Counting Tips

We have only a few tools at our disposal for counting: we can find the number of ways to complete task X followed by task Y given the number of ways to complete each individually (and some assumptions of independence of these numbers), we can count the number of subsets of A of a given size (or all sizes), we can count the number of ordered subsets of A of a given size, and we can use set-theoretic manipulations. The whole trick is to figure out which of these to use; much of the time you need more than one or to use one in multiple ways. Much of the time there is more than one perfectly valid path to the correct number, as well.

- Coin-toss problems: to find the number of sequences of 12 coin flips with exactly 3 heads, we are ordering 9 indistinguishable tails and 3 indistinguishable heads. The shortest path to the correct number is to think not of ordering them, but of picking a subset of the 12 positions to hold the heads; the rest will hold tails: $\binom{12}{3}$.

  However, you can also do it by ordering the 12 flips and then dividing out by your overcount. A given sequence of heads and tails has been counted (number of ways to order the heads)(number of ways to order the tails) times, so we need to divide the total by that value: $12!/(3!9!)$

- The previous idea extends to more than two kinds of objects. How many rolls of 10 dice have two 1s, three 2s, four 3s, and a 4? We can do this by serially choosing subsets of the positions: $\binom{10}{2}\binom{8}{3}\binom{5}{4}$ (with an invisible $\binom{1}{1}$ at the end). We can also do it by ordering our ten objects, 10! ways, and dividing out by the number of ways to order each subset of indistinguishable elements: 2!3!4!. Writing out the product of combinations and then doing some basic cancellation takes you from that formula to this one.

- The set-theoretic manipulations I've talked about typically come into play when you have either a property for which the objects lacking that property are easier to count, or a universe which naturally partitions into subsets for which some value is fixed, and you want a collection of those subsets. Looking at the latter, if you consider a group of 10 people in their 20s, 12 in their 30s, and 10 in their 40s from which a committee of 4 for a civic organization is to be drawn, if you then are wