Cake Cutting[*]

Suresh Venkatasubramanian[†]

November 20, 2013

By a cake is meant a compact convex set in some Euclidean space. I shall take the space to be $\mathbb{R}$, so that the cake is simply a compact interval I, which without loss of generality I shall take to be [0,1]. If you find this thought unappetizing, by all means think of a three-dimensional cake.
– Woodall[12]

In the 2nd world war, if you're a collection of Polish mathematicians sitting out the war in Lvov, your thoughts naturally turn to ... cake. Or more precisely, how to cut a cake.

This is the beginning of what we now call the problem of "fair division". Fair division algorithms are examples of what we might now call a *mechanism*, much like an auction. These are protocols designed to allocate resources among competing entities so that they all feel that they've received a fair share. This resource could be an entire city[1], or the shared property of a couple during a divorce settlement[2]. But for now we will follow our Polish mathematicians and think of a more pleasant resource: cake.

How do you divide a cake between two people so that each of them is convinced that they've received at least half the cake ?

[*]Almost all of this lecture is drawn from [8], with a few nods towards Wikipedia. The only original contribution here is the sequencing of the material, and the cake itself.

[†]This work is licensed under a Creative Commons Attributions-ShareAlike 3.0 Unported License

[1]Berlin, during the Potsdam conferences

[2]Steven Brams, one of the researchers involved in fair division, consulted for divorce lawyers.

I like what I like, and the "No Crumbs" principle. We will assume that each player has their own idea of what parts of the cake are valuable. We'll also assume that the cake can be cut without generating crumbs[3]:

Formally, we assume that each player has a finitely additive valuation $\mu_i : [0,1] \to \mathbb{R}$. Then the goal of k-player fair division is to partition the interval into k parts $S_1, S_2, \ldots S_k$ (each of which may be disconnected) so that for all $i = 1, \ldots k$, $\mu_i(S_i) \geq \frac{1}{k}$.

1 Fair Division

I cut, you choose. There is an algorithm to divide cake between two people fairly, and its origins are lost in the mists of time (2800 years or so back, when people were still eating cake). The algorithm in its totality is:

I cut, you choose.

Why does this work ? I cut the cake, so according to my viewpoint, the pieces are equal, and I'm happy with either of them. Now you get to choose your piece, and at least one of the two must be at least half the cake, so you're happy as well. Note the subtle asymmetry: the cutter is guaranteed to get half the cake, and the chooser is guaranteed to get *at least* half the cake.

But what if I mess up and don't cut the pieces evenly ? Then there's a chance that I get a smaller piece than I'd like, but that's only my fault. The chooser doesn't suffer at all. This illustrates an important aspect of cake cutting algorithms, which can be described using the legal term *caveat cuttor*[4]. If any player makes a mistake cutting the cake, it only affects them, and no one else.

This is reminiscient of the notion of a *truthful* auction[6]. Indeed, in both settings, the mechanism is designed so that the optimal strategy for any player is to reveal their true preferences.

Disagreement is good. Steinhaus pointed out that fair division is an unusual setting where disagreements actually make the division *easier*, instead of harder. This is not terribly surprising: after all, if I like cake and you like icing, I can take all the cake and you can take all the icing, and we're both convinced we got the entire value of the cake. What's neat is that in general, you can *prove* that the mere existence of a disagreement guarantees a fair

[3]When demoing this in class, use a cheesecake and a very sharp knife.
[4]not really.

division in which each player think they receive *strictly more* than their fair share of the cake.

Let's see how this works for two players. Let's say that I divided up the cake into two parts that I believe to be equal. You disagree: you think one part has 60% (which you take). Now, appealing to your better nature, I suggest that you might donate some of your "excess" back to me. Let's say I convince you to give me 5%, leaving you with 55% of the cake.

Now you can't just cut a sliver and give it to me - I might not have assigned any value to it at all. But here's a trick. You divide your piece into 12 equal parts. Since you believe the piece is worth 60% of the cake, you believe that each piece is worth 5%, and you're willing to give one of them to me. Now I believe that the piece was worth 50% of the cake, so *there is at least one piece that I think is worth* 50%/12.

I take that piece, giving me a net share of $\frac{1}{2} + \frac{1}{24}$, and you retain 55% of the cake by your estimate. And lo and behold, we both have more than half the cake[5].

