the sponsored search example, the right idea is to charge each bidder the "harm" they inflict on others by being considered. These are called the Clarke pivot payments.

Definition 2 (**ALG-assisted** Clarke Pivot Payments)**.** ***For simplicity of notation, let*** $\vec{b}_{-i}$ ***denote the vector of bids that replaces*** b_i ***with a valuation that has*** -2 ***for all outcomes.*** *Let* $x^* := \text{ALG}(b_1, \ldots, b_n)$*. The* **ALG-assisted** Clarke pivot payment p_i *for bidder* i *is defined*[3]

$$p_i \quad := \quad \sum_{j \neq i} b_j(\text{ALG}(\vec{b}_{-i})) - \sum_{j \neq i} b_j(x^*) \ .$$

Observe that p_i is equal to the total welfare that all bidders except for i *would have had* if i were not present (**and we use ALG**), minus the total welfare that all bidders except i *currently have* because i is present (**and we use ALG**). Intuitively, what the **ALG-assisted** Clarke pivot payments are getting at is that bidder i causes us to pick x^* instead of $\text{ALG}(\vec{b}_{-i})$, which causes the bidders other than i to be less happy overall – we'll charge bidder i exactly this amount. Now, we will show that the VCG auction is truthful and never overcharges bidders.

Proposition 3. *If bidder* i *reports* $b_i(\cdot)$*, bidder* i*'s payment is at most* $b_i(x^*)$*. This implies that bidder* i *gets non-negative utility when reporting truthfully.*

[2] Unless the additions are inside a math environment – in that case, they appear as regular text. It will be obvious when such additions occur, however, because they use ALG.

[3] The definition below replaces $\max_{x \in A}\left\{\sum_{j \neq i} b_j(x))\right\}$ with $\sum_{j \neq i} b_j(\text{ALG}(\vec{b}_{-i}))$.

Proof. Since $v_i(\cdot)$ is non-negative, the ~~optimal~~ welfare for all bidders **in the outcome** $\textbf{ALG}(\vec{b})$ is at least the ~~optimal~~ welfare for all bidders except i **in the outcome** $\textbf{ALG}(\vec{b}_{-i})$, and so[4]

$$\begin{aligned}
\sum_{j \neq i} b_j(\text{ALG}(\vec{b}_{-i})) &\leq \sum_j b_j(x^*) \\
\sum_{j \neq i} b_j(\text{ALG}(\vec{b}_{-i})) &\leq \sum_{j \neq i} b_j(x^*) + b_i(x^*) \\
\sum_{j \neq i} b_j(\text{ALG}(\vec{b}_{-i})) - \sum_{j \neq i} b_j(x^*) &\leq b_i(x^*) \\
p_i &\leq b_i(x^*) \ .
\end{aligned}$$

□

Theorem 4. *The* **ALG-*assisted*** *VCG auction is incentive compatible.*

Proof. For any bidder i, fix any $v_i(\cdot)$ and any $\vec{b}_{-i}$, and suppose bidder i reports $b_i(\cdot)$. Let $x^* := \text{ALG}(\vec{b})$ be the outcome that is selected, and observe that bidder i's utility is[5]

$$v_i(x^*) - p_i = v_i(x^*) + \sum_{j \neq i} b_j(x^*) - \sum_{j \neq i} b_j(\text{ALG}(\vec{b}_{-i})) \ .$$

Observe that bidder i cannot influence the last term, which only depends on $\vec{b}_{-i}$. In other words, bidder i will pay this term no matter what they report, and their utility-maximizing bid is the one which maximizes

$$v_i(x^*) + \sum_{j \neq i} b_j(x^*) \ .$$

Then because[6]

$$x^* := \text{ALG}(\vec{b}) \ ,$$

the ~~utility-maximizing~~ bid **that maximizes** $v_i(x^*) + \sum_{j \neq i} b_j(x^*)$ is exactly $b_i(\cdot) = v_i(\cdot)$! This means that bidding $v_i(\cdot)$ is always a best response for bidder i, so the **ALG-assisted** VCG auction is incentive compatible. We'll again omit the calculations, but it's also the case that truth telling is a (weakly) dominant strategy. □

Your mission

You must decide if Proposition 3 and Theorem 4 are correct *no matter what Algorithm* ALG *is* (we have already promised it is correct when ALG selects the welfare-maximizing outcome). Explicitly, you must do the following:

- Write one of the following statements. To make it easier on the graders, please write the sentence verbatim, and include the number from the list. We promise that exactly one of the statements below are true, and the others are false.

