Discrete Mathematics
Counting

Saad Mneimneh

1 n choose k

Consider the problem of seating n people on n chairs. In how many ways can we do that? Let's come up with an algorithm that generates a seating. Our algorithm consists of n stages. In stage 1 we choose a person for chair 1, in stage 2 we choose a person for chair 2, etc... In stage n we put the last person in chair n. State 1 can be carried out in n ways, stage 2 can be carried out in $n-1$ ways, etc... In general, stage i (for $i = 1 \dots n$) can be carried out in $n - i + 1$ ways. Therefore, the entire activity can be carried out in $n(n-1)(n-2)\dots 1$ ways. This is equal to $n!$ by definition. Now we need to verify the three conditions:

- every outcome is a valid seating: this is obvious.
- all seatings can be generated: given a seating, we can set the choices for each stage appropriately to generated it.
- every seating is generated exactly once: given a seating, it must have been generated by making a specific choice in stage 1, a specific choice in stage 2, etc...

Therefore, the $n!$ represents the number of possible seatings. In general, $n!$ is the number of permutations (orderings) on n objects. This can be seen by considering chairs to be the ranks. Putting person i on chair j is equivalent to giving person i the j^{th} rank.

Assume now that we only have $k \leq n$ chairs. How many seatings are possible (some people may not be given a chair)? Observe that the same algorithm above when stopped at stage k will work. By the multiplication rule, the number of seatings is

$$n(n-1)\dots(n-k+1) = \frac{n!}{(n-k)!}$$

What happens if $n = k$ (one should be able to retrieve $n!$, read further)? The above quantity is the number of ways of choosing k from n objects if their order is important. To see this, observe that we are effectively choosing k from n people and ranking them (by placing them on chairs).

We are about to make the final step for obtaining n choose k. If all we care about is the set of k people that we choose to be seated, then the above

modified algorithm overcounts our possibilities by $k!$. Why? Observe now that any permutation of a seating results in an equivalent one (because it preserves the set of k people that are seated). But we just learned that the number of permutations on k objects is $k!$. Therefore, the number of ways of choosing k from n objects is:

$$\left(\begin{array}{c} n \\ k \end{array} \right) = \frac{n!}{k!(n-k)!}$$

When $k = 2$, it is easy to verify that we retrieve our previous expression.

$$\left(\begin{array}{c} n \\ 2 \end{array} \right) = \frac{n!}{2!(n-2)!} = \frac{n(n-1)(n-2)(n-3)\dots}{2(n-2)(n-3)\dots} = \frac{n(n-1)}{2}$$

By definition, $0! = 1$ (the empty product). So,

$$\left(\begin{array}{c} n \\ 0 \end{array} \right) = \frac{n!}{0!(n-0)!} = 1$$

Does this make sense? This is essentially saying that there is only one empty subset in a set of size n (clearly!). Similarly, there are $\left(\begin{array}{c} n \\ k \end{array} \right)$ subsets of size k in a set of size n.

2 Sets, relations, functions, and more

A set is an **unordered** collection of elements. It is usually represented by listing the elements between braces separated by commas (each elements appears once). For instance, the following

$$S = \{a, b, c\}$$

is a set called S that contains the elements a, b, and c. The size of a set is denoted by $|\ |$. For example, $|S| = 3$. By $a \in S$ we signify that element a is in S. A set T is a subset of S if all elements of T are in S, we denote this by $T \subset S$. By definition, the empty set is a subset of every set (including itself). A proper subset of S is a subset of S that is not S itself. The intersection of two sets S and T is denoted by $S \cap T$ and is the set of all elements that are in S and in T. The union of two sets S and T is denoted by $S \cup T$ and is the set of all elements that are in S or in T. Sometimes it is useful to describe a set without explicit enumeration of its elements (because it is impossible for instance). Here are few examples:

$$\mathbb{N} = \{1, 2, 3, \dots\}$$

$$\mathbb{Z} = \{\dots, -3, -2, -1, 0, 1, 2, 3, \dots\}$$

$$\mathbb{Q} = \{x | x = a/b \text{ where } a \in \mathbb{Z} \text{ and } b \in \mathbb{N}\}$$

$$S = \{x | x = 2k - 1 \text{ where } k \in \mathbb{N}\} = \{1, 3, 5, \dots\}$$

$$S \cap T = \{x | x \in S \text{ and } x \in T\}$$

$$S \cup T = \{x | x \in S \text{ or } x \in T\}$$

The product of two sets S and T is defined as following set of tuples (note that it is not commutative):

$$S \times T = \{(x, y) | x \in S \text{ and } y \in T\}$$

A relation R on $S \times T$ is a subset of $S \times T$. Consider $S = \{GO, TOO, YOYO\}$ and $T = \{2, 3, 4\}$.

