CS 787: Advanced Algorithms

Lecture 15: Compositions and Partitions

Instructor: Eric Bach | Scribe: Siddharth Barman

DRAFT

We continue our discussion on enumeration of combinatorial objects. In this lecture we introduce the concept of compositions and partitions. We also give some counting arguments and formulas for number of partitions finally we describe an algorithm for enumeration of all possible partitions.

1 Composition

A composition of m into l parts is a sum $m = x_1 + x_2 + ... + x_l$, where x_is are integers ≥ 0 and $x_i \leq n$. Note that the ordering of x_is matter i.e. in our scheme of things given $m = 4$ and $l = 2$, $4 = 3 + 1$ and $4 = 1 + 3$ are two different compositions. The key in understanding compositions is to realize that a k subset of n determines a composition of $n - k$ into $k + 1$ parts (see Figure 1). Distances between marked points sum to $n - k$ and including the end points we have $k + 1$ of them. To enumerate composition of m, l parts enumerate $(l - 1)$ subsets of $m + l - 1$.

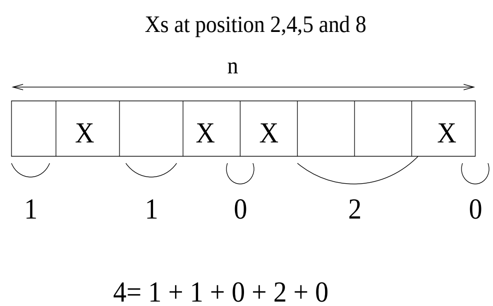

Figure 1: Composition as subsets

Corollary 1. *Number of l length compositions of m are $\binom{m+l-1}{l-1}$.*

2 Partition

Definition 1. *A partition of n is a sum such that $n = x_1 + x_2 + x_3 + ... + x_k$, $x_i \geq 1$ and x_is are integers such that $x_1 \geq x_2 ... \geq x_k$.*

The important point here is that k can vary.
Example: So for $n = 5$ we have the following 7 partitions.
$5 = 5$; $5 = 4 + 1$; $5 = 3 + 2$;
$5 = 3 + 1 + 1$; $5 = 2 + 1 + 1 + 1$;

$5 = 2 + 2 + 1$ and $5 = 1 + 1 + 1 + 1 + 1$. ⊠

Definition 2. *$P(n)$ = Number of partitions of n.*

As shown above P(5)=7. A visual aid in understanding partitions is provided by *Ferrers Diagram*. In Ferrers Diagrams integer k is represented by a tower made of k dots. The diagram suggests a number of combinatorial theorems for partitions. A lot of combinatorial relations are proved using reasoning such as rotation of diagrams (See Figure 2).

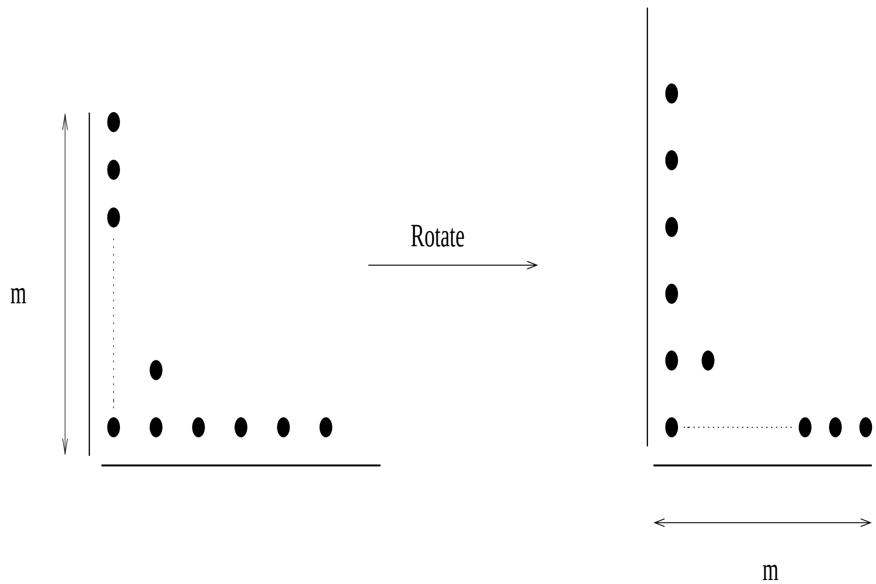

Figure 2: Ferrers Diagram

For $P(n)$ we also have the following generating function,

$$\sum_{n\geq 0} P(n)z^n = \prod_{i\geq 1} \frac{1}{1-z^i}$$

To see why this holds we can expand the terms on the right hand side, $\frac{1}{1-z} = 1+z^{1.1}+z^{2.1}+z^{3.1}+....$ Similarly $\frac{1}{1-z^2} = 1 + z^{1.2} + z^{2.2} + z^{3.2} + ...$

Now we can write $n = m_1.1 + m_2.2 + m_3.3 + ...$, where m_is are the multiplicities. Essentially all possible combinations of m_is are enumerated in the products as well, so the coefficient of z^n equals $P(n)$.

