Economics and Computation (Spring 2025)
Assignment #4
— Solutions —

Due: 4/7/2025 11:59pm ET

Problem 1: Indivisible goods

[**20 points**] Recall that envy-freeness up to *any* good (EFX) is known to be feasible for up to three players with additive valuations, but it is an open problem if it is feasible for four or more players.

Assume, then, that the n players have additive, *identical* valuations, i.e., for all $g \in G$ and $i, j \in N$, $V_i(g) = V_j(g)$. Under this assumption, design a polynomial-time algorithm that computes an EFX allocation.

Solution:

Algorithm. A greedy algorithm will do. We order the items from most to least valuable as $g_1, \ldots, g_m$, so $v(g_1) \geq v(g_2) \geq \cdots \geq v(g_m)$. We initialize empty bundles for each player $A_1, \ldots, A_n = \emptyset$. For each item $g \in G$, assign g to the player i with the currently smallest value for their bundle A_i.

EFX. There are two possible cases to describe the envy relation between any two players i, j. Note that all players have identical valuations, which we will denote with v.

1. $v(A_i) \geq v(A_j)$: Then clearly i does not envy j.
2. $v(A_i) < v(A_j)$: Then we want to show that EFX is still satisfied as in $v(A_i) \geq v(A_j \setminus \{g\})$. In other words, no player i should envy another player j's bundle after removing any one good from it. Since the valuations are additive and identical, we only need to consider the removal of the least valuable item g^*, as the result of removing any other item follows. Given the greedy algorithm, we know that g^* must have been allocated to j because $A_j \setminus \{g^*\}$ was, at that step, weakly worse off than any A_i. Therefore, the EFX condition holds.

Time Complexity.

1. Sorting the m goods takes $O(m \log m)$.
2. Distributing the m goods requires us to check for the poorest of the n players each time, so this step takes $O(mn)$.

In total, the time complexity is $O(m \log m + mn)$, which is polynomial. Other more efficient algorithms may exist.

Problem 2: Online matching algorithms

[**20 points**] Consider a variant on the worst-case online matching model introduced in Slide 7 of Lecture 13. As before, an adversary constructs the set of vertices that will arrive, but the arrival order of these vertices is uniform at random. If the algorithm is deterministic, the "game" proceeds as follows: we announce the algorithm, the adversary constructs the n vertices in V (i.e., defines the edges incident on each vertex in V), and then a random permutation π of V determines the arrival order. The algorithm has competitive ratio α if $\mathbb{E}[ALG(G, \pi)]/OPT(G) \geq \alpha$ for every graph G, where the expectation is taken over the randomness of the permutation π.

Prove that, in this model, the competitive ratio of a deterministic algorithm must be at most 3/4.

Solution: Consider an instance with two offline vertices u_1, u_2, and two online vertices v_1, v_2. v_1 has edges to both offline vertices, but v_2 has an edge to only one of the offline vertices.

The "adversary" observes what the deterministic algorithm would do when v_1 arrives first: if it is matched with u_1, the instance is such that v_2 can only be matched with u_1, and if v_1 is matched with u_2, v_2 can only be matched with u_2.

Now, with probability 1/2, v_1 arrives first, and the algorithm only matches a single edge. Therefore, the algorithm can match at most 3/2 edges in expectation (over the random arrival order of v_1 and v_2), compared to an optimum of 2, so the ratio is 3/4.

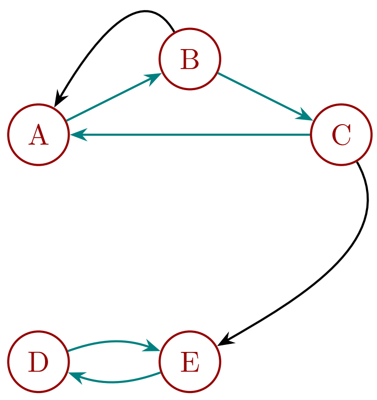
Problem 3: Kidney exchange

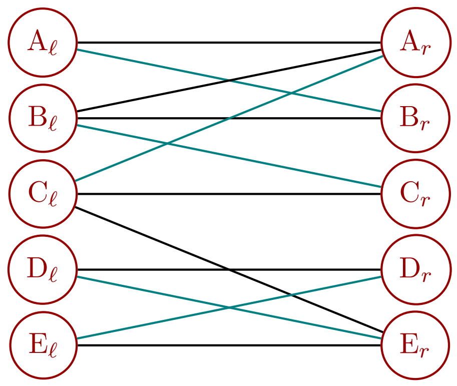
[**15 points**] In the Cycle Cover problem, we are given a directed graph and two integers k, t; we are asked whether it is possible to cover at least t vertices with disjoint cycles of length at most k. We stated in class (lecture 14, slide 6) that the Cycle Cover problem is NP-hard when there is a given upper bound k on the length of cycles. Show that if there is no such upper bound then the problem can be solved in polynomial time.