$$S \times T = \{(GO, 2), (GO, 3), (GO, 4), (TOO, 2), (TOO, 3), (TOO, 4),$$

$$(YOYO, 2), (YOYO, 3), (YOYO, 4)\}$$

The following relation maps a word in S to the number of O's in it.

$$R = \{(TOO, 2), (YOYO, 2)\}$$

When every element in S appears exactly in one tuple of R, R is also said to be a function from S to T, denoted $f : S \to T$ and $f(x) = y$ if and only if $(x, y) \in R$. If every element in T appears in at least one tuple, the function is called onto function. A function is a one-to-one correspondence (also called a bijection) if every element in T appears exactly in one tuple, i.e. it is onto and $f(x) \neq f(y)$ if $x \neq y$. If such one-to-one correspondence exists, then $|S| = |T|$. The following function maps every word in S to its length and is a one-to-one correspondence.

$$R = \{(GO, 2), (TOO, 3), (YOYO, 4)\}$$

$$f(GO) = 2, f(TOO) = 3, f(YOYO) = 4$$

The elements of a set can be anything, including sets themselves. Here's the set of all subsets of $S = \{a, b, c, \}$ (we call it the power set):

$$2^S = \{\emptyset, \{a\}, \{b\}, \{c\}, \{a, b\}, \{a, c\}, \{b, c\}, \{a, b, c\}\}$$

where $\emptyset$ is the empty set, which is a subset of every set. Verify that we have $\begin{pmatrix} 3 \\ 0 \end{pmatrix}$ subsets of S of size 0, $\begin{pmatrix} 3 \\ 1 \end{pmatrix}$ subsets of S of size 1, $\begin{pmatrix} 3 \\ 2 \end{pmatrix}$ subsets of S of size 2, and $\begin{pmatrix} 3 \\ 3 \end{pmatrix}$ subsets of S of size 3. If we add them up, we get 8, which is the total number of subsets of S. This leads to a question: how many subsets does a set of size n have?

Here's an algorithm that generates a subset. It consists of n stages. In stage i, the i^{th} element (any order is OK) is either included in the subset or not. It is easy to verify that this algorithm can generate any subset, including the empty set. Moreover, every subset can be generated in only one way (because we go through the elements in a predetermined order). Therefore, by the multiplication rule, and since each stage can be carried out in two ways, the total number of subsets is 2^n.

Here's another way to compute the number of subsets using a one-to-one correspondence. Consider the set of all binary strings of length n. The size of this set is 2^n (each bit can be either 0 or 1, apply the multiplication rule). Without loss of generality, assume that $S = \{1, 2, \ldots, n\}$. Given a subset of S, we can construct a binary string such that the i^{th} bit is 1 if and only if the i^{th} element is in the subset. This is obviously a one-to-one correspondence because every subset defines a unique binary string of length n and every binary string of length n defines a unique subset. Therefore, the number of subsets of $S = \{1, 2, \ldots, n\}$ and the number of binary strings of length n are equal. So $|2^S| = 2^n$. Here's the one-to-one correspondence for $S = \{1, 2, 3\}$.

$$\{(\emptyset, 000), (\{1\}, 100), (\{2\}, 010), (\{3\}, 001),$$
$$(\{1, 2\}, 110), (\{1, 3\}, 101), (\{2, 3\}, 011), (\{1, 2, 3\}, 111)\}$$

We just proved by a combinatorial argument that:

$$\left(\begin{array}{c} n \\ 0 \end{array} \right) + \left(\begin{array}{c} n \\ 1 \end{array} \right) + \ldots + \left(\begin{array}{c} n \\ n \end{array} \right) = \sum_{k=0}^{n} \left(\begin{array}{c} n \\ k \end{array} \right) = 2^n$$

It is also not hard to prove algebraically that:

$$\left(\begin{array}{c} n \\ k \end{array} \right) = \left(\begin{array}{c} n \\ n-k \end{array} \right)$$

$$\left(\begin{array}{c} n \\ k \end{array} \right) = \left(\begin{array}{c} n-1 \\ k-1 \end{array} \right) + \left(\begin{array}{c} n-1 \\ k \end{array} \right), 0 < k < n$$

Can you come up with combinatorial arguments for the two identities above?