3 Computing P(n)

Definition 3. *We define $P(n, k)$ to be the number of partitions of n with largest part k.*

Euler gave the following recurrence relation, $P(n, k) = P(n-1, k-1) + P(n-k, k)$. The first term counts for the partitions containing exactly one copy of k and the second term contributes for the partitions have more than one copy of k. Essentially given the $P(n-1, k-1)$ partitions for $n-1$ with largest element $k-1$ we add 1 to $k-1$ to get partitions for n with exactly one k. Same thing applies for partitions of $n-k$ where in we add an extra k.

With the following boundary conditions we can compute $P(n,k)$ values in a tabular fashion, $P(0,k) = 0$, $P(1,1) = 1$ and $P(1,2), P(1,3), ... = 0$.

	0	1	2	3	4	5	6	$P(n)$
$n=1$	0	1	0	0	0	0	0	1
2	0	1	1	0	0	0	0	2
3	0	1	1	1	0	0	0	3
4	0	1	2	1	1	0	0	5
5	0	1	2	2	1	1	0	7
6	0	1	3	3	2	1	1	11

$P(n)$ is obtained by computing the row sum.

Theorem 1. *Hardy and Ramanujan (1920s) proved that* $P(n) \sim \frac{1}{4\pi\sqrt{3}} e^{\pi\sqrt{\frac{2n}{3}}}$

This implies that the length of $P(n)$, $l = \Theta(\sqrt{n})$ (number of bits required to express $P(n)$ is $log\ P(n)$). To compute $P(n)$ this way we do $\Theta(n^2)$ additions, which is the number of entries in our table. Each of numbers is $\leq P(n)$. So in all we do $O(n^{5/2})$ i.e. $O(l^5)$ operations.

4 Algorithm for Enumerating Partitions

For this algorithm we use the multiset notation, wherein we express $n = m_1.p_1 + m_2.p_2 + m_3.p_3 + .. + m_l.p_l$, with parts decreasing strictly $p_1 > p_2 > p_3 > ... > p_l$. So the partition $8 = 2+2+1+1+1+1$ is written as $8 = 2.2 + 4.1$.

The output of the algorithm will be in dictionary order. By dictionary order we mean that given $a = a_1a_2a_3...a_l$ and $b = b_1b_2...b_l$ we have $a < b$ iff $\exists i$ st $a_j = b_j$ for $j < i$ and $a_i < b_i$.

The following algorithm is due to Gideon Ehrlich (1973). To get successor of a partition $n = m_1.p_1 + m_2.p_2 + m_3.p_3 + .. + m_l.p_l$, with $p_1 > ... > p_l$ we start with $n.1$ and use one of the following two rules

1. Case 1: $m_l > 1$ (right most element has multiplicity more than one) remove two copies of p_l and replace that by one copy of $p_l + 1$ and $(p_l - 1)$ 1s.

2. Case 2: $m_l = 1$ and $l \geq 2$ remove $m_{l-1}p_{l-1} + p_l$ and replace by one copy of $p_{l-1} + 1$ along with $(m_{l-1} - 1)p_{l-1}$& $(p_l - 1)$ 1s.

For comfort we work through an example.

$n=5$	Case	Action
5.1	1	$2.1 \to 1.2$
1.2 + 3.1	1	$2.1 \to 1.2$
2.2 + 1.1	2	$2 + 2 + 1 \to 1.3 + 2.1$
1.3 + 2.1	1	$2.1 \to 1.2$
1.3 + 1.2	2	$3 + 2 \to 1.4 + 1.1$
1.4 + 1.1	2	$4 + 1 \to 5$
1.5		done

Finally we sketch a proof of correctness of the algorithm. We note that the rules always apply except when the partition is $n = 1.n$. The algorithm is linear and produces unique successors. This means that the partitions are produced in dictionary order one after the other π' with $\pi < \pi'$ with no partition in between.

The algorithm may be implemented very efficiently using a stack. By storing pairs (m_l, p_l) and performing push-pop operations as the case may be. In all the implementation requires $O(P(n))$ arithmetic operations.