Harvard CS 1360, Spring 2026
Instructor: Ariel Procaccia
Scribe: Rhea Acharya, Jakob de Raaij

Cake Cutting

Lecture 10

In the last lectures, we saw mechanisms for allocating a set of items to players. Players were assumed to act strategically, and we made use of payments to incentivize them to report their true valuations. In most cases, the goal was to optimize the total social welfare.

We now switch to a related class of problems, known as *fair division*. Again, the task is to allocate items to the players; however, now the goal is to be *fair* to each individual player. We will formalize this in the next three lectures, starting in this lecture by considering the setting of dividing a single, divisible item among players, where different players may have different preferences over the parts of the item. This is often visualized through the metaphor of *cake cutting*: Each player may value different parts of the cake (e.g., one player might like the strawberry topping while another player prefers more chocolate); the goal is to find a fair way of cutting the cake into pieces and allocating them among the people.

1 Cake Cutting

Definition 1 (Cake Cutting)**.** Cake cutting is a general model for dividing a divisible item among players with heterogeneous preferences. The *cake* is the unit interval $[0, 1]$. A *piece of cake* is a subset $X \subseteq [0, 1]$ of the cake; it is *connected* if X is an interval. An instance of *cake cutting* consists of a set of players $N = \{1, \ldots, n\}$, and for each player $i \in N$, a non-negative, continuous valuation function $V_i(X)$ for any piece of cake, with the following properties:

- **Additivity**: If X and Y are disjoint pieces of cake, then $V_i(X \cup Y) = V_i(X) + V_i(Y)$.
- **Normalization**: All players value the entire cake as 1, i.e., $V_i([0, 1]) = 1$.
- **Divisibility**: Given a connected piece of cake X and $\lambda \in [0, 1]$, there exists a connected piece of cake $X' \subseteq X$ such that $V_i(X') = \lambda V_i(X)$.

An *allocation* $A_1, ..., A_n$ is a partition of the cake into n pieces of cake, i.e., $A_i \cap A_j = \emptyset$ for any players $i \neq j$ and $\bigcup_{i \in N} A_i = [0, 1]$.

Note that different players might value different parts of the cake differently. For instance, we can imagine a cake which is chocolate for the interval $[0, 0.5)$ and vanilla for $[0.5, 1]$, and two players: one who values the chocolate part much more highly, and one who values the vanilla part much more highly.

We aim to find a fair allocation of cake to the players. There are two main fairness criteria we will consider.

Definition 2 (Proportionality)**.** An allocation $A_1, ..., A_n$ is *proportional* if for all players $i \in N$, it holds that $V_i(A_i) \geq \frac{1}{n}$. In other words, each player receives at least their fair share of $1/n$ their value for the total cake.

Definition 3 (Envy-Freeness)**.** An allocation $A_1, ..., A_n$ is *envy-free* if for all players $i, j \in N$, $V_i(A_i) \geq V_i(A_j)$. In other words, no player i prefers another player's piece of cake A_j over their own piece of cake A_i.

Let's compare these two notions of fairness to each other. Is one strictly more demanding than the other?

Theorem 1. *For two players, an allocation is proportional if and only if it is envy-free.*

Proof. If player 1 does not envy player 2, then $V_1(A_1) \geq V_1(A_2)$. Since $V_1(A_1) + V_1(A_2) = 1$, this implies that $V_1(A_1) \geq \frac{1}{2}$. Analogously, $V_2(A_2) \geq \frac{1}{2}$, so any envy-free allocation is proportional.