The moral of the story, and a recurring theme in cake cutting, is that disagreements over value make the problem easier, not difficult.

Moving Knifes. The setting with three players gets a little trickier. You could imagine that the first player cuts up the cake into three equal pieces, but then who chooses next ? The other two players might both think that a single piece is their best choice, and now we have a disagreement.

An approach proposed by Dubins and Spanier [2] (who in turn reference Banach and Knaster[5]) uses a "moving knife". Imagine a referee that moves a knife from left to right across the cake. Each player is told to shout "STOP" when they feel that the portion to the left of the knife is a fair share. When the first player calls "STOP", they are given that portion, and the process continues.

Why does this work ? The first player to call a halt is clearly satisfied with the piece they get. In contrast, the other two players are sure that what's remaining is at least 2/3 of the cake, since they would have otherwise called. This can now be divided into two parts, and by an argument similar to cut-and-choose, they are both satisfied.

Counting Cuts. This method generalizes to any number of players. In fact it's the first algorithm we've encountered that works for any number of players. So why bother with any other algorithm ?

If you think about it, running the moving knife algorithm requires players to make an infinite (indeed, an uncountable) number of decisions, since at each position of the knife, each player has to decide whether to call "STOP" or not. If we wish to minimize the number of cuts made in the algorithm, (which essentially boils down to the number of decisions), we need a way to count cuts, and so the moving knife algorithm (and others

[5]And all our children are above average...

like it) aren't really admissible. They do have great power though, and we'll return to this point later.

Trimmings. Here's a different algorithm due to Banach and Knaster called the "last diminisher" method that divides a cake fairly among k players with a finite number of cuts. It works in k rounds, with each round ending when one player gets a piece. Assume the players are ordered arbitrarily from 1 to k.

Algorithm 1 A single round of trimming

```
Player 1 cuts a piece of size $1/k$.
for $i = 2$ to $k - 1$ do
    Player $i$ trims the current piece if they think it's too big. They then pass it on.
end for
Player $k$ can either choose to take the piece or leave it. If they leave it, then it
goes back to the last player who trims it, or Player 1 if there was no trimming.
```

You should be able to convince yourself that whoever gets the piece is satisfied with it. Either Player k takes the piece and therefore must be satisfied, or someone who trimmed it down to size (and is therefore happy with it) gets it. Moreover, each remaining player had a chance to scrutinize the piece and find it wanting, and so all the remaining players are convinced that at least a $\frac{k-1}{k}$ fraction of the cake remains.

Successive Pairs. We'll look at one final algorithm due to Saaty [9] for a fair division among k players. This algorithm requires less discussion among the players, but ends up fragmenting the cake much more[6]. It lends itself to an easy recursive formulation and proof of correctness.

Algorithm 2 Successive Pairs

```
Recursively divide the cake among $k - 1$ players.
Each of the $k - 1$ players divides their piece into $k$ equal parts.
Player $k$ chooses a part from each of the other $k - 1$ players.
```

[6]This is an occupational hazard in cake cutting. Describing a different algorithm, Stromquist[10, page 641] says, "A player who hopes only for a modest interval of cake may be presented instead with a countable union of crumb".

The proof of correctness follows inductively by an argument very similar to that used for cut-and-choose (**How?**).

2 Envy-free Division

Fair division is all well and good, but envy is the machine that drives the modern capitalist enterprise[7]. Suppose it's no longer good enough that I get a fair share. Suppose I want to have a larger share than everyone else ? Such a division is called an *envy-free* division of the resource.

Formally, if fair division is expressed as $\mu_i(S_i) \geq 1/k$, envy-free division is expressed as $\mu_i(S_i) \geq \mu_i(S_j) \forall i, j$.

The cut-and-choose algorithm is envy-free, because the cutter is indifferent to the choice of pieces, and the chooser is free to choose the best piece. But it's much harder to design an envy-free algorithm for three or more players. Consider the moving knife algorithm. The player who chooses last is clearly envy-free since they've held out that long. But what about the first player ? It's quite possible that while they got their fair share, they realize that another player got a larger piece (under their valuation) from what's left.

The first algorithm for constructing an envy-free division among three people was discovered independently by Selfridge and Conway[8]. It's a rather complicated method that we'll build up to gradually, and it introduces an interesting graph theoretic viewpoint on the problem.