Given a set S, we often consider a relation on $S \times S$. I am going to present two types of such relations: an equivalence relation, and a partial order relation. I will use the notation aRb to denote $(a, b) \in R$. I will also denote an equivalence relation by $\equiv$ and a partial order relation by $\prec$.

An equivalence relation satisfies the following three properties:

- reflexive: $a \equiv a$
- symmetric: $a \equiv b \Leftrightarrow b \equiv a$
- transitive: $a \equiv b$ and $b \equiv c \Rightarrow a \equiv c$

For instance, equality is an equivalence relation. Equality of remainder in the division by n is also an equivalence relation. For instance, $8 \equiv_7 15$ means $(8, 15) \in \equiv_7$, where $\equiv_7$ is the relation containing all pairs (a, b) such that a and b have the same remainder in the division by 7. It is not hard to see that $\equiv_n$ is an equivalence relation (simply verify the three properties).

Every equivalence relation defines equivalence classes. An equivalence class is a set of elements that are equivalent. For instance, since the remainder in the division by 7 can take the values 0, 1, 2, 3, 4, 5, and 6, every integer must be equivalent to one of these. This partitions the integers into 7 classes:

$$\{\ldots, -14, -7, 0, 7, 14, \ldots\}$$
$$\{\ldots, -13, -6, 1, 8, 15, \ldots\}$$
$$\{\ldots, -12 - 5, 2, 9, 16, \ldots\}$$
$$\{\ldots, -11, -4, 3, 10, 17, \ldots\}$$
$$\{\ldots, -10, -3, 4, 11, 18, \ldots\}$$
$$\{\ldots, -9, -2, 5, 12, 19, \ldots\}$$
$$\{\ldots, -8, -1, 6, 13, 20, \ldots\}$$

It is always true that the equivalence classes are disjoint and their union is equal to the entire set (that's why we say a partition of S into equivalence classes). To prove this, let C_a be the equivalence class for a, i.e.

$$C_a = \{b \in S | b \equiv a\}$$

Since $a \equiv a$, every element $a \in S$ must appear in some equivalence class C_a. Therefore,

$$\bigcup_{a \in S} C_a = S$$

Now take C_a and C_b. We will prove that if they are not disjoint, then they must be equal. If $C_a \cap C_b \neq \emptyset$, then let $c \in C_a \cap C_b$. Therefore, $c \equiv b$ and (by symmetry) $a \equiv c$. By transitivity, $a \equiv b$:

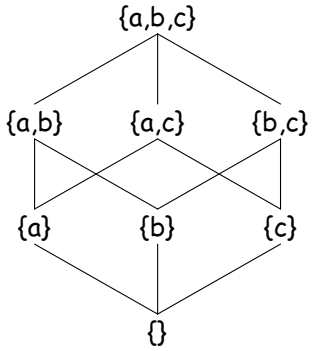
$$e \in C_a \Rightarrow e \equiv a \Rightarrow e \equiv b \text{ (by transitivity) } \Rightarrow e \in C_b$$

We conclude that $C_a \subset C_b$. Similarly, $C_b \subset C_a$, so $C_a = C_b$.

A partial order relation satisfies the following:

- anti-symmetric: $a \prec b \Rightarrow b \not\prec a$, i.e. $(a, b) \in \prec \Rightarrow (b, a) \notin \prec$
- transitive: $a \prec b$ and $b \prec v \Rightarrow a \prec c$.

It can be either reflexive or non-reflexive. An example of a partial order relation is the $<$ (or $\leq$) relation on $\mathbb{R}$. Another example is set inclusion on the power set of a given set S. One way to visually represent a partial order relation is by using a Hasse diagram. Here's an example for the set inclusion on the power set of $S = \{a, b, c\}$.