If an allocation is proportional, then $V_1(A_1) \geq \frac{1}{2}$. Since $V_1(A_1) + V_1(A_2) = 1$, this implies $V_1(A_1) \geq V_1(A_2)$, so player 1 does not envy player 2. Analogously, player 2 does not envy player 1, so any proportional allocation is envy-free. □

Theorem 2. *For three or more players, envy-freeness implies proportionality, but the converse is not true.*

Proof. If an allocation is envy-free, we know that for any player i, it holds that $V_i(A_i) \geq V_i(A_j)$ for any other allocated piece of cake A_j; player i receives their favorite piece out of the pieces in the allocation. Since

$$1 = \sum_{j=1}^{n} V_i(A_j) \leq \sum_{j=1}^{n} V_i(A_i) = nV_i(A_i),$$

it must be that $V_i(A_i) \geq \frac{1}{n}$. Thus, any envy-free allocation is also proportional.

For disproving the converse direction, assume that each player i values their own piece at $V_i(A_i) = 1/n$, but values the next player's piece at $V_i(A_{i+1}) = (n-1)/n$ (where A_{n+1} corresponds to A_1). This allocation is proportional, but for $n \geq 3$ it is not envy-free. □

We now have defined what a fair allocation looks like. However, does an envy-free or proportional assignment always exist? And if yes, how can we find it? We start by considering an algorithm for 2 players that some may know from having to split a sweet like a cookie or cake with their sibling:

Algorithm 1 Cut-and-Choose

1: Player 1 splits the cake into two pieces that they value equally.
2: Player 2 then chooses the piece they prefer, player 1 receives the other piece.

To illustrate this, consider the cake below, with blue being player 1. Blue splits the cake into two pieces, X and Y, that both give them equal value $1/2$. Player 2, in orange, gets the higher value of $2/3$ from piece Y, so they choose it. Player 1 thus receives piece X.

Theorem 3. *An allocation returned by the cut-and-choose algorithm is envy-free (and thus also proportional).*

Proof. Player 1 values both pieces of cake equally, so they will not envy player 2, no matter which of the two pieces they pick. Player 2 chooses the piece they value more, so they will certainly not envy the piece of player 1. □

When the valuation functions V_i of the players are complex, it may not be feasible in practice for a player to convey their entire valuation function (e.g., it would certainly be a bummer at a celebration if each guest had to announce their numerical value for each inch of cake). Thus, we assume that algorithms can do two simple queries, for which the results can be easily conveyed. We measure the complexity of a cake cutting algorithm as the number of queries it needs.

Definition 4 (Robertson-Webb Query Model)**.** In the *Robertson-Webb model* we can query the valuation functions in two ways

- An *evaluation query* for any connected piece of cake $X = [x, y]$ and player i tells us the value of player i for the piece of cake X. That is, $\text{Eval}_i(x, y) = V_i([x, y])$.
- A *cut query* for any starting point x, target value α, and player i tells us how to cut a connected piece of cake starting at x that player i values at α. That is, $\text{Cut}_i(x, \alpha) = y$ such that $V_i([x, y]) = \alpha$.

Under the Robertson-Webb model, the cut-and-choose algorithm requires two queries:

Algorithm 2 Cut-and-Choose (with queries)

1: Let $y = \text{Cut}_1(0, \frac{1}{2})$ be the point such that player 1 values $[0, y]$ and $[y, 1]$ equally as $1/2$.
2: Let $v = \text{Eval}_2(0, y)$ be the value of player 2 for the first piece.
3: If $v \geq 1/2$, player 2 is allocated the piece $[0, y]$; otherwise, they are allocated the piece $[y, 1]$. Player 1 is allocated the other piece.