[4] Below, the LHS of all three equations replaces $\max_{x \in A} \left\{ \sum_{j \neq i} b_j(x)) \right\}$ with $\sum_{j \neq i} b_j(\text{ALG}(\vec{b}_{-i}))$, and nothing else changes.

[5] Below, we replace $\max_{x \in A} \left\{ \sum_{j \neq i} b_j(x) \right\}$ with $\sum_{j \neq i} b_j(\text{ALG}(\vec{b}_{-i}))$.

[6] Below, we replace $\text{argmax}_{x \in A} \left\{ b_i(x) + \sum_{j \neq i} b_j(x) \right\}$ with $\text{ALG}(\vec{b})$.

1. Proposition 3 is correct. This does not contradict my work on the error-prone second-price auction because the error-prone second-price auction is not a special case of an ALG-assisted VCG auction. [You must then provide a brief proof of this claim – 2-3 sentences should suffice.]
2. The first sentence in the proof of Proposition 3 is false. Here is a counterexample when ALG is the error-prone argmax. [You must then provide a counterexample and explain why it establishes your claim.]
3. The first line in the display math in the proof of Proposition 3 does not follow from the first sentence. Here is a counterexample when ALG is the error-prone argmax. [You must then provide a counterexample and explain why it establishes your claim.]
4. The second line in the display math in the proof of Proposition 3 does not follow from the first line. Here is a counterexample when ALG is the error-prone argmax. [You must then provide a counterexample and explain why it establishes your claim.]
5. The third line in the display math in the proof of Proposition 3 does not follow from the second line. Here is a counterexample when ALG is the error-prone argmax. [You must then provide a counterexample and explain why it establishes your claim.]
6. The fourth line in the display math in the proof of Proposition 3 does not follow from the third line. Here is a counterexample when ALG is the error-prone argmax. [You must then provide a counterexample and explain why it establishes your claim.]
7. Proposition 3 does not follow from the fourth line in the display math in the proof of Proposition 3. Here is a counterexample when ALG is the error-prone argmax. [You must then provide a counterexample and explain why it establishes your claim.]

Write one of the following statements. To make it easier on the graders, please write the sentence verbatim, and include the number from the list. We promise that exactly one of the statements below are true, and the others are false.[7]

1. Theorem 4 is correct. This does not contradict my work on the error-prone second-price auction because the error-prone second-price auction is not a special case of an ALG-assisted VCG auction. [You must then provide a brief proof of this claim – 2-3 sentences should suffice. If you selected this option for your first statement, you can write 'see above', or simply copy/paste.]
2. Bidder i's utility is not $v_i(x^*) - p_i$. Here is a counterexample when ALG is the error-prone argmaximizer. [You must then provide a counterexample and explain why it establishes your claim.]
3. The equality in the first display math is false. Here is a counterexample when ALG is the error-prone argmaximizer. [You must then provide a counterexample and explain why it establishes your claim.]
4. The sentence beginning "Observe,...," is false. Here is a counterexample when ALG is the error-prone argmaximizer. [You must then provide a counterexample and explain why it establishes your claim.]
5. The sentence beginning "In other words,...," (including the display math) is false. Here is a counterexample when ALG is the error-prone argmaximizer. [You must then provide a counterexample and explain why it establishes your claim.]

[7] Recall that "X follows from Y" means that "If Y is true, then X must be true." "X does not follow from Y" means "it is possible for X to be false even though Y is true."

6. The sentence beginning "Then because ...," (including the display math) is false. Here is a counterexample when ALG is the error-prone argmaximizer. [You must then provide a counterexample and explain why it establishes your claim.]
7. The sentence beginning "Then means that ...," does not follow from the previous sentence. Here is a counterexample when ALG is the error-prone argmaximizer. [You must then provide a counterexample and explain why it establishes your claim.]

Above, we intend to provide clear instructions no matter which option you select. If you find the instructions for one option to be more clear than others, please do *not* infer that this means the more-clear instructions are for the correct choice (instead, please ask us to clarify the unclear instructions, because we intend the intsructions to be fully clear for all options, including incorrect ones).