In a Hasse diagram, we draw a line going up from a to b if $a \prec b$; however, we do not show all the lines. We omit all the lines that can be inferred by transitivity. This leads to the concept of the transitive closure of a relation R, denoted by $TC(R)$. $TC(R)$ is the smallest transitive relation that includes R. Therefore, a partial order relation is the transitive closure of its Hasse diagram (the relation explicitly depicted by the Hasse diagram). In the above example, the Hasse diagram explicitly depicts the relation

$$R = \{(\phi, \{a\}), (\phi, \{b\}), (\phi, \{c\}),$$

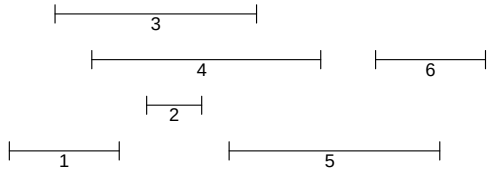
$$(\{a\}, \{a, b\}), (\{a\}, \{a, c\}), (\{b\}, \{a, b\}), (\{b\}, \{b, c\}), (\{c\}, \{a, c\}), (\{c\}, \{b, c\}),$$

$$(\{a, b\}, \{a, b, c\}), (\{a, c\}, \{a, b, c\}), (\{b, c\}, \{a, b, c\})\}$$

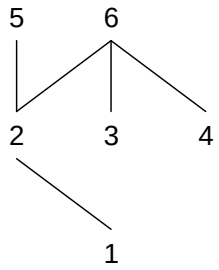
and set inclusion $\subset$ on 2^S is nothing but $TC(R)$.

A nice remark is that the Hasse diagram above looks like a three dimensional cube. This is not a coincidence. If $|S| = n$, then the Hasse diagram for set inclusion on the power set of S will look like a cube in n dimensional space. Recall that there is a one-to-one correspondence between subsets of S and binary strings of length n. Those in turn, when regarded as points in n dimensional space, are the corners of a unit cube.

A typical application of partial orders is in scheduling. For instance, consider the following tasks with their starting time and ending time. If two tasks overlap, they cannot be scheduled on the same machine. The question is how many machines are needed to perform all the tasks.



If we let $a \prec b$ means task a ends before task b starts, then we can easily show that $\prec$ is a partial order relation (simply verify that it is anti-symmetric and transitive). The Hasse diagram for the above set of tasks is the following:



To schedule the tasks, we need to partition the above diagram into disjoint **chains**. For instance, $\{\{1,2,6\},\{5\},\{3\},\{4\}\}$. Every chain represents a set of tasks that may be scheduled on the same machine. The question then reduces to finding the smallest number of disjoint chains that partition the Hasse diagram. It turns out that this number is equal to the size of the largest **antichain**. An antichain is a set A such that if $a, b \in A$, then $a \not\prec b$. In other words, it is a set of elements that are pairwise unrelated by $\prec$. Given a partition into disjoint chains, an antichain can include at most one element from each chain. For the above example, we can form $A = \{1,3,4\}$. Therefore, the size of any partition is always an upper bound on the size of any antichain. When the sizes meet, we have an optimal scheduling. For instance, the partition $\{\{1,2,5\},\{3,6\},\{4\}\}$, and the antichain $\{1,3,4\}$ have the same size. So that's the best we can do, we need three machines.

When a partial order relation is defined on a finite set S, we can always find a *minimum*. A minimum is an element x of S, such that no element y of S satisfies $y \prec x$. The proof is by contradiction. Assume that no minimum exists, then starting from an arbitrary element, we can find a sequence such that $a_1 \prec a_2 \prec a_3 \prec \ldots$. Since the set S is finite, we must cycle, i.e. we find a sequence $a_i \prec a_{i+1} \prec a_{i+2} \prec \ldots \prec a_{i+n} \prec a_i$. By transitivity (repeated), $a_i \prec a_{i+n}$. This means that the relation is symmetric, a contradiction. The same reasoning can be applied for the concept of a *maximum*.

3 Anagrams

How many anagrams (not necessarily meaningful) can we build from a given word? A quick answer would be as many ways the letters of the word can be permuted. For example, the word MATH has four letters and every permutation of those letters will make an anagram. Therefore, we have $4! = 24$ anagrams that we can build from the word MATH. But what if the word was MATHEMATICA, can we apply the same reasoning? We now have 11 letters and hence we have 11! anagrams. As it turns out, this is wrong. The letters now are not unique.

The number of anagrams of an n-letter word depends on how many times letters of the word are repeated. For instance, permuting the Ms or the As or the Ts in the word MATHEMATICA will result in the same anagram. M

is repeated twice, A is repetead three times, and T is repeated twice. The number of permutations that preserve the anagram is $2!3!2! = 24$. Therefore, the 11! permutations overcount the number of anagrams by 24. The number of anagrams that we can build from the word MATHEMATICA is

$$\frac{11!}{2!3!2!}$$

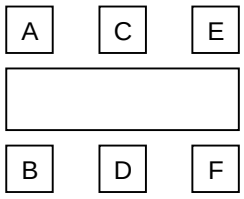
In general, given a n-letter word with k unique letters where letter i appears n_i times, the number of anagrams that we can build from this word will be

$$\frac{n!}{\prod_{i=1}^{k} n_i!} = \frac{n!}{n_1!n_2!\ldots n_k!}$$

4 Forming teams

Given $2n$ players, we would like to pair them up to form n teams of two. How many configurations of teams are possible? I will present three ways for counting those configurations.