Hint: You may rely on the fact that a maximum weight perfect matching in a bipartite graph can be computed in polynomial time.

Solution:
Graph construction: Create a bipartite graph with two sets of vertices: V_R and V_L, each corresponding to the vertices of the original graph such that $|V_R| = |V_L| = |V|$. Each vertex $v \in V$ will have $v_l \in V_L$ and $v_r \in V_R$. Add an edge of weight 0 between each u_L and u_R. For each directed edge $(u, v) \in E$ in the original graph, create an edge from u_L to v_R in the bipartite graph. I.e., we add an edge between the vertices of the two sets whenever there is a directed edge between corresponding vertices in the original graph. Assign a weight of 1 to each edge in the bipartite graph. This way, the weight reflects whether an edge can be included in the cycle cover.

Directed graph





Bipartite Graph

Reduction to maximum weight perfect matching in a bipartite graph: Now, we need to find a maximum weight perfect matching in the bipartite graph, where a perfect matching is a matching that covers all vertices in V_R and V_L. A perfect matching would correspond to a set of disjoint cycles in the original graph. This is because each edge in the perfect matching corresponds to a directed edge in a cycle of the original graph. Since each vertex in the original graph has exactly one incoming and one outgoing edge in the matching, we can group these edges into disjoint cycles in the original graph. Hence there is a perfect matching with weight t in this graph if and only if there is a valid cycle cover that covers t vertices.

After finding the maximum weight perfect matching, we check the number of vertices involved in the cycles (i.e., the vertices that are part of the matching). If at least t vertices are covered, then we have a valid solution to the Cycle Cover problem. The main computational task here is finding a maximum weight perfect matching in a bipartite graph, which can be done in polynomial time.

Problem 4: Stable matching

[**15 points**] Prove that no bipartite matching mechanism with two-sided preferences is strategyproof (on both sides) and stable.

Guidance: Consider an instance with three students, three courses, and the following preferences:

s_1	s_2	s_3
t_1	t_2	t_1
t_2	t_1	t_2
t_3	t_3	t_3

t_1	t_2	t_3
s_2	s_1	s_1
s_1	s_2	s_2
s_3	s_3	s_3

There are two stable matchings; argue that in each them, one of the players can report a ranking leading to a unique stable matching that is (truly) better for them.

Solution: Assume for contradiction that there exists some two-sided matching that is both stable and strategy-proof for the student and teacher example. There are two stable matchings:

- $(s_1, t_1), (s_2, t_2), (s_3, t_3)$
- $(s_1, t_2), (s_2, t_1), (s_3, t_3)$

Assume that there exists a mechanism V that is both stable and strategy-proof. This means that no student or teacher should benefit from misreporting their preferences, and the matching produced must be stable for the reported preferences.

If V resulted in the first stable matching, t_1 can misreport their preferences as $s_2 \succ s_3 \succ s_1$ so their preference order changes. In this case, the only stable matching is $(s_1, t_2), (s_2, t_1), (s_3, t_3)$. This can be verified by inspecting the possible matchings and checking stability:

- (s_1, t_1) not stable because t_1 prefers s_3 (according to reported preferences), and s_3 prefers t_1
- (s_1, t_3) not stable because s_1 prefers t_2 over t_3 and t_2's top choice is s_1 so t_2 would switch
- (s_2, t_3) not stable because s_2 prefers t_1 over t_3 and t_1's top choice is s_2 so t_1 would switch

It follows that V must output it (under t_1's deviation), and thus t_1 strictly gains by misreporting their preferences. Hence V is not strategy-proof, a contradiction. We can apply the same logic in the case that V selects the second stable matching.

If $n > 3$, we can simply add more students and teachers where each new student i ranks teacher i first, and each new teacher i ranks student i first. This forces the matching mechanism to match student i with teahcer i for all $i > 3$ and we can repeat the argument above for the original three students and teachers.

Problem 5: Formulate a research question

[**30 points**] Formulate a research question that is relevant to one of the topics covered in this assignment: indivisible goods, online matching algorithms, kidney exchange, and stable matching. Refer to this document for guidelines.

During the process of formulating your question, keep track of your findings in a "research journal." At a minimum, it should include brainstorming ideas for questions and notes on relevant papers that you have identified.

Please submit the following deliverables:

1. Your research question.
2. A *brief* explanation of why it satisfies each of the following criteria:
   (a) Relevant: Which course topics is the question related to?
   (b) Nontrivial: What is an immediate way of attempting to answer the question and why does it fail?
   (c) Feasible: How would you tackle the question if you had the entire semester?
   (d) Novel: List the 1–3 most closely related papers that you have identified in your literature review and explain how your question differs.

   Note: *Your writeup of all four parts of Item 2 must be at most two pages long overall.*
3. Append your research journal to the PDF that contains your solutions. The research journal will not be graded; it is there to show your work.