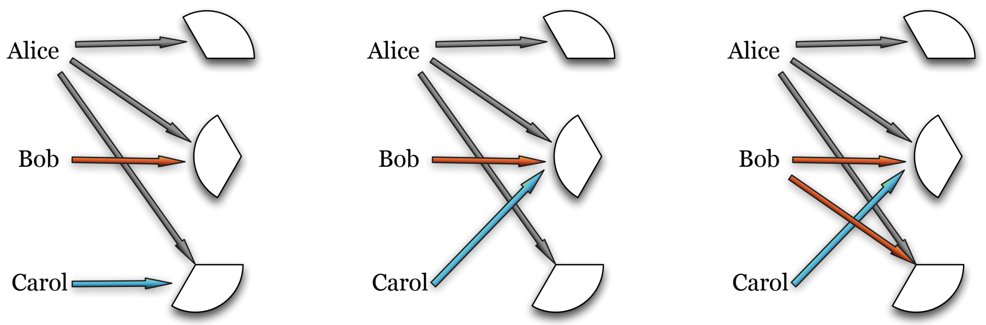
Let's say our three players are Alice, Bob and Carol. We start by asking Alice to divide the cake into three equal pieces. She is now indifferent to the pieces chosen by the other two and is therefore envy-free. If Bob and Carol each view different pieces as the best under their valuation, we are done. If we draw a bipartite graph with the left side representing players and the right side representing pieces, with an edge between a player and piece if that player will take that piece without envy, then the above situation looks like Figure 1(a). What's really happening is that the graph admits a perfect matching, and so everyone gets the piece they need.

But suppose Bob and Carol feel that the same piece is the best under their own valuations. The resulting graph looks like Figure 1(b) and now we have a problem: this graph doesn't admit a perfect matching[8].

Suppose we could somehow get Bob to value *two* pieces equally highly. Then we have a graph that looks like Figure 1(c) and now we have a perfect matching once again. Algorithmically, Alice divides the pieces, and Carol picks her favorite piece. There's always some piece left for Bob to pick without envy, and as noted before, Alice can pick any piece.

[7]...or something like that...

[8]Once again, disagreement makes things easier, and agreement makes it harder.



(a) Initial preferences (b) A conflict – no envy-free assignment (c) Fixing the conflict

Figure 1: Underlying graph structure for envy-free cutting

How do we create this scenario ? Simple - we force it. After Alice makes the cuts, we ask Bob to pick his top two pieces – let's call these S_1, S_2 – and then *trim* the largest piece S_1 to make it equal in size to S_2. We call this trimmed piece S_1' and the remaining trimmings T. If we now consider the pieces S_1', S_2, S_3, the conditions for a perfect matching apply, and we can allocate the pieces in an envy-free manner. A minor detail here is that Bob must pick either S_1' or S_2, which can easily be arranged.

But what about T ? We could recurse, but this process could continue forever (remember that cake is continuous). We need a different approach to dividing T.

We know something about T. It was taken from S_1, which Alice believes to be equal to the piece she got. So *if the person who got S_1' gets any part of T, Alice doesn't care*. Let's say that Bob got the trimmed piece. Then we have Carol divide T into three equal parts. Bob picks first, followed by Alice. Bob is envy-free by virtue of picking first. Alice is envy free because she picks *before* Carol. Finally, Carol is envy-free because the pieces are equal in value by her valuation.

The composition of two envy-free divisions is envy-free (**why?**), and thus we have an envy-free division. The overall algorithm is summarized below.

Generalizing this to an arbitrary number of players is extremely hard, and was done by Brams and Taylor in 1995[1].

3 Minimizing the number of cuts

We'd like to minimize the number of cuts used when dividing a cake. After all, in these algorithms, the cut is a resource, and our business is the minimization of resources ! It's

Algorithm 3 Envy-free division among 3 people.

```
Alice divides the cake into three equal parts $S_1, S_2, S_3$
Bob picks the largest two pieces (say $S_1 \geq S_2$), and *trims* $S_1 = S'_1 \cup T$ so that $S'_1$
and $S_2$ are equal ($T$ can be empty)
Carol picks the largest among $S'_1, S_2$ and $S_3$, followed by Bob (who must pick
$S'_1$ if it's available) followed by Alice.
Suppose Bob picks $S'_1$. Then Carol divides $T$ into three equal parts (flip the two
players if not). These are picked in order by Bob, Alice, and finally Carol.
```

important to make sure a "cut" is well-defined – we've already seen that with the moving knife, we're making an infinite number of decisions that are essentially equivalent to cuts.

