COL866: Special Topics in Algorithms **Fall 2023**

Assignment 2

Total points: 100 Deadline: Sept 11 (Monday)

1. Given below is a list of cake-cutting subroutines. You are required to show how these subroutines can be implemented using Robertson-Webb queries. Also mention the worst-case number of queries needed by your implementation.

   Specifically, you can use the following two queries:

   - $\alpha \leftarrow \texttt{eval}_a(X)$: returns the value α that agent a assigns to the (possibly non-contiguous) piece X, and
   - $Y \leftarrow \texttt{cut}_a(X, \alpha)$: returns a piece $Y \subseteq X$ such that $v_a(Y) = \alpha$. Here X and Y need not be contiguous segments in $[0, 1]$.

   a) [**5 points**] EQUALIZE(X, Y, a): This subroutine takes as input two pieces X and Y of the cake $[0, 1]$ and an agent a (where, as a matter of convention, we assume that $v_a(X) \geqslant v_a(Y)$), and returns two pieces X' and Y' such that there is *no wastage* (i.e., $X \cup Y = X' \cup Y'$), and the returned pieces are of *equal value* according to agent a (i.e., $v_a(X') = v_a(Y')$). Additionally, the piece Y should be completely contained in Y'. That is, the equalization is done by taking something away from the more valuable piece and adding it to the less valuable piece.

   b) [**5 points**] EQ-DIVIDE(X, a, k): This subroutine takes as input a piece X of the cake $[0, 1]$, an agent a, and a positive integer k, and returns k mutually disjoint pieces $X_1, \ldots, X_k$ such that $X_1 \cup X_2 \cup \cdots \cup X_k = X$ and $v_a(X_i) = v_a(X_j)$ for all $i, j \in \{1, \ldots, k\}$. That is, the subroutine divides the piece X into k equally valued pieces according to agent a.

   c) [**5 points**] SELECT$(\{X_1, \ldots, X_\ell\}, a, k)$: This subroutine takes as input ℓ pieces $X_1, \ldots, X_\ell$ of the cake $[0, 1]$ (where ℓ is a positive integer), an agent a, and a positive integer $k \leqslant \ell$, and returns the top-k favorite pieces of agent a among $X_1, \ldots, X_\ell$. As a matter of convention, we will assume that the returned pieces are sorted in non-increasing order of values.

   d) [**5 points**] TRIM$(\{X_1, X_2\}, a)$: This subroutine takes as input two pieces X_1, X_2 of the cake $[0, 1]$ and an agent a (where, for convention, we assume $v_a(X_1) \geqslant v_a(X_2)$), and returns three pieces X_1', X_2, T such that $v_a(X_1') = v_a(X_2)$ and $X_1' = X_1 \setminus T$. That is, agent a trims the more valuable piece X_1 to make it equal in value to the less valuable piece X_2.

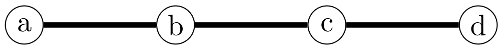
2. [**25 points**] Consider a simple undirected graph where each vertex represents an agent and the edges denote friendships. Only friends are allowed to envy each other. A cake division among the agents on this graph is envy-free if no pair of friends envy each other (though,

agents who are not friends may envy each other). Note that the model discussed in class is a special case of this setting when the graph is complete.

Design a discrete cake-cutting protocol (in the Robertson-Webb query model) for finding an envy-free division when the graph is a *path on four vertices* (see below). Prove the correctness of your protocol and also mention the number of queries made.

Hint#1: Use the subroutines from Problem 1.

Hint#2: Revisiting the Selfridge-Conway procedure may be helpful.



3. Recall the envy-cycle elimination algorithm for computing an allocation satisfying envy-freeness up to one good (EF1). A fairness notion stronger than EF1 is *envy-freeness up to any good* (EFX), which states that any pairwise envy can be eliminated by removing *any* good from the envied bundle. Formally, an allocation $A = (A_1, \ldots, A_n)$ satisfies EFX if for every pair of agents i, k and every good $g \in A_k$, we have $v_i(A_i) \geqslant v_i(A_k \setminus \{g\})$. Observe that an EFX allocation satisfies EF1.

   a) [**10 points**] Provide examples of instances with additive valuations where the round-robin and envy-cycle elimination algorithms fail to return an EFX allocation.