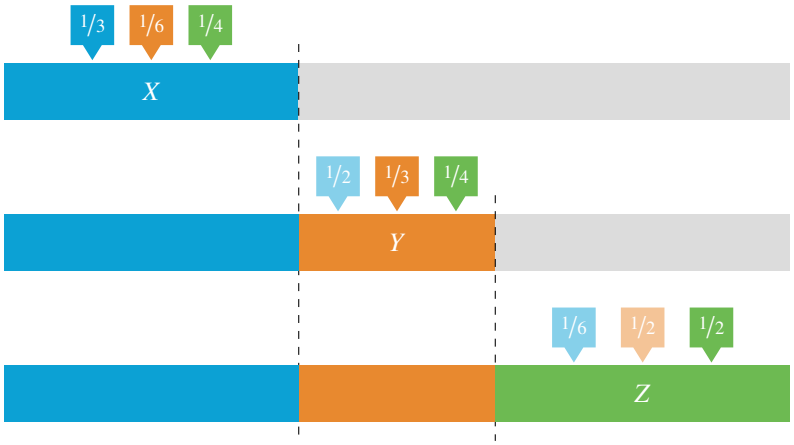
2 Optimal Proportional Algorithms

The cut-and-choose algorithm gives us envy-free and proportional allocations for 2 players. We can get proportional allocations for more than 2 players with a *moving knife* algorithm:

Algorithm 3 Dubins-Spanier

1: A referee moves a knife continuously across the cake from the left end (0) to the right end (1).
2: When any player values the segment to the left of the knife with $1/n$, they shout "stop". They receive that segment and are removed from the game.
3: The process repeats with the remaining players until one player is left, who receives the remainder of the cake.

To illustrate this, consider the cake below. The first player to shout "stop" is blue, when the knife reaches the point where piece X gives them utility $1/3$. They are the first to shout stop because the other two players don't yet get value $1/3$ from this piece. The blue player is given piece X and removed. Now, the knife continues along the cake. The next player to shout "stop" is orange, when they receive value $1/3$ from Y. Finally, the green player receives the remainder of the cake.



Theorem 4. *The Dubins-Spanier algorithm returns a proportional allocation.*

Proof. Any player who shouted "stop" at some point received a segment worth $1/n$ to them, so they received their proportional share. The last player did not shout "stop" at all, so they must have valued the previous $n-1$ pieces each with at most $1/n$. Hence, they value the remaining piece that they receive with at least $1-(n-1)1/n = 1/n$. To see that the algorithm always finishes before the knife reaches the right end of the cake, note that at any point all remaining players value the remainder of the cake with at least $1/n$, by the same argument. □

We can implement the algorithm in the Robertson-Webb query model:

Algorithm 4 Dubins-Spanier (with queries)

1: Let $R \leftarrow N$ be the set of remaining players and let $s \leftarrow 0$ be the left end of the cake.
2: **while** $|R| > 1$ **do**
3: For each player $i \in R$, use a cut query $\text{Cut}_i(s, 1/n) = y_i$ to find the connected piece $[s, y_i]$ that they value at $1/n$.
4: Let $i^* = \arg\min_{i \in N} y_i$. Player i^* is allocated the piece $[s, y_{i^*}]$ and is removed, $R \leftarrow R \backslash \{i^*\}$.
5: Let the new left end be $s = y_{i^*}$
6: **end while**
7: Allocate the remaining piece $[s, 1]$ to the remaining player in R.

The algorithm uses $n + (n-1) + \cdots + 2 = \Theta(n^2)$ cut queries.

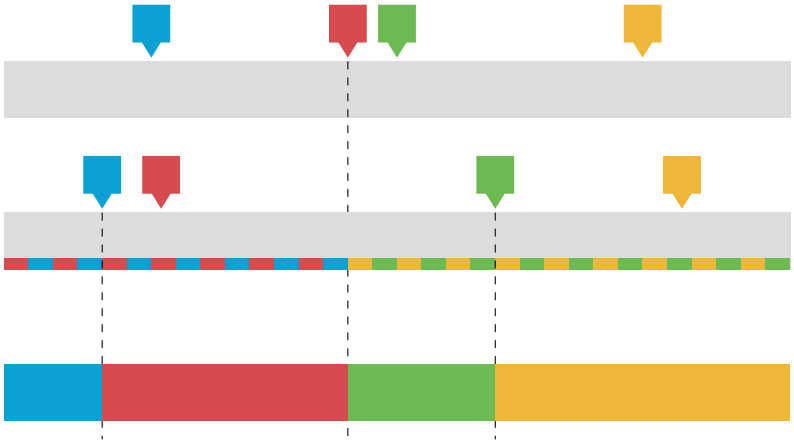
It turns out that this can be improved on with a divide-an-conquer-style algorithm

Algorithm 5 Even-Paz

▷ For ease of exposition, we describe the algorithm for $n = 2^k$ players.