We will assume that anyone can divide a cake in the ratio $a : b$ for any a, b according to their valuation. We will also assume that no one is expected to make cuts based on someone else's valuation.

Under these assumptions, it is possible to make precise statements about the number of cuts needed to divide a cake. First, let's consider some upper bounds. In the trimming algorithm, each round ends with one satisfied player after k cuts (or decisions not to cut). Thus, the total number of cuts is given by $T(k) = T(k-1) + k$, and so the overall number of cuts is $T(k) = O(k^2)$. In the successive cuts algorithm, each of the $k-1$ players makes $k-1$ cuts before presenting a choice to player k. Thus the recurrence here is $T(k) = T(k-1) + (k-1)^2$, which yields $T(k) = O(k^3)$.

The expression $O(k^2)$ uses something called "Big-Oh" notation. Informally, it's a convenient way to say that something grows at a certain rate, without worrying about specific details. For example, an expression $O(k^2)$ could mean $10k^2$, or $0.0001k^2$ or could even mean $5k$! But all we care about is that the expression grows *no faster than* k^2. Big-Oh notation is a fundamental concept in mathematics and computer science, and you can read more about it on the web[11].

Can we do better than the trimming algorithm ? It turns out that our old favorite, divide-and-conquer, comes to our rescue in a very clever way.

Let's assume we have $2k$ players. Ask the first $2k-1$ players to make marks on the cake where they think the halfway point is. Consider the median mark among these, and ask the $2k^{\text{th}}$ player to choose which side of the median mark they'd prefer. The key observation here is that no matter which side player $2k$ picks, *there are $k-1$ players that agree with them, and k players who'd prefer the other side*. This observation yields a divide-and-conquer algorithm by recursing on the two sides. The running time of this approach is $T(2k) = 2T(k) + 2k - 1$, which yields $T(k) = O(k \log k)$.

Equations like $T(k) = T(k-1) + k$ and $T(2k) = 2T(k) + 2k - 1$ are called *recurrence relations*: they define the unknown function $T(k)$ in terms of itself recursively. One of the things we study in discrete mathematics is how to solve recurrence relations of this kind. If you'd like to know more, this lecture by Jeff Erickson[4] is quite excellent.

4 Unequal Shares

Dividing things equally is all well and good. But suppose we need to make an uneven split. Maybe you came home from classes and found your roommate eating the cake that you bought the night before. He offers to split it equally, but you're nobody's fool. You can see that he's eaten about a third of it, and you demand that the remaining cake be split in a $3:1$ ratio to even things out.

Due to advanced cloning technology, there's a relatively easy way to solve the problem for any ratio of integers $p:q$. You clone yourself p times, and your roommate clones himself q times. The $p+q$ clones now perform a fair division using any of the methods described above, and then they collapse back into single individuals with the right amount of cake.

But this approach is very inefficient. After all, the goal is to find a single cut that divides the cake into two pieces, and yet we use (say using the most efficient divide-and-conquer strategy) $O((p+q)\log(p+q))$ cuts.

In fact, this algorithm does not even use a polynomial number of cuts. The numbers p, q can be expressed using $N = \log p + \log q$ bits, and yet the algorithm needs roughly e^N cuts !

There's a much more involved method that uses very elegant combinatorics. But let's motivate it with an example. Suppose we have players Alice and Bob who wish to divide a piece of cake fairly in the ratio $8:5$. Here's how the protocol works. We ask Bob to make a cut where he thinks a fair $8:5$ division would be, and ask Alice two questions:

Q1: Is the larger piece *at least* $8/13$ of the cake ?

Q2: Is the smaller piece *at most* $5/13$ of the cake ?

Suppose Alice says YES to Q1. Then she can take that piece, Bob takes the other piece, and they are both satisfied. Similarly, if Alice says YES to Q2, she can give that piece to Bob, and take the rest. But suppose she says NO to Q1 and NO to Q2 ? She doesn't want to take the first piece, and will certainly not want Bob to get the second one.

