EECS 336: Algorithms **Homework 9 Solutions**

1 No Free Lunch

This problem is NP-Complete. We reduce from Set Cover.

Construction: An input to Set Cover is a list $E = \{1, \ldots, n\}$ of elements, a collection of subsets of $\{1, \ldots, n\}$, which we will call S_j for $j \in \{1, \ldots, m\}$, and an integer k. We will create a child e_i for each element in E, and a child s_j for each set S_j. We will create a food f_j for each of the sets j.

Each "element" child e_i's preference ordering will rank all the foods corresponding to sets containing i in an arbitrary order, followed by all other foods. Each "set" child s_j's preference order will have f_j at the top, followed by all other foods in an arbitrary order.

Set the cost of child e_i for any food corresponding to a set containing i to be 0, and the cost of all other foods to be infinity. Set the cost of child s_j to be 1 for f_j and 0 otherwise.

The algorithm to solve Set Cover runs the black box to No Free Lunch with the k we were given as input to set cover, and returns the result.

Correctness: Assume there is a set cover $S \subseteq \{1, \ldots, m\}$ which uses k sets. We will construct a solution to the No Free Lunch instance with total cost less than or equal to k. To do so, choose the foods corresponding to the sets in S. We break the total cost of this choice into the total cost from the "element" children and the total cost from the "set" children.

- The total cost from the element children is 0. This follows from the fact that each element is covered by S, so each "element" child will have a food which has cost 0 available.
- The total cost from the "set" children is k. For every child corresponding to a set not in S, the only foods that child can choose are foods for which that child's cost is 0. For every child corresponding to a set in S, their favorite food is available, which incurs cost 1.

Hence, the total cost of this choice of foods is k.

Conversely, assume we have some set of foods S which incurs cost $k < \infty$. Then it must be that for each "element" child e_i, that child chooses a food corresponding to a set containing i - otherwise, the cost of our solution would be infinite. Consequently, if we choose the collection of subsets corresponding to S, we have a set cover. Moreover, this set cover uses k sets - this follows from the fact that the total cost from the "set" children must be exactly k.

Runtime: The size of our instance of No Free Lunch is polynomial in n and m. We use $n + m$ children and m foods. Furthermore, we call the black box exactly once.

No Free Lunch is in NP: The certificate is just the list of foods you should choose (linear in the input size), and the certifier algorithm simulates lunchtime - it determines what each child would eat, and then adds up the total cost of these choices. This will take $O(nm)$ time.

2 Perfect Party

This problem is NP-Complete. We will reduce from Independent Set.

Construction: We are given as input an undirected graph $G = (V, E)$ and an integer k. To transform this into an instance of Perfect Party, let $\overline{E}$ be the set of edges not in E. To solve Independent Set, solve Perfect Party on $\overline{G} = (V, \overline{E})$ with the same k, and return the result.

Correctness: Assume we have an Independent Set S in G of size k. Use the same set of vertices in $\overline{G}$. Because the vertices in S are mutually nonadjacent in G, they are mutually adjacent in $\overline{G}$. Therefore they constitute a Perfect Party solution of size k.

Conversely, assume we have a Perfect Party S of size k in $\overline{G}$. Use the same set of vertices in G. Because the vertices in S are mutually adjacent in $\overline{G}$, they are mutually nonadjacent in G. They therefore constitute an Independent Set of size k.

Runtime: Transforming G to $\overline{G}$ takes $O(m)$ time. We call the black box once. Clearly polynomial time.

Perfect Party is in NP: The certificate is the vertices in a Perfect Party solution (length $O(n)$). The certifier checks: are these vertices mutually adjacent? ($O(n^2)$ time) Are there k or fewer of them? ($O(n)$ time)

3 The Future of Trade

This problem is NP-Complete. We reduce from Subset-Sum.

Construction: An instance of Subset-Sum is a list of non-negative integers $s_1, \ldots, s_n$ and a target integer t. We will construct a market with $n+1$ items, and two users. User 1 owns items 1 through n, and user 2 owns item $n+1$. For $i \in \{1, \ldots, n\}$, both users value item i at s_i. Finally, player 1 values item $n+1$ at $t+1$, and player 2 values item $n+1$ at $t-1$. To solve Subset-Sum, run a black box for Future of Trade on this construction, and return the result.

Correctness: Assume there is a subset S of $\{s_1, \ldots, s_n\}$ adding up to t. Then user 1 and user 2 will wish to trade items in S for item $n+1$, as user 1's value for S is t and their value for $n+1$ is $t+1$, while user 2 values $n+1$ at $t-1$ and S at t.

Conversely, assume there is a trade in our constructed instance of Future of Trade. Let S be the items traded away by user 1. It must be that the total value V of either player for the items in S satisfies $t - 1 < V < t + 1$. Since the values are integral, it must be that $V = t$. Hence, taking the integers in S is a subset of $s_1, \ldots, s_n$ which adds up to t - a solution to Subset-Sum.

Runtime: Our construction has a constant number of users, and $O(n)$ items. We call the black box once.

No Free Lunch is in NP: The certificate is the set of items to be traded for each of the two users. The certifier just checks that the two users involved would agree to the trade, which requires linearly many additions.