Change-Making with Partition Theory

Noah Bergam (njbergam@gmail.com) Columbia University

Abstract How many ways are there to change a dollar into coins? How quickly can we figure it out? These questions and their immediate generalizations take us on an fascinating tour of elementary combinatorics. In this article, we analyze the change-making problem using generating functions, and we use our solution to motivate the theory of counting integer partitions (where a partition is essentially an arrangement of n objects into k piles). In the process, we find a new solution to a 2003 Putnam problem, and we analyze two proofs of Euler's Identity for partitions, one of which is based solely on generating functions.

Change-making Change is awkward. It's uncomfortable but necessary. If you traverse the self-help section of your local bookstore, you'll find lot of motivational material written about *change*, in the sense of changing one's life goals or habits. But what about changing \$43.21 into pennies, nickels, and dimes—or, more broadly, changing an arbitrary number of cents into an arbitrary coinage system? This sort of change can seem banal and peripheral (even from a forgiving mathematical perspective), but when you really dig into the question from a combinatorial point of view, you quite naturally motivate some powerful mathematical abstractions: the two critical ones in this exploration being *generating functions* and *integer partitions*.

Let us formulate more rigorously the problem at hand. Say you have a **principal** of $k \in \mathbb{N}$ cents, and a system of **coinage** encoded as $w = (w_1, w_2, ..., w_n)$, where each $w_i \in \mathbb{N}$ denotes the value of the coin in cents. A valid **change of k** is an tuple $x \in \mathbb{Z}^n$ with $x_i \geq 0$ such that $\sum_{i=1}^{n} x_i w_i = k$. Let $C(k; w)$ be the number of distinct, valid changes of k. Observe that if we want to ensure that $C(k) > 0$, we ought to ensure that a penny exists, i.e. $w_1 = 1$.

We want an algorithm for computing $C(k)$. Of course, this counting problem does not encompass all the inquiries one might have about this setup. Related questions, which might also be included under the broader umbrella of so-called change-making problems, include the following.

1. What is the minimal number of coins necessary to express a given principal? In other words, given k and w, find a valid change x which minimizes $\sum_{i=1}^{n} x_i$. You may recognize this as a special case of the **integer knapsack problem**.
2. What series of coinage minimizes the necessary number of coins, over some set of principals K? If $x^{w,k}$ denotes the minimal change for a given coinage and principal (i.e. the solution to integer knapsack problem), then can formulate this **optimal denomination problem** as follows:

$$\arg\min_{w \in \mathbb{N}^d} \mathbb{E}_{k \in K} \Big[\sum_{i=1}^{n} x_i^{w,k} \Big]$$

   Observe that, with the above formulation, we run into convergence issues for K infinite. (Shallot 2002) discusses this problem for K consisting of principals less than $k = 10000$ (i.e. \$100), and finds that $w = (1, 5, 18, 25)$ and $(1, 5, 18, 29)$ are the two minimizing coinages.

The reader is encouraged to consider other potential optimization problems. For now, though, let us return to our own investigation, which we will dub the **change-**

making problem. From a computational point of view, there are two immediate solutions one might suggest.

1. **Naive Recursion**: Given some principal k, subtract each each coin denomination, yielding n smaller principal values, denoted $k_{1i} = k - w_i$. If $k_{1i} < 0$, disregard that solution. If $k_{1i} = 0$, we found a valid solution. If $k_{1i} > 0$, repeat the process. Once all k_{ji} have terminated, count your valid solutions. As you might sense, this runtime of this solution is roughly exponential in k, since you are traversing a tree with a depth that grows linearly in k.
2. **Dynamic Programming**: This solution involves building up solutions from 0 to $n - 1$ before solving for n. Note that, as with many DP problems, you can formulate this in a top-down approach, where you essentially run the naive recursion with strategic memoization (so as not to repeat certain paths through the list). In any case, this reduces the runtime significantly compared to naive recusion. If k is the principal, and m is the number of coin, the runtime of this DP approach is $O(km)$.

In the first section of this writeup, we work towards a remarkable $O(1)$ solution to the change-making problem in the context of good-old American coinage $(1, 5, 10, 25, 50, 100)$. This solution, famously shown in Graham, Knuth, and Patashnik's 1988 textbook *Concrete Mathematics*, employs **generating functions** in very clever yet intuitive ways. We will carefully work up to such intuition.

The Intuition of Generating Functions Recall the grade-school exercise of multiplying binomials, popularly known as FOIL-ing (first, outside, inside, last).

$$(1 + x)(1 + x) = 1 + x + x + x^2 = 1x^0 + 2x^1 + 1x^2$$

For our purposes, the motivating observation of generating functions is that this multiplication of polynomials can encode useful combinatorial information. A physical example could be helpful. Let x denote the event of taking one step forward, and 1 denote the event of staying still (zero steps). Now frame the polynomial as a choice: "$($stay OR step$)$ AND $($stay OR step$)$". After this choice, there are four combinations of choices and three distinct outcomes, which are fully reflected in the final polynomial answer.

1. 1 way to take 0 steps $(1x^0)$
2. 2 ways to take 1 step $(2x)$
3. 1 way to take 2 steps $(1x^2)$

This might seem like quite a pedantic perspective on simple mathematics, but it's worth observing that the operations are encoding useful information: $+$ behaves like "or" while $*$ is like "and." This framework extends gracefully when we consider, say, the binomial theorem: $(1 + x)^n = \sum_{k=1}^{n} \binom{n}{k} x^k$. This tells us that after n rounds of binary choices between 1 and x, there are $\binom{n}{k}$ ways to have chosen x k times. That is effectively the definition of the choose function.

Remark. We can find a rather practical application of this thought process in genetics, and specifically the Hardy-Weinberg Principle[1]. Let $x_1, x_2, ..., x_n$ denote the proportion of n distinct alleles in a population—for instance, alleles expressing brown, blue,

[1] Yes, the same G.H. Hardy (1877-1947) who comes up later in this article. In light of his writings in *A Mathematician's Apology*, it is a bit ironic that he is perhaps best remembered for his singular contribution to applied mathematics.

and green eye color, respectively. For organisms with k copies of each chromosome[2], the Hardy-Weinberg equilibrium of genotypic frequencies is given by a multinomial expansion:

$$(x_1 + x_2 + ... + x_n)^k = \sum_{k_1+...+k_n=k} \binom{k}{k_1, ..., k_n} x_1^{k_1} \cdots x_n^{k_n}$$

We can interpret this equation according to familiar intuition. When an organism is born, its genetic makeup is determined as follows: in order to determine the alleles assigned at each chromosome copy, we choose from the pool of genotypes $\{x_1, ..., x_n\}$ k times. The multinomial factor encodes how many times we can make that choice. For instance, there is only 1 way for an organism to have k copies of the x_1 allele: namely, if we choose x_1 from each genetic bucket.

Now that we better understand the enumerative powers of polynomial multiplication, we will formalize the concept of generating functions. It is worth noting that in the following definition, we generalize from polynomials to potentially infinite series.

Definition (Generating Function). The (ordinary) generating function associated with a sequence $(a_n)_{n\in\mathbb{N}}$ is given by:

$$A(x) = \sum_{k=0}^{\infty} a_k x^k$$

This can seem like a pretty arbitrary thing to do some innocent sequence of numbers. The best way to think about it is as an *encoding* which has nice algebraic properties for multiplication and addition. Furthermore, the information in this encoding is easily recoverable: to get the k-th element of the sequence, take the k-th derivative, divide by $k!$, and evaluate at 0. This may bring to mind two familiar applications of generating functions:

- The **moment generating function** of a probability distribution encodes the sequence of its moments (e.g. expectation, variance, skew).

$$M_X(t) = \mathbb{E}[e^{tX}] = \sum_{k=0}^{\infty} \frac{\mathbb{E}[X^k]}{k!} x^k$$

- The **Taylor Series** associated with a smooth function $f : I \mapsto \mathbb{R}$ encodes the sequence of its derivatives at some point $a \in I$.

$$P_a f(x) = \sum_{k=0}^{\infty} \frac{f^k(a)}{k!}(x)(x-a)^k$$

Generating Functions and Coins Let p_k be the number of ways to construct a principal of k cents using only pennies. Clearly, given any number of cents, there is only one way to construct it with pennies (i.e. assembling k pennies), so the sequence is $(p_n)_{n\in\mathbb{N}} = (1, 1, 1, ...)$ and the generating function is as follows. Note that we can

[2]For humans and other diploid organisms $k = 2$.

conveniently simplify it into a closed-form expression using a basic identity about geometric series.

$$P(x) = 1 + x + x^2 + ... = \frac{1}{1-x}$$

Remark. Real analysis class may condition you to frown at this use of the geometric series expansion without specifying that $|x| < 1$. The truth is that we don't actually need convergence. The variable is perfectly fine being indeterminate: the important fact behind the scenes here is really the *uniqueness* of power series expansions.

Similarly, let n_k be the number of ways to construct k cents using only nickels. The logic is similar to the above, except $n_k = 0$ if k is not divisible by 5 (for instance, there is no way to construct 8 cents using only nickels). This gives us a sequence $(1, 0, 0, 0, 1, 0, ...)$ and the following generating function.

$$N(x) = 1 + x^5 + x^{10} + ... = \frac{1}{1-x^5}$$

Observe that if I have a bag of 2 nickels and 1 penny, then I can encode the total number of change combinations as:

$$(1 + x^5 + x^{10})(1 + x) = 1 + x^5 + x^{10} + x + x^6 + x^{11}$$

In other words, there are six possible outcomes: 0 cents (pick nothing), 5 cents (1 nickel), 10 cents (2 nickels), 1 cent (1 penny), 6 cents (1 nickel, 1 penny), or 11 cents (2 nickels, 1 penny).

We can swiftly generalize this intuition. Consider that for each coin of value w_i, the generating function is $\frac{1}{1-w_i}$. So the generating function for the combinations of all the different coin types is given by their product:

$$A(x) = \frac{1}{(1-x)(1-x^5)(1-x^{10})(1-x^{25})(1-x^{50})(1-x^{100})}$$

Finding the answer to our question for some number of cents k simply comes down to finding the coefficient of x^k in the polynomial version of $A(x)$, which would be the Maclaurin series (Taylor series centered at zero). Done naively, this is roughly an $O(mk)$ operation, as we have to take the k-th derivative of a function which has m terms. What can seem miraculous, though, is that we can do better.

Constant-Time Solution Since almost all the coin values are divisible by 5, we can factor A in a rather convenient way. First, multiply both the top and bottom by $(1 + x^5)$.

$$A(x) = \frac{(1-x^5)/(1-x)}{(1-x^5)^2(1-x^{10})(1-x^{25})(1-x^{50})(1-x^{100})}$$

Observe that the denominator comes out to $1 + x + x^2 + x^3 + x^4$ (by the geometric series finite summation formula). Meanwhile, we note that the denominator is a function of x^5. Define a new polynomial $B(x)$ to help us reduce out this x^5.

$$B(x) = \frac{1}{(1-x)^2(1-x^2)(1-x^5)(1-x^{10})(1-x^{20})}$$

So we can rewrite the whole polynomial as follows:

$$A(x) = (1 + x + x^2 + x^3 + x^4)B(x^5)$$

This is a neat representation. If b_k denotes the coefficient of x^k in $B(x)$, and we define a_k likewise, we have the following relationship:

$$b_k = a_{5k} = a_{5k+1} = a_{5k+2} = a_{5k+3} = a_{5k+4}$$

This is helpful, because we want to find a_n. This tells us, essentially, that it suffices to know b_n. This will come in handy. Let us examine $B(x)$ more closely. We find that it can be rewritten in the following format:

$$B(x) = \frac{C(x)}{(1 - x^{20})^6}$$

where $C(x)$ is a (disgusting, but) finite polynomial. For the sake of this proof, all we need to record is that $C(x)$ is of degree 81. We now invoke the following well-known result in combinatorics, which we use to further simplify $B(x)$.

Lemma: $$\frac{1}{(1-x)^n} = \sum_{k=0}^{\infty} \binom{n+k-1}{n-1} x^k$$

By the above lemma, we can rewrite the expression for $B(x)$ as follows:

$$B(x) = \Big[\sum_{j=0}^{81} c_j x^j\Big]\Big[\sum_{k=0}^{\infty} \binom{k+5}{5} x^{20k}\Big]$$

Here is the key step. Suppose we want to find b_n, the coefficient of x^n in this infinite polynomial $B(x)$. This term is found by multiplying some term from the left sum with some term from the right sum. In other works, we want to find nonnegative integers j, k such that $j + 20k = n$, which implies $n \equiv j \mod 20$. Observe that j runs between 0 and 81. There are 5 such j. Thus, the solution n is a sum of at most 5 terms which we can find in constant time, by checking the congruence condition. ■

This proof might feel a bit cheap due to all of the unforeseeable factoring tricks. The exercise for the reader is to consider how we might generalize this proof to a broader class of coinage. In doing so, the reader should also take note of which steps are not as essential for the generalization (for instance, the reduction from $A(x)$ to $B(x)$).

Zooming Out Let us step back from the familiar American coinage system and consider something a bit more mathematically motivated. What if, instead of pennies, nickels, dimes, etc, we had a coinage system which covered all positive integers? In other words, what if we had a k-cent piece for all $k \in \mathbb{Z}_{\geq 0}$. Then, as you can imagine, our generating function for this coinage system would become:

$$P(x) = \frac{1}{1-x} \cdot \frac{1}{1-x^2} \cdot \frac{1}{1-x^3} \cdot ... = \prod_{k=1}^{\infty} \frac{1}{1-x^k}$$

This is a well-studied generating function known as the **partition generating function**. Each coefficient of its associated Maclaurin series records to number of ways we

can partition a given integer as a sum of other positive integers. This is quite a natural extension of our discussion of change-making, except now, instead of tokenizing some principal with respect to a very restricted set of coins, we have a maximal coinage system. Let us now formalize the concept of the partition.

Definition. Given an integer $n \in \mathbb{Z}_{\geq 0}$, let $p(n) = p_n$ denote the number of distinct ways we can write n as a sum of positive integers. By convention, let $p(0) = 1$. We call $p(n)$ the **partition function** or partition sequence.[3]

There are plenty of helpful visual ways of thinking about partitions. If you're sick of thinking about change-making, you can of the partitions of n as the number of distinct ways we can pile n poker chips. Or you can think of a line segment of length n, and consider how many distinct ways we can split it into subsegments of integer length.

For example, consider $p(5) = 7$. These are the seven ways to partition 5.

$$5 = 4+1 = 3+2 = 3+1+1 = 2+2+1 = 2+1+1+1 = 1+1+1+1+1$$

We can rewrite these sums as *monotone decreasing tuples* (in order to induce order and make counting easier), and we can consider an equivalent relation $w \sim v \iff \sum_i w_i = \sum_i v_i$ (it is left as an exercise for the reader to verify that this is indeed an equivalence relation). One can think of the partition function as counting the size of equivalence classes in this arrangement.

$$(5) \sim (4,1) \sim (3,2) \sim (3,1,1) \sim (2,2,1) \sim (2,1,1,1) \sim (1,1,1,1,1)$$

In order to formalize this notion, we call the tuple $(a_1, ..., a_k)$ be a k-**partition** of n if $a_1 \geq ... \geq a_k$ and $\sum_{i=1}^{n} a_i = n$. As you might expect, p_n counts the number of k-partitions over all $k \leq n$. Call each entry of the tuple a **part**. This allows us to distinguish, for instance, partitions which have exclusively even parts, or partitions in which all parts differ by at most three.

This motivates a more specific question: for a fixed k, how many k-partitions of n are there? For instance, in the above example, there are two ways to partition five into tuples of length 3: $(3,1,1)$ and $(2,2,1)$. Thus, we record the following definition:

Definition. Given integer $n, k \in \mathbb{Z}_{\geq 0}$, with $0 \leq k \leq n$, let $p_{n,k}$ be the number of distinct ways we can write n as a sum of k positive integers. In other words, $p_{n,k}$ counts the number of k-partitions of n. We call $p_{n,k}$ the **refined partition function**, and we make the convention $p_{0,k} = \mathbf{1}[k = 0]$.

In terms of our above example, we have $p_{5,0} = 0, p_{5,1} = p_{5,4} = 1, p_{5,5} = 1$, and $p_{5,2} = p_{5,3} = 2$. As you would expect, this adds up to 7. This is indicative of a general and very important and hopefully self-evident relation:

$$p(n) = p_n = \sum_{k=1}^{n} p_{n,k}$$

The message here is that in order to find p_n, it suffices to know $p_{n,k}$. One could view this progression in analogy with the binomial theorem and Pascal's Triangle. If p_n, the number of partitions of n, could be compared with 2^n, the number of subsets of a set of size n, then $p_{n,k}$ could be compared with $\binom{n}{k}$, the number of subsets of size k. We can arrange this data in a Pascal's Triangle type fashion (see Fig. 1)[4].

[3] Not to be confused with partition functions in the context of statistical mechanics.

[4] It is worth noting that this a rather common technique in combinatorics; for instance, a similar data structure used for counting alternating sign matrices (Bressoud 1999).

n	$p_{n,k}$							p_n
1				1				1
2			1		1			1
3		1		1		1		2
4	1		2		1		1	3
5		1		2		2		5
6	1		3		3		2	7
7		1		3		4		11

Figure 1. "Pascal's Triangle" for partitions.

Recurrence Relation Our goal is to understand the partition function, and perhaps record its fundamental behaviors in the process. Finding an explicit formula is not at all obvious. As such, our first step towards evaluating p_n is creating a recurrence relation regarding $p_{n,k}$.

Suppose $(a_1, ..., a_k)$ is k-partition of n. Since this tuple is by definition decreasing, then for any such partition there are two possible cases: $a_k = 1$ or $a_k > 1$.

Consider the case where $a_k = 1$. Define a map, from the set of k-partitions of n that end with a 1 to the set of $(k-1)$-partitions of $n-1$, which essentially shears off the trailing $a_k = 1$.

$$(a_1, a_2..., a_{k-1}, 1) \mapsto (a_1, a_2..., a_{k-1})$$

We claim that this map is a bijection. This is because it has an inverse, which maps $(k-1)$-tuples to k-tuples by appending a 1 to the end. Since it is a bijection, then the domain and codomain have equivalent cardinality. In other words, the number of k-partitions of n that end with a 1 is equivalent to the total number of $(k-1)$-partitions of $n-1$, which is precisely $p_{n-1,k-1}$.

Now consider the set of partitions such that $a_k > 1$. Then we can find another bijection, between this set and the set of k-partitions of $n-k$, defined as follows:

$$(a_1, a_2..., a_k) \mapsto (a_1 - 1, a_2 - 1, ..., a_k - 1)$$

The inverse is defined by adding 1 to each item. Since this map is bijective, then this subset of k-partitions of n which have $a_k > 1$ is the same size as the set of k-partitions of $n-k$, which is precisely $p_{n-k,k}$.

By combining these two pieces of information, we arrive at the following recursive formula for the refined partition function.

$$p_{n,k} = p_{n-1,k-1} + p_{n-k,k}$$

If we continue with the Pascal's Triangle analogy, this is similar to the recursive formula for the choose function, $\binom{n}{k} = \binom{n-1}{k-1} + \binom{n-1}{k}$.

An Application The key ideas underlying this recurrence relation can come in handy for problem-solving. Consider the following problem from the 2003 Putnam Mathematical Competition. Our solution applies some of the machinery used above.

A1, 2003 Putnam: Let n be a fixed positive integer. How many ways are there to write n as a sum of positive integers, $n = a_1 + a_2 + \ldots + a_k$, with k an arbitrary positive integer and $a_1 \leq a_2 \leq \ldots \leq a_k \leq a_1 + 1$? For example, with $n = 4$ there are four ways: $4, 2+2, 1+1+2, 1+1+1+1$.

Solution: There are n ways.

Let $\tilde{p}_{n,k}$ be the number of k-partitions of n which satisfy the given property—call these *special partitions*. Observe that a special partition $(a_k, ..., a_1)$ (written as a monotone decreasing tuple) must satisfy either $a_k = a_1$ or $a_k = a_1 + 1$.

Claim (exercise for reader): For any special k-partition of n, $k|n \iff a_k = a_1$.

Take an arbitrary special k-partition of n. Consider two cases.

(A) If k divides n, then $a_k = a_1$. In particular, $a_1 = ... = a_k = n/k$. So this special k-partition of n is unique, and hence $\tilde{p}_{n,k} = 1$.

(B) If k does not divide n, then $a_k = a_1 + 1$. Then we have two subcases:

1. $a_1 > 1$. Then a simple bijection tells us that $\tilde{p}_{n,k} = \tilde{p}_{n-k,k}$. We apply this logic iteratively until $a_1 = 1$.
2. $a_1 = 1$. Then we have $\tilde{p}_{n,k} = \tilde{p}_{n-1,k-1}$. This condition will apply until $a_1 = a_k$

By induction, we have that case (B1) reduces to case (B2), which reduces to case (A). This shows that $\tilde{p}_{n,k} = 1$ for all $n, k \in \mathbb{Z}_{\geq 0}$, concluding the proof. ■

Euler's Identity The above proof features a very important theme in integer partition theory, and that is the idea of considering partitions with some kind of restriction. We can formalize this with the notation $p(n|A)$, where A is a condition on the partition. In the above proof, we showed that $p(n|a_k \leq a_1 + 1) = n$.

We can prove many kinds of relationships on many kinds of special partition functions. Take the following, one of many results known as **Euler's identity**:

$$p(n|\text{ odd parts}) = p(n|\text{ distinct parts})$$

In the language of coins, this is saying that the number of valid changes made from exclusively odd-valued coins is the same as the number of valid changes where all coins are different denominations. This is not at all obvious at first glance.

Over the course of this article, we have built up two techniques for solving this kind of problem. The first one, which might be more immediately apparent, is a bijective proof, where we specify some sort of invertible process to translate between the two sets of partitions. But we could also approach this using generating functions—which was, in fact, was Euler's original approach. We show both methods below.

Proof by Bijection

We will construct a bijection from the set of integer partitions of n with odd parts to the set of partitions with distinct parts (also called strict partitions). We describe this function and its inverse as follows.

Given a partition into odd parts, record the frequency of each part in terms of its binary representation $b_k b_{k-1} ... b_1$. Construct a new tuple $(p \cdot 2^{i-1} \cdot b_i)_{i \in \{1,...,k\}}$, with the convention that a zero part does not count. For instance, $(5, 3, 3, 1, 1, 1, 1, 1, 1) \mapsto (6, 5, 4, 2)$, since the binary representation of 6 is 110 (hence, the 4 and 2).

This map is well-defined. Clearly, it does not change the sum n. The parts it creates are distinct, due to the fundamental theorem of arithmetic. Furthermore, $x = y \implies f(x) = f(y)$ due to the uniqueness of binary representations of integers.

This map is bijective because its inverse is given by the following process: split all even parts in equal halves, and repeat until all parts are odd. For instance, $(10, 9, 6) \mapsto (9, 5, 5, 3, 3)$. This reverses the "congealing" of the odd parts.

Thus, the two sets of partitions are equal. ■

Proof by Generating Functions

A more straightforward, symbolic approach to the problem can be obtained using the uniquness of generating function representations. Let us think carefully about the *generative process* behind distinct partitions and odd partitions, respectively.

Let us think back to the coin analogy. An k-partition of n is equivalent to making change for a principal of size n, where we have access to coins of all possible integer values. Any partition of distinct parts must therefore be constructed by taking either zero or one of each coins. We can encode this operation, in the language of generating functions, as follows: $(1 + x)(1 + x^2)(1 + x^3) \cdots$.

Meanwhile, to construct an odd partition, we can take arbitrary amounts of coins that have odd value. It should not be surprising, then, that the generating function is given by: $(1 + x + x^2 + ...)(1 + x^3 + x^6 + ...) \cdots = \frac{1}{(1-x)(1-x^3)(1-x^5)\cdots}$.

It suffices to show that these two generating functions are one and the same. We demonstrate this below. The main trick here is to factor out a difference of squares.

$$\begin{aligned}
\sum_{n=1}^{\infty} p(n|\text{ distinct})x^n &= (1+x)(1+x^2)(1+x^3)(1+x^4)\cdots \\
&= \frac{1-x^2}{1-x}\cdot\frac{1-x^4}{1-x^2}\cdot\frac{1-x^6}{1-x^3}\cdot\frac{1-x^8}{1-x^4}\cdots \\
&= \frac{\prod_{k=1}^{\infty} 1-x^{2k}}{\prod_{k=1}^{\infty} 1-x^k} = \frac{1}{\prod_{k=1}^{\infty} 1-x^{2k-1}} \\
&= \frac{1}{1-x}\cdot\frac{1}{1-x^3}\cdot\frac{1}{1-x^5}\cdots \\
&= \sum_{n=1}^{\infty} p(n|\text{ odd})x^n \quad \blacksquare
\end{aligned}$$

Zooming Out There are many more beautiful and surprising results in basic partition theory, from Euler's Pentagonal Number Theorem to the Jacobi Triple Product Identity to Hardy and Ramanujan's asymptotic formula for the partition function. In this talk, we took a motivated walk towards these results, using the seemingly banal example of change-making to propel us towards a fascinating world of enumerative combinatorics.

Acknowledgment. The author would like to thank the Columbia Undergraduate Math Society for hosting the talk which eventially became this paper.

References

1. John Harris, Jeffry Hirst, Michael Mossinghoff. *Combinatorics and Graph Theory*, Springer (2008).
2. George Andrews, Kimmo Erikkson. *Integer Partitions*, Cambridge (2004).
3. Shallit, Jeffrey. "What this country needs is an 18c piece." *Mathematical Intelligencer*, Springer (2003).