In total, you should write two statements. Each statement should be briefly justified as described by connecting to the error-prone argmaximizer/second-price auction.

Problem 3: Circular Fair Division (80 points)

For this problem, there are n players, and a single cake represented as the interval $[0, 1]$. Each player i has an additive, normalized, divisible valuation $V_i(\cdot)$ over the cake (refer to Lecture Fair Division for definitions of these terms).

This problem asks you to design a protocol. Your protocol is allowed to take any well-defined operation based on a single V_i in one step (but it is not allowed take a step that requires access to multiple valuations – it must break such a step down into smaller substeps that access only a single V_i at a time). If it helps, you can look at the description of the Dubins/Spanier and Selfridge/Conway protocols in Lecture Fair Division to see the expected level of detail for a single step. For example, your algorithm:

- Can "let x_1 be such that $V_1([0, x_1]) = 1/2$. Note that such an x_1 must exist as V_1 is normalized and divisible."
- Can "given that $V_1(S) > V_1(T)$, trim S into $S' \sqcup S''$ such that $V_1(S') = V_1(T)$. Note that this is possible as V_1 is divisible."
- Cannot "let x be such that $V_1([x, 2x]) = 1/2$" (because such an x might not exist, unless you prove it).
- Cannot "let S_1, S_2 be such that $V_1(S_1) = V_2(S_2)$" (because this requires access to both V_1 and V_2. If you want to get such an S_1, S_2, you will have to break this down into further steps).

Definition 5 (Envy-Free-One)**.** *A partition of the cake into $S_1, \ldots, S_n$ is* EF1 *if for all Players i, $V_i(S_i) \geq V_i(S_{i+1 \pmod n})$. That is, a partition is EF1 if no Player i envies Player $i + 1 \pmod n$.*[8]

Design a protocol that finds an EF1 partition for n players with additive, normalized, divisible valuations, and prove that your protocol is correct.

For full credit, your protocol should have use a number of steps polynomial in n. For a proof to be 'complete', it must show that the protocol is correct, but does not need to analyze its runtime,[9], nor explicitly confirm that each 'step' touches only a single valuation.[10]

[8] Put another way, the players sit in a circle, with Player $i + 1$ to the right of Player i, and Player 1 to the right of Player n. A partition is EF1 if no player envies the player to their right.

[9] Most reasonable protocols, including ones we've seen in class, use polynomial steps. The purpose of this requirement is rule out trivial (or non-trivial) exhaustive search approaches.

[10] Most reasonable protocols, including ones we've seen in class, are stated using steps that touch only a single valuation at a time. The purpose of this requirement is to rule out trivial protocols such as "cut the cake into an EF1 allocation, then award those pieces to the players."

Problem 4: Re-Optimized Matchings (100 points)

Definition 6 (Odd)**.** *A matching is Odd for $\vec{\succ}$ if everyone on both sides is matched to a partner they rank at an odd position.*[11] [12]

Design an algorithm that takes as input a stable matching instance $\vec{\succ}$ of size n (there are n on each side, and each side has strict, complete preferences over the other side) and outputs a **single matching** M that is **both Stable for $\vec{\succ}$ and Odd for $\vec{\succ}$**, if such a matching exists. If such an M does not exist, your algorithm should output "FAIL." Prove that your algorithm is correct.

For full credit, your algorithm must run in time polynomial in n. For a proof to be 'complete', it must show that the algorithm is correct, but does not need to analyze its runtime.[13]

Note: Depending on how you approach the problem, you may or may not find it helpful to cite concrete lemmas from Lecture 1, Lecture 2, or the staff solutions to Midterm Problem 4. If you choose to do so, you must be absolutely sure that you cite them exactly and apply them correctly.

[11] For example, if everyone is matched to their seventh choice, the matching is Odd. If a single participant is matched to their second choice, the matching is not Odd (even if all other participants are matched to their third choice).

[12] For example, if $n = 2$, a matching is Odd if and only if everyone gets their first choice.

[13] Most reasonable algorithms, including ones we've seen in class, run in polynomial time. The purpose of this requirement is to rule out 'trivial' brute-force solutions, such as one iterating over all $n!$ matchings and checking whether each one satisfies the condition.