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.

   ALGORITHM 1: EQUALIZE

   Input: Pieces X and Y and agent a
   Output: Pieces X' and Y'

   1 Let $x = \texttt{eval}_a(X)$ and $y = \texttt{eval}_a(Y)$.
   2 $X' \leftarrow \texttt{cut}_a(X, \frac{x+y}{2})$ ▷ `this is feasible because` $x \geqslant y$
   3 $Y' \leftarrow Y \cup (X \setminus X')$
   4 **return** X', Y'

   Number of queries: Two $\texttt{eval}(\cdot)$ queries and one $\texttt{cut}(\cdot)$ query.

   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.

ALGORITHM 2: EQ-DIVIDE

Input: Piece X, agent a, positive integer k
Output: Pieces $X_1, \ldots, X_k$

Let $x = \texttt{eval}_a(X)$.
for $i \in \{1, 2, \ldots, k-1\}$ **do**
    $X_i \leftarrow \texttt{cut}_a(X, a, x/k)$
    $X \leftarrow X \setminus X_i$
$X_k \leftarrow X$
return $X_1, \ldots, X_k$

Number of queries: One $\texttt{eval}(\cdot)$ query and $k-1$ $\texttt{cut}(\cdot)$ queries.

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.

ALGORITHM 3: SELECT

Input: Pieces $X_1, \ldots, X_\ell$, agent a, positive integer $k \leqslant \ell$
Output: Pieces $X_{i_1}, \ldots, X_{i_k}$

for $i \in \{1, 2, \ldots, \ell\}$ **do**
    Let $x_i = \texttt{eval}_a(X_i)$.
Sort $x_1, \ldots, x_\ell$ in non-increasing order (ties are broken arbitrarily) and return the pieces corresponding to the first k values.

Number of queries: ℓ $\texttt{eval}(\cdot)$ queries, no $\texttt{cut}(\cdot)$ query

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.

ALGORITHM 4: TRIM

Input: Pieces X_1, X_2 and agent a
Output: Pieces X_1', X_2, T

Let $x = \texttt{eval}_a(X_2)$. ▷ x is the value of less valuable piece.
$X_1' \leftarrow \texttt{cut}_a(X_1, a, x)$
$T \leftarrow X_1 \setminus X_1'$
return X_1', X_2, T

Number of queries: One $\texttt{eval}(\cdot)$ queries, one $\texttt{cut}(\cdot)$ query

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.

ALGORITHM 5: Envy-freeness for four agents on a path

Input: A path graph over the agents a, b, c, d and a cake $[0, 1]$
Output: An allocation A_a, A_b, A_c, A_d of the cake

▷ ----------Phase 1: Creating trimmings----------

$X_1, X_2, X_3, X_4 \leftarrow$ EQ-DIVIDE$([0, 1], b, 4)$ ▷ agent b is the "cutter" for the cake $[0, 1]$

$X_1 \leftarrow$ SELECT$(\{X_1, X_2, X_3, X_4\}, a, 1)$ ▷ agent a selects its favorite piece

$X_3, X_4 \leftarrow$ SELECT$(\{X_2, X_3, X_4\}, c, 2)$ ▷ agent c selects its top two favorite pieces among the remaining three pieces

$X'_3, X_4, T \leftarrow$ TRIM$(\{X_3, X_4\}, c)$ ▷ agent c trims the more valuable piece; thus, $v_c(X'_3) = v_c(X_4)$ and $X'_3 \leftarrow X_3 \setminus T$

▷ ----------Phase 2: Equalizing----------

$T_1, T_2, T_3, T_4 \leftarrow$ EQ-DIVIDE$(T, b, 4)$ ▷ agent b is the "cutter" for the trimmings T

$T_1 \leftarrow$ SELECT$(\{T_1, T_2, T_3, T_4\}, a, 1)$ ▷ agent a selects its favorite piece from the trimmings

$T_3, T_4 \leftarrow$ SELECT$(\{T_2, T_3, T_4\}, c, 2)$ ▷ agent c selects its top two favorite pieces among the remaining three pieces of the trimmings