   b) [**10 points**] Consider the indivisible goods problem under additive valuations. Show that when agents have identical *rankings* of the goods (but not necessarily identical numerical values), an EFX allocation can be computed in polynomial time. For example, in the instance given below, both agents rank the goods as $g_1 \succ g_2 \succ g_3$ but have different numerical valuations for them. Also note that the ranking of bundles need not be identical; indeed, agent a_2 prefers the bundle $\{g_2, g_3\}$ over $\{g_1\}$, while agent a_1 has the opposite preference.

	g_1	g_2	g_3
a_1	11	7	2
a_2	8	7	5

   c) [**15 points**] Whether an EFX allocation always exists under additive valuations remains an important open problem in fair division. In view of this, one can ask whether relaxations of EFX (which, remember, is itself a relaxation of EF) always exist. Specifically, one can consider a "multiplicative" approximation of EFX defined as follows: Given any $\alpha \in (0, 1]$, an allocation $A = (A_1, \ldots, A_n)$ is said to satisfy α-EFX if for every pair of agents i, k and every good $g \in A_k$, we have $v_i(A_i) \geqslant \alpha \cdot v_i(A_k \setminus \{g\})$. Thus, 1-EFX is equivalent to exact EFX, which is stronger than, say, $\frac{1}{2}$-EFX.

   Consider a fair division problem with n agents where all valuations are *additive* and *integral* (i.e., for every agent i and every good g, $v_i(\{g\})$ is a non-negative integer). Show that a $\frac{1}{2}$-EFX allocation always exists.

   Hint#1: Use the envy-cycle elimination algorithm. Think about what happens when assigning a good g^* to the source agent i violates $\frac{1}{2}$-EFX from the perspective of some agent k. That is, there is some good $g \in A_i$ such that $v_k(A_k) < \frac{1}{2}v_k(A_i \cup \{g^*\} \setminus \{g\})$.

Hint#2: You may find it helpful to "unassign" some of the currently allocated goods and return them to the pool of unallocated goods.

4. In this problem, we will focus on a subclass of additive valuations induced by *geometric sequences* (e.g., $2^0, 2^1, 2^2, 2^3, \dots$). Suppose there are n agents with additive valuations over m goods. For any agent i, we will assume that it values its most preferred good at 2^{m-1}, next most-preferred good at 2^{m-2}, and so on, and its least-preferred good at $2^0 = 1$ (thus, every agent has strictly different valuations for all m goods). See the instance below for an example with two agents and four goods. We will call such instances *geometric additive instances.*

	g_1	g_2	g_3	g_4
a_1	8	4	2	1
a_2	4	1	8	2

a) [**5 points**] Show that for any geometric additive instance with indivisible goods, an allocation is EFX if and only if any envied bundle contains exactly one good.

b) [**10 points**] Let us denote the n agents by $a_1, \dots, a_n$ and the m goods by $g_1, \dots, g_m$. A *picking sequence* is an m-length ordered tuple $\sigma := \langle s_1, s_2, \dots, s_m \rangle$ where, for every $i \in [m]$, we have $s_i \in \{a_1, \dots, a_n\}$, and starting with s_1, agents take turns according to σ to pick their favorite remaining item. For example, consider the instance below with three agents a_1, a_2, a_3 and six goods $g_1, \dots, g_6$.

	g_1	g_2	g_3	g_4	g_5	g_6
a_1	$\underline{8}$	$\underline{4}$	2	1	32	16
a_2	4	1	2	$\underline{16}$	$\underline{8}$	32
a_3	1	2	$\underline{4}$	8	16	$\underline{32}$

Suppose $\sigma := \langle a_3, a_2, a_2, a_1, a_3, a_1 \rangle$. This means that under σ, agent a_3 goes first and picks its favorite good g_6. Then, agent a_2 picks its favorite remaining good g_4. The next turn also belongs to a_2, where it picks the good g_5, and so on.

An allocation is said to be *sequencible* if there is an m-length picking sequence of agents which results in that allocation. In the above example, the underlined allocation is sequencible since it is induced by the picking sequence σ.

Show that for any geometric additive instance with indivisible goods, an allocation is Pareto optimal (PO) if and only if it can be induced by a picking sequence.

c) [**5 points**] Design an algorithm that, given as input any geometric additive instance, returns an allocation that is EFX and PO.