Method 1: Imagine a large table with n chairs on each side (a total of $2n$ chairs). Seating the players on the chairs obviously defines the teams (players who face each other on the table are in the same team). It should be clear by now that this seating can be done in $(2n)!$ ways (a permutation). But does that really represent the number of possible configurations of teams? Consider the following example where the teams are (A, B), (C, D), (E, F):



Permuting players within teams produces equivalent configurations. That's a $2 \times 2 \times 2$ multiplicity. Similarly, permuting teams on the table produces equivalent configurations. That's a 3! multiplicity. Therefore, each configuration in the above example is counted exactly $2^3 3!$ times. In general, the $(2n)!$ represent an overcounting by $2^n n!$. The number of possible configurations of teams is therefore:

$$\frac{(2n)!}{2^n n!}$$

Method 2: An algorithm to generate teams could proceed as follows. We first choose two players from $2n$ to form the first team. Then we choose another two players from $2n-2$ to form the second team, etc... This consists of n stages where in stage i $(i = 1, \ldots, n)$ we choose 2 players from $2n-2(i-1)$. Obviously,

stage i can be carried out in $\begin{pmatrix} 2n-2i+2 \\ 2 \end{pmatrix}$ ways. By the multiplication rule, the entire activity can be carried out in

$$\begin{pmatrix} 2n \\ 2 \end{pmatrix}\begin{pmatrix} 2n-2 \\ 2 \end{pmatrix}\begin{pmatrix} 2n-4 \\ 2 \end{pmatrix}\cdots\begin{pmatrix} 2 \\ 2 \end{pmatrix}$$

ways. However, there is overcounting. By permuting the n choices we still obtain an equivalent configuration. Therefore, we are overcounting by $n!$. The number of configurations of teams is therefore:

$$\frac{\begin{pmatrix} 2n \\ 2 \end{pmatrix}\begin{pmatrix} 2n-2 \\ 2 \end{pmatrix}\begin{pmatrix} 2n-4 \\ 2 \end{pmatrix}\cdots\begin{pmatrix} 2 \\ 2 \end{pmatrix}}{n!}$$

It is not hard to verify algebraically that the two expressions obtained in Method 1 and Method 2 are equal.

Method 3: Here's another algorithm to generate teams. Fix an order on the players; for instance, based on their height. The algorithm proceeds as follows. Repeatedly pick the shortest player among the remaining ones and choose a partner. This is done in n stages. The first stage can be carried out in $2n-1$ ways. The second stage can be carried out in $2n-3$ ways, etc... In general, the i^{th} stage can be carried out in $2n-2i+1$ ways. By the multiplication rule, the entire activity can be carried out in

$$(2n-1)\times(2n-3)\times\ldots\times 5\times 3\times 1$$

ways. That's the product of the first n odd numbers. Does this expression overcounts? The answer is no. To see this, consider a possible outcome, say $(A,B),(C,D),(E,F)$ when $n=3$. Our algorithm can definitely generate this configuration. More importantly, it can generate this configuration only once! Why? Assume without loss of generality that $A<B<C<D<E<F$. The only way to generate this configuration is by choosing B as a partner for A, then D as a partner for C, then F as a partner for E. We have established the following identity:

$$1\times 2\times 3\times\ldots\times(2n-1)=\frac{(2n)!}{2^n n!}$$

The result can be generalized to mn people and teams of size m as follows

$$\frac{(mn)!}{(m!)^n!}$$

This can be further generalized to non-homogenous teams using the same counting approach. Given n people, we can form α_i teams of size i, $1\leq i\leq n$, where $\sum_{i=1}^{n} i\alpha_i = n$ in the following number of ways:

$$\frac{n!}{\prod_{i=1}^{n} i!^{\alpha_i}\alpha_i!}$$

As an example, consider 25 people to be placed into 4 groups of size 3, two of size 4, and one of size 5. This is how many ways it can be done:

$$\frac{25!}{3!^4 4!^2 5!^1 4!2!1!}$$

5 The binomial theorem (Pascal triangle)

You might have seen the Pascal triangle before (each entry is the sum of the two entries above it):

$$\begin{array}{c}
1 \\
1 \quad 1 \\
1 \quad 2 \quad 1 \\
1 \quad 3 \quad 3 \quad 1 \\
1 \quad 4 \quad 6 \quad 4 \quad 1 \\
1 \quad 5 \quad 10 \quad 10 \quad 5 \quad 1 \\
1 \quad 6 \quad 15 \quad 20 \quad 15 \quad 6 \quad 1 \\
1 \quad 7 \quad 21 \quad 35 \quad 35 \quad 21 \quad 7 \quad 1 \\
\vdots
\end{array}$$

You might have used it to expand expressions of the form $(x+y)^n$ where n is a non-negative integer. For instance, here are a few attempts:

$$(x+y)^0 = 1$$
$$(x+y)^1 = x+y$$
$$(x+y)^2 = x^2+2xy+y^2$$
$$(x+y)^3 = x^3+3x^2y+3xy^2+y^3$$
$$(x+y)^4 = x^4+4x^3y+6x^2y^2+4xy^3+y^4$$

Observe that the coefficients in the expansion are exactly those that appear in the corresponding row of the Pascal triangle. These are called the binomial coefficients. It is not hard to see that the k^{th} term in the n^{th} row is nothing but $\left(\begin{array}{c} n \\ k \end{array} \right)$ (indexing starts at 0). The symmetry is obvious, another illustration that $\left(\begin{array}{c} n \\ k \end{array} \right) = \left(\begin{array}{c} n \\ n-k \end{array} \right)$. Also, we see that $\left(\begin{array}{c} n \\ k \end{array} \right)$ is largest when $k = \lfloor n/2 \rfloor$. So how big is $\left(\begin{array}{c} n \\ n/2 \end{array} \right)$? Let's find out. We can use Stirling's approximation of $n!$ (when n is large):

$$n! \sim \sqrt{2\pi n}\left(\frac{n}{e}\right)^n$$

$$\left(\begin{array}{c} n \\ n/2 \end{array} \right) = \frac{n!}{(n/2)!(n/2)!} \sim \frac{\sqrt{2\pi n}(\frac{n}{e})^n}{\sqrt{2\pi n/2}(\frac{n/2}{e})^{n/2}\sqrt{2\pi n/2}(\frac{n/2}{e})^{n/2}} = \frac{\sqrt{2\pi n}(\frac{n}{e})^n}{\pi n(\frac{n}{2e})^n} = \sqrt{\frac{2}{\pi n}}2^n$$

Back to $(x+y)^n$. The binomial theorem claims that:

$$(x+y)^n = \sum_{k=0}^{n} \left(\begin{array}{c} n \\ k \end{array} \right) x^{n-k} y^k$$

The proof is easy. Here's the argument when $n = 5$, but it works in general. Think of expanding

$$(x+y)^5 = (x+y)(x+y)(x+y)(x+y)(x+y)$$

We get each term in the expansion by choosing one of the two terms in each factor, and multiplying them. If we choose x 2 times, then we must choose y 3 times, and so we get x^2y^3. How many times do we get this same term? Clearly, as many times as the number of ways to choose the three factors that supply y (the remaining factors supply x). Thus we have to choose three factor from five, which can be done in $\left(\begin{array}{c} 5 \\ 3 \end{array} \right)$ ways. Therefore, the expansion looks like this:

$$(x+y)^5 = \left(\begin{array}{c} 5 \\ 0 \end{array} \right) x^5 + \left(\begin{array}{c} 5 \\ 1 \end{array} \right) x^4 y + \left(\begin{array}{c} 5 \\ 2 \end{array} \right) x^3 y^2 + \left(\begin{array}{c} 5 \\ 3 \end{array} \right) x^2 y^3 + \left(\begin{array}{c} 5 \\ 4 \end{array} \right) x y^4 + \left(\begin{array}{c} 5 \\ 5 \end{array} \right) y^5$$

We can apply this argument in general to obtain the binomial theorem. Here are a few quick applications of the binomial theorem.

$$(1+1)^n = \left(\begin{array}{c} n \\ 0 \end{array} \right) + \left(\begin{array}{c} n \\ 1 \end{array} \right) + \ldots + \left(\begin{array}{c} n \\ n \end{array} \right) = 2^n$$

$$(1-1)^n = \left(\begin{array}{c} n \\ 0 \end{array} \right) - \left(\begin{array}{c} n \\ 1 \end{array} \right) + \ldots + (-1)^n \left(\begin{array}{c} n \\ n \end{array} \right) = 0^n$$

which is 0 when $n > 0$ and 1 when $n = 0$. This translates to the following: If the terms of a row in the Pascal triangle alternate signs, they add up to 0, except for the first row (when $n = 0$) which adds up to 1. Looking at the Pascal triangle, this result is obvious due to symmetry when n is odd, but not when n is even.

6 Selection with repetition

When it comes to choosing k from n, the selection can be either ordered or non-ordered. But what if we allow repetitions? So far we have developed results when repetition is not allowed. This is illustrated in the table below.

	ordered	unordered
no repetition	$\frac{n!}{(n-k)!}$	$\left(\begin{array}{c} n \\ k \end{array} \right)$
repetition	?	?

In this section, we will complete the table. But why is it useful to do so? Often we can model our problem as a selection process. The modeling is the hard part. Once it is done, we can immediately figure out the answer from the table above. Here's a guide on how to model the problem. We are usually dealing with two sets A and B. First, determine the set that you are choosing from. The size of this set will be n, say $|A| = n$, and you will be choosing $|B| = k$ from n. Second, determine the property of the selection in the following way: (a) repetition/no repetition: can you choose an element in A more than once? (b) ordered/unordered: this often boils down to whether elements in B are distinguishable (ordered) or not (unordered). Third, choose your answer from the table.

We will first complete the table and then follow it by an example to illustrate the above process. Let's start with the easy one first when the selection is ordered and repetition is allowed. Let's consider an example of choosing two elements from of the set $S = \{a, b, c\}$. The possibilities are:

$$aa \quad ab \quad ac$$

$$ba \quad bb \quad bc$$

$$ca \quad cb \quad cc$$

In general, let's figure out an algorithm that generates all the possible outcomes. This consists of k stages, where in each stage i we simply make the i^{th} choice to be any of the n elements (because repetition is allowed). Each stage can be carried out in n ways. Therefore, the entire activity can be carried out in n^k ways. And this is what goes into the table. Another way of deriving the result is by forming a word length k where our alphabet size is n. Every letter in the word has n possibilities resulting in n^k possible words. Observe now that n and k are not related, i.e. we do not require that $k \leq n$.

	ordered	unordered
no repetition	$\frac{n!}{(n-k)!}$	$\binom{n}{k}$
repetition	n^k	?

What happens when we ignore the order. For instance, in the above example ab and ba are considered the same. Here's the list of all possible outcomes when the selection is unordered:

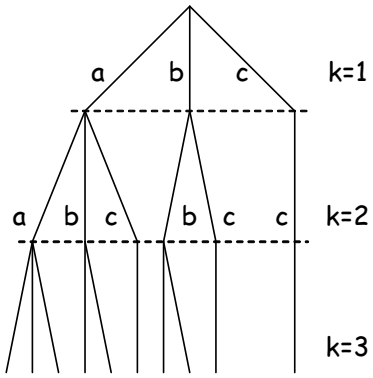
$$aa \quad ab \quad ac$$

$$\quad\; bb \quad bc$$

$$\quad\quad\; cc$$

The number drops from 9 ($n^k = 3^2$) to 6. How do we compute this number in general? Arguments based on overcounting cannot be easily adapted here because each outcome is overcounted a different number of times. Since order

is not relevant, one might think about an algorithm for generating a valid outcome by choosing the a's first, then the b's, then the c's, etc... This can be represented by the following tree where each outcome appears exactly once (no overcounting).



However, it is not obvious how to count the number of leaves in this tree. The branching in the tree depends on the choices made so far. In other words, when viewing this algorithm as an activity that consists of a number of phases (e.g. two when $k = 2$), the number of ways a phase can be carried out depends on the choices made for the previous phases. Therefore, we cannot easily apply the multiplication rule here. In fact, if $A(n, k)$ is the number we are looking for, analyzing the structure of the tree will give:

$$A(n,k) = \sum_{i=1}^{n} A(i, k-1)$$

where $A(n, 0) = 1$ and $A(0, k) = 0$ for $k > 0$. While this provides a way to compute $A(n, k)$, it does not help in terms of getting a nice expression for $A(n, k)$.

Observe that what is important is the number of times each element is selected.

	a	b	c
aa	2	0	0
ab, ba	1	1	0
ac, ca	1	0	1
bb	0	2	0
bc, cb	0	1	1
cc	0	0	2

The sum of each row in the table above is 2 (k in general). The table depicts a function from the words of size two to an **ordered partition** of the integer 2 (k in general).

$$\{(aa,(2,0,0)),(ab,(1,1,0)),(ba,(1,1,0)),(ac,(1,0,1)),(ca,(1,0,1)),$$

$$(bb,(0,2,0)),(bc,(0,1,1)),(cb,(0,1,1),(cc,(0,0,2))\}$$

But if we consider ab and ba to be the same, and similary for ac and ca, and bc and ca, then we are ready to make a one-to-one correspondence. Each selection corresponds to n non-negative integers $x_1, \ldots x_n$ that add up to k. Similarly, every $x_1, \ldots, x_n$ non-negative integers that add up to k uniquely define a selection. Therefore, it is enough to count the number of solutions for the following equation:

$$x_1 + x_2 + \ldots + x_n = k$$

$$0 \leq x_i \leq k$$

In other words, in how many ways can we partition k into n parts (parts are distinguishable and may be empty)? Consider the example of $k = 8$ and $n = 6$. The following depicts a possible partitioning ($x_1 = 2, x_2 = 1, x_3 = 0, x_4 = 3, x_5 = 1, x_6 = 1$).

In general, any partitioning can be encoded as a string of k 0's and $n-1$ 1's. How many such strings can we have? We simply have to choose k 0's from $n+k-1$ bits. This is $\left(\begin{array}{c} n+k-1 \\ k \end{array} \right)$. Note that when $n = k = 0$, the answer should be 1. Therefore, we define $\left(\begin{array}{c} -1 \\ 0 \end{array} \right) = 1$. We now complete our table (again observe that we do not require that $k \leq n$):

	ordered	unordered
no repetition	$\frac{n!}{(n-k)!}$	$\left(\begin{array}{c} n \\ k \end{array} \right)$
repetition	n^k	$\left(\begin{array}{c} n+k-1 \\ k \end{array} \right)$

Because $A(n+1, k+1) = \sum_{i=0}^{n} A(i+1, k)$, we have also established the following:

$$\left(\begin{array}{c} k+n+1 \\ k+1 \end{array} \right) = \sum_{i=0}^{n} \left(\begin{array}{c} k+i \\ k \end{array} \right)$$

Let's apply the knowledge that we have acquired so far. Assume that we have n kids and k gifts, and we wish to distribute the gifts among the kids. All gifts must go. Furthermore, consider the following scenarios:

Scenario A.1: gifts are identical and every kid must receive at most one gift. We have two sets, the set of kids and the set of gifts. Since all gifts must go, let us model our problem as a selection from the set of kids. We must choose k from n kids. With each selection, a gift is given. Can we choose a kid more than once? No because every kid must receive at most one gift. Is the order of the selection important? No because gifts are identical. The answer is $\left(\begin{array}{c} n \\ k \end{array} \right)$. Note that $k \leq n$ is implicit in this scenario.

Scenario A.2: gifts are identical and kids may receive multiple gifts. Again we must choose k from n kids. Can we choose a kid more than once? Yes because kids may receive multiple gifts. Is the order of the selection important? No because gifts are identical. The answer is $\left(\begin{array}{c} n+k-1 \\ k \end{array} \right)$.

Scenario A.3: gifts are identical and kids may receive multiple gifts. But every kid must receive at least one gift. We first give each kid a gift and then handle the $k-n$ remaining gifts in the same way we did in **A.2**. The answer is $\left(\begin{array}{c} n+(k-n)-1 \\ k-n \end{array} \right) = \left(\begin{array}{c} k-1 \\ k-n \end{array} \right)$.

Scenario B.1: gifts are distinguishable and every kid must receive at most one gift. Again we must choose k from n kids. Can we choose a kid more than once? No because every kid must receive at most one gift. Is the order of the selection important? Yes because gifts are distinguishable. The answer is $n!/(n-k)!$. Note that $k \leq n$ is implicit in this scenario.

Scenario B.2: gifts are distinguishable and kids may receive multiple gifts. Again we must choose k from n kids. Can we choose a kid more than once? Yes because kids may receive multiple gifts. Is the order of the selection important? Yes because gifts are distinguishable. The answer is n^k.