$T'_3, T'_4 \leftarrow$ EQUALIZE$(\{T_3, T_4\}, c, 2)$ ▷ agent c equalizes the pieces selected in the previous step such that $T'_3 \subseteq T_3$ and $T'_4 \supseteq T_4$

▷ ----------Phase 3: Bundling and Allocation----------

$A_a \leftarrow X_1 \cup T_1$ ▷ Agent a gets its preferred pieces from Phases 1 and 2

$A_b \leftarrow X_2 \cup T_2$ ▷ Agent b (the "cutter") gets the leftovers

▷ Combine the "enriched" trimmed piece T'_4 with the "diminished" main piece X'_3

if eval$_d(X'_3 \cup T'_4) >$ eval$_d(X_4 \cup T'_3)$ **then**
  $A_d \leftarrow X'_3 \cup T'_4$ ▷ Give agent d its preferred bundle
  $A_c \leftarrow X_4 \cup T'_3$
else
  $A_d \leftarrow X_4 \cup T'_3$
  $A_c \leftarrow X'_3 \cup T'_4$

return A_a, A_b, A_c, A_d

Query count: The algorithm uses the EQ-DIVIDE subroutine twice, the SELECT subroutine four times, the TRIM subroutine once, and the EQUALIZE subroutine once, followed by two eval$(\cdot)$ queries in the final phase. By directly substituting the query bounds obtained in the above problems, we get a total of 21 eval$(\cdot)$ queries and 8 cut$(\cdot)$ queries.

Note that we can shave off some eval queries thanks to the normalization assumption. Indeed, in the first two steps in Phase 1, we can use the fact that the entire cake is valued at 1 to save two eval queries, one in the EQ-DIVIDE step and another in the SELECT step. Likewise, for the TRIM step in Phase 1 and EQUALIZE step in Phase 2, we can simply reuse the evaluations for agent c done by the preceding SELECT operations.

This further saves three `eval` queries, giving an improved bound of 16 `eval` queries overall.

Correctness: It can be easily verified that the algorithm terminates. To argue envy-freeness, let us consider the perspective of the individual agents.

- Agent a does not envy agent b because it prefers both pieces in its final allocation, namely X_1 and T_1, over the respective pieces in the bundle of agent b, namely X_2 and T_2.
- Agent b does not envy agent a because, by virtue of being the "cutter", it values the pieces X_1 and X_2 (and, likewise, the pieces T_1 and T_2) equally.

  It also does not envy agent c because of the following reason: If $A_c = X'_3 \cup T'_4$, then agent b has an "irrevocable advantage" over agent c and does not envy it even if the entirety of the trimmings T are given to agent c. Otherwise, if $A_c = X_4 \cup T'_3$, then since $T'_3 \subseteq T_3$ (and because agent b is the "cutter" in both Phases 1 and 2), agent b incurs no additional envy.
- Agent d does not envy agent c because the bundle that is less preferred by agent d is given to agent c in Phase 3.
- Finally, let us analyze the envy from agent c's perspective. Note that agent c values the pieces X'_3 and X_4 equally (and, likewise, the pieces T'_3 and T'_4). Therefore, agent c has equal value for the bundles A_c and A_d and does not envy agent d.

  Furthermore, regardless of whether $A_c = X'_3 \cup T'_4$ or $A_c = X_4 \cup T'_3$, agent c does not envy agent b because of the following reason: By the `SELECT` and `TRIM` steps in Phase 1, we have $v_c(X'_3) = v_c(X_4) \geqslant v_c(X_2)$ and by the `SELECT` and `EQUALIZE` steps in Phase 2, we have $v_c(T'_3) = v_c(T'_4) \geqslant v_c(T_4) \geqslant v_c(T_2)$. As before, envy-freeness follows by additivity.

Acknowledgement: This problem was based on Section 3.1 of Ghalme et al. (2022).

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.

Round-robin fails EFX on the following instance:

	g_1	g_2	g_3	g_4
a_1	$\underline{5}$	2	$\underline{2}$	1
a_2	5	$\underline{2}$	2	$\underline{1}$

Envy-cycle elimination will also fail to give EFX on the above instance if the goods are allocated in the order g_2, g_3, g_4, g_1 since, in that case, the output allocation is the same as under the round-robin algorithm.

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

Algorithm (sketch): At each step, ask the source agent to pick its favorite remaining good. The existence of source agent is guaranteed by resolving envy cycles.

Correctness (sketch): The EFX property is maintained while assigning the good to the source because the most recently added good in any bundle is also the least valuable good in that bundle w.r.t. any agent's preferences (due to identical rankings assumption). EFX is also maintained while resolving envy cycles even when the rankings are not identical (this fact will be useful in the next part).

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.

Algorithm (sketch): At each step, give an unallocated good, say g^*, to the source agent i as long as doing so preserves $\frac{1}{2}$-EFX. However, if $\frac{1}{2}$-EFX is violated from the perspective of, say, agent k (i.e., for some $g \in A_i$, $v_k(A_k) < \frac{1}{2}v_k(A_i \cup \{g^*\} \setminus \{g\})$), then return agent k's existing bundle A_k to the set of unallocated goods, and instead assign to it the good g^*.

Termination (sketch): To show that the algorithm terminates, we will use a *potential function* argument. This involves constructing an objective function and an upper (or, lower) bound on this function, and, in addition, showing that at each step, the objective strictly increases (or, decreases).

A natural idea might be to think of the "number of unallocated goods" as a potential. However, note that this number can decrease (when assigning the good to the source while preserving $\frac{1}{2}$-EFX) as well as increase (when returning the goods back to the unallocated pool) during the execution of algorithm. So, the size of the unallocated pool alone may not be a useful potential function.

However, note that the sum total of agents' values for their own bundles (specifically, the "utilitarian social welfare" objective $\sum_i v_i(A_i)$) weakly increases when giving a good to a source and strictly increases when returning goods back to the pool. The weak increase is due to the fact that the source agent gets an additional good while every other agent's bundle is unchanged. The strict increase is due to the fact that agent k gets a strictly better bundle, namely the good g^*, at the expense of its old bundle A_k.

Indeed, when assigning a good to the source agent, if $\frac{1}{2}$-EFX is maintained, then the said objective function clearly increases (since $|U|$ strictly decreases and sum of utilities does not decrease). Otherwise, the algorithm substitutes the bundle A_k with the good g^*. However, note that since agent i was the source to begin with, it must be that $v_k(A_k) \geqslant v_k(A_i)$. Furthermore, by violation of $\frac{1}{2}$-EFX, we have that $v_k(A_k) < \frac{1}{2}v_k(A_i \cup \{g^*\} \setminus \{g\})$ for some $g \in A_i$. By additivity, we get that $v_k(\{g^*\}) > v_k(A_k)$. Thus, agent k is strictly happier under the new allocation while every other agent is as happy as before.

Define the following objective (or *potential*) function for any allocation A:

$$\phi(A) \coloneqq m \cdot \sum_i v_i(A_i) - |U|,$$

where m is the total number of goods and U is the set of unallocated goods.

Note that if assigning a good to a source vertex does not violate $\frac{1}{2}$-EFX, then the

objective increases by at least 1 since the number of unassigned goods $|U|$ strictly decreases and the sum of agents' utilities $\sum_i v_i(A_i)$ does not decrease. Similarly, the objective increases by at least 1 also when there is a violation of $\frac{1}{2}$-EFX; in this case, although $|U|$ may increase, the sum of agents' utilities strictly increases by at least 1 (recall that valuations are integral). Since the social welfare term is scaled by a factor of m, the overall objective will nevertheless strictly increase in each step.

Note that the potential function cannot increase beyond $m \cdot \max_i \sum_i v_i(M)$ where M is the set of goods. Thus, there is a finite upper bound on the potential function. Therefore, the algorithm must terminate after a finite number of steps.

Correctness (sketch): Recall that if assigning a good g^* to the source agent creates a violation of $\frac{1}{2}$-EFX, then instead we assign the good g^* to any agent k who experiences this violation. It suffices to show that substituting agent k's old bundle A_k with the singleton bundle g^* maintains $\frac{1}{2}$-EFX.

This is easily verified for pairs of agents whose bundles are unchanged. Further, since agent k is strictly happier, there are no new $\frac{1}{2}$-EFX violations from its perspective. Therefore, any violation of $\frac{1}{2}$-EFX can only be towards agent k. However, since agent k's bundle contains only one good, no agent will envy agent k up to the removal of this good. Therefore, $\frac{1}{2}$-EFX is satisfied for any envy directed towards agent k, implying that the overall allocation is $\frac{1}{2}$-EFX.

Acknowledgement: This problem was based on Section 6 and Section A of Plaut and Roughgarden (2020).

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.

If the envied bundle is a singleton, then any envy towards it can be eliminated by removing the only good in that bundle, thus satisfying EFX.

To prove the converse, let us argue by contradiction. Suppose the allocation A is EFX and agent i envies agent k under A, but A_k is not a singleton, i.e., $|A_k| > 1$. Due to geometric valuations, the bundle A_k must contain a good, say g, such that $v_i(A_i) < v_i(\{g\})$. Without loss of generality, let g be the highest-valued good in A_k according to agent i. Then, even after removing the good of lowest value in A_k (in agent i's valuation), agent i will continue to envy the residual bundle of agent k since it still has the good g in it. This is a violation of EFX, giving us the desired contradiction.

b) [**10 points**] Let us denote the n agents by $a_1, \ldots, a_n$ and the m goods by $g_1, \ldots, g_m$. A *picking sequence* is an m-length ordered tuple $\sigma \coloneqq \langle s_1, s_2, \ldots, s_m \rangle$ where, for every $i \in [m]$, we have $s_i \in \{a_1, \ldots, a_n\}$, 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 $g_1, \ldots, g_6$.

	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 $\sigma \coloneqq \langle a_3, a_2, a_2, a_1, a_3, a_1 \rangle$. 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.

(PO $\Rightarrow$ sequencible). In any PO allocation under additive valuations (geometric or otherwise), some agent must get its favorite item (because if not, then there should be an envy-cycle of favorite items that Pareto improves the current allocation). Eliminate the corresponding item and record the name of the corresponding agent. Among the remaining items, some agent must again get its favorite item. Repeat this procedure to construct a picking sequence of agents.

(Sequencible $\Rightarrow$ PO). Suppose, for contradiction, that a sequencible allocation A is not PO. Then, there must be another allocation B that Pareto dominates it. Let σ_A be the sequence that generates A (recall that A is sequencible by assumption).

Since A and B are distinct, there must be some agent who, during its turn under the picking sequence σ_A, picks a good that it receives under A but not under B.

Let i be the *first* such agent under σ_A, and let g be the good picked by i in that turn. Note that $g \in A_i \setminus B_i$.

Under geometric valuations, any two distinct bundles must have distinct values. Since B Pareto dominates A and agent i's bundles under A and B are distinct, we have that $v_i(B_i) > v_i(A_i)$. Again, due to geometric valuations, we get that there must a good $g' \in B_i \setminus A_i$ such that $v_i(\{g'\}) > v_i(A_i)$; thus, in particular, $v_i(\{g'\}) > v_i(\{g\})$.

Consider again the picking sequence σ_A. In this sequence, all agents before agent i's first turn pick a good that they get under both A and B. Thus, the goods g and g' must both be available at the time of agent i's first turn. However, under σ_A, agent i chose to pick good g even though it has a higher value for g' as observed above. This is a contradiction. Thus, the allocation A must be Pareto optimal.

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

Run the picking sequence $a_1, a_2, \ldots, a_{n-1}, \underbrace{a_n, \ldots, a_n}_{m-n+1 \text{ times}}$.

Acknowledgement: This problem was based on Theorem 1 of Hosseini et al. (2021).

References

Ganesh Ghalme, Xin Huang, and Nidhi Rathi. Envy-Free Cake Cutting with Graph Constraints. *arXiv preprint arXiv:2205.12559*, 2022. (Cited on page 5)

Hadi Hosseini, Sujoy Sikdar, Rohit Vaish, and Lirong Xia. Fair and Efficient Allocations under Lexicographic Preferences. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 5472–5480, 2021. (Cited on page 10)

Benjamin Plaut and Tim Roughgarden. Almost Envy-Freeness with General Valuations. *SIAM Journal on Discrete Mathematics*, 34(2):1039–1068, 2020. (Cited on page 8)