But Alice agrees that the smaller piece has at least $5/13$ of the total. So she takes it ! Now the remaining cake should be divided in the ratio $3:5$. This time, Alice makes the cut, and the process repeats (with the players reversed). Writing out the process, we get

$$\begin{aligned}
8 &= 5 \cdot 1 + 3 \\
5 &= 3 \cdot 1 + 2 \\
3 &= 2 \cdot 1 + 1 \\
2 &= 1 \cdot 1 + 1
\end{aligned}$$

and then we are left with two pieces of the same size.

This algorithm is called the GCD method, and generates the sequence of pieces $5, 3, 2, 1, 1, 1$. It turns out that this sequence is a special case of a class of partitions of 13 called *Ramsey partitions.*

Definition 4.1. *A partition of $p + q$ is called a* Ramsey partition *with respect to p, q if for any subset of the elements in the partition either they sum up to at least p, or their complement sums up to at least q.*

For example, consider the partition $5, 3, 2, 1, 1, 1$ of 13 with respect to $8, 5$. Let's pick the subset (marked in bold) $\underline{\mathbf{5}}, 3, 2, \underline{\mathbf{1}}, 1, \underline{\mathbf{1}}$. The marked set sums to 7, which is less than 8. But among the unmarked elements we can pick $3 + 2 = 5$. You can verify that this holds no matter which subset you pick, and so the partition is a Ramsey partition with respect to $8 + 5$.

How does this help ? If we now desire to divide a cake in a $p : q$ ratio and have a Ramsey partition $r_1, r_2, \ldots, r_k$ of $p + q$ with respect to $p : q$, we ask one player to divide the cake into pieces of size $r_1, r_2, \ldots, r_k$. Then we ask the other player to pick any pieces they feel have been cut correctly according to their valuation.

There are two possibilities. Either the second player marks as acceptable enough pieces to obtain her share. If not, the Ramsey partition property implies that among the other pieces, there are enough pieces for the first player to take their share. The number of cuts needed is one less than the size of the Ramsey partition, and can be shown to be related to the number of steps in the GCD computation for p and q.

5 Notes.

This lecture does not cover a number of interesting topics relating to cake cutting algorithms, such as

- general procedures for envy-free cuttings
- what to do if items are indivisible
- chore allocation (each player wants to receive as *little* as possible)
- nearly exact procedures (allowing for approximate sharing)
- auction mechanisms for fair division.

References

[1] Steven J Brams and Alan D Taylor. An envy-free cake division protocol. *The American Mathematical Monthly*, 102(1):9–18, 1995.

[2] L.E. Dubins and E.H. Spanier. How to cut a cake fairly. *The American Mathematical Monthly*, 68(1):1–17, 1961.

[3] Jeff Edmonds and Kirk Pruhs. Cake cutting really is not a piece of cake. In *In Proceedings of the Seventeenth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA*, pages 271–278, 2006.

[4] Jeff Erickson. Solving recurrences. `http://www.cs.uiuc.edu/~jeffe/teaching/algorithms/notes/99-recurrences.pdf`, 2011.

[5] T.P. Hill. Counterexamples in cake-cutting. *arXiv preprint arXiv:0807.2277*, 2008.

[6] Noam Nisan, Tim Roughgarden, Eva Tardos, and Vijay V Vazirani. *Algorithmic game theory*. Cambridge University Press, 2007.

[7] Ariel D Procaccia. Thou shalt covet thy neighbor's cake. *Proc. of 21st IJCAI*, pages 239–244, 2009.

[8] J. Robertson and W. Webb. *Cake-cutting algorithms: Be fair if you can*. AK Peters, 1998.

[9] Thomas L Saaty. *Optimization in Integers and Related Extremal Problems: By Thomas L. Saaty*. McGraw-Hill, 1970.

[10] W. Stromquist. How to cut a cake fairly. *The American Mathematical Monthly*, 87(8):640–644, 1980.

[11] Wikipedia. Big-O notation. `http://en.wikipedia.org/wiki/Big_O_notation`.

[12] D.R Woodall. Dividing a cake fairly. *Journal of Mathematical Analysis and Applications*, 78(1):233 – 247, 1980.