Denote the interval of the cake as $[s, t]$.
if $n = 1$ **then** the single player receives $[s, t]$.
else
    For each player i, let $v_i = \text{Eval}_i(s, t)$ be their value for $[s, t]$ and let $z_i = \text{Cut}_i(s, 1/2 v_i)$ be the point that divides $[s, t]$ into two connected pieces $[s, z_i]$ and $[z_i, t]$ that player i values the same.
    Let z^* be the $n/2$th mark from the left.
    Recurse on $[s, z^*]$ with the $n/2$ players who marked $z_i \le z^*$. Recurse on $[z^*, t]$ with the remaining $n/2$ players.
end if

To illustrate this, consider the cake below with 4 players. First, each player places their marker at the point such that they derive the same value from the cake on either side of their marker. The red marker is the $4/2 = 2$nd marker, so the cake is split at this point into the left part (with the red and blue player) and the right part (with the green and yellow player). The algorithm proceeds recursively on the two parts. Each player places their marker at the point such that they derive the same value from their part of the cake either side of the marker. In these parts, the blue and the green marker are now the $2/2 = 1$st marker, along which the parts are further divided. Each of the now 4 parts of the cake is assigned a single player, so each player is allocated their current part of the cake as their piece.



Theorem 5. *The Even-Paz algorithm returns a proportional allocation.*

Proof. We prove via induction that after j recursive steps, each player values the piece of cake in their recursive call with at least $1/2^j$. As the base case, note that for $k = 0$ this is true because initially the piece is the entire cake with every player assigned to it. Every player values the complete cake at $1/2^0 = 1$.

For the induction step, we can assume that the piece of cake at step k is $[s, t]$ and any player i, of the $\frac{n}{2^k}$ players assigned to it, values it with $v_i \ge \frac{1}{2^k}$. In the next, $(k+1)$th, recursive step, any player i, of the $1/2 \cdot \frac{n}{2^k}$ players assigned to $[s, z^*]$, all with their mark $z_i \le z^*$, value their new piece of cake $[s, z^*]$, at

$$V_i([s, z^*]) \ge V_i([s, z_i]) = \frac{v_i}{2} \ge \frac{n}{2^{k+1}}.$$

Analogously, any player i, of the $1/2 \cdot \frac{n}{2^k}$ players assigned to $[z^*, t]$, all with their mark $z_i \ge z^*$, value their new piece of cake $[z^*, t]$, at

$$V_i([z^*, t]) \ge V_i([z_i, t]) = \frac{v_i}{2} \ge \frac{n}{2^{k+1}}.$$

The process ends after $\log_2 n$ steps with a single player being allocated to each piece. The value of each player for their piece is at least

$$\frac{1}{2^{\log_2 n}} = \frac{1}{n}$$

□

The algorithm has a recursive depth of $\log_2 n$. In each layer of the recursion tree, we make one evaluation query and one cut query per player. Thus, the total number of queries the algorithm needs is $2n \log_2 n = \Theta(n \log n)$. It turns out that this actually is the (asymptotically) optimal query complexity for finding proportional allocations.

Theorem 6. *Any algorithm that returns a proportional allocation to any cake cutting instance requires $\Omega(n \log n)$ queries in the Robertson-Webb model.*

3 Envy-Free Algorithms

In general, envy-free cake-cutting is algorithmically much more challenging than (the strictly less demanding) proportional cake-cutting. We discuss an envy-free cake-cutting algorithm for three players.

Algorithm 6 Selfridge-Conway

Stage 0:

1: Player 1 cuts the cake into three pieces they value equally.
2: Player 2 'trims' the most valuable piece of the three such that it is as valuable as the second most valuable piece, in their eyes.
▷ There are now four pieces: the two equally, most valuable pieces, one of which was trimmed, the piece cut of from the trimmed piece, and the remaining original piece. We refer to the trimming as cake 2 and to the three other pieces as cake 1.

Stage 1:

3: Player 3 chooses their highest-valued piece in cake 1.
4: If player 3 did not choose the trimmed piece, player 2 is allocated the trimmed piece. Otherwise, player 2 chooses their highest-valued piece out of the two remaining ones.
5: The remaining piece is allocated to player 1.
▷ At this point, either Player 2 or 3 received the trimmed piece. Among players 2 and 3, let T be the player that received the trimmed piece, and T' be the other player.

Stage 2:

6: Player T' divides Cake 2 into three pieces they value equally.
7: Player T, then player 1, then player T' choose one of the three pieces each in that order.

Theorem 7. *The Selfridge-Conway algorithm returns an envy-free allocation.*

Proof. Since valuations are additive, we know that if a player is envy-free on both cake 1 and cake 2, we know that they are envy-free on the entire cake. We will do this for each player.

- *Player 1 on cake 1.* Recall that player 1 divided the cake into three pieces of equal value to them in stage 0. They did not receive the trimmed piece, so they received a piece worth $1/3$ to them from cake 1, while no other piece of cake 1 is worth more than that to them; they are envy-free.
- *Player* 2 *on cake 1.* Recall that player 2 trimmed the two largest pieces such that they were equally valuable to them in stage 0. When they picked their piece in stage 1, at least one of their two equally, highest valued pieces of cake 1 was still available. Thus, they are envy-free, since they received their highest-valued piece of cake 1 overall.
- *Player* 3 *on cake 1.* Player 3 got to choose first on cake 1, so they received their highest-valued piece out of all 3 original pieces; they are envy-free.
- *Player T on cake 2.* On cake 2, player T gets to chose first, so they receive their highest-valued piece and are envy-free.
- *Player T' on cake 2.* They are envy-free on cake 2, since they divided it into three pieces they value equally and received one of them.

- *Player 1 on cake 2.* Player 1 does not envy player T' because they choose before player T'. Thus, the only concern left is player T, for which we will argue combining both cakes: Player T received the trimmed piece from cake 1 and some part of cake 2, the trimming, so their piece of both cakes combined is a subset of one of the three original pieces cut by player 1. Since each of these original pieces was worth $1/3$ to player 1, we know that they assign a value of at most $1/3$ to the piece of cake of player T. However, player 1 received one of the original three pieces (since they didn't get the trimmed piece), so they will assign a value of at least $1/3$ to their piece of the cake. Thus, they do not envy T. □

After this algorithm for three players was proposed independently by John L. Selfridge and John H. Conway in 1960, it took until 1995 to find any algorithm for more than three players.

Theorem 8 (Brams and Taylor, 1995)**.** *There exists an envy-free cake-cutting algorithm for any number of players under the Robertson-Webb model.*

However, the algorithm by Steven J. Brams and Alan D. Taylor takes an unbounded number of queries: The number of queries needed is not bounded above by any function of n. It was conjectured that no bounded envy-free cake-cutting algorithm for more than three players exists, until Haris Aziz and Simon Mackenzie disproved this in 2016.

Theorem 9 (Aziz and Mackenzie, 2016)**.** *There exists an envy-free algorithm for cake-cutting for any number n of players using $O\left(n^{n^{n^{n^n}}}\right)$ queries in the Robertson-Webb model.*

This upper-bound on the number of queries is huge. Interestingly, it is far away from the best-known lower bound:

Theorem 10 (Procaccia, 2009)**.** *Any envy-free cake-cutting algorithm requires $\Omega(n^2)$ operations in the Robertson-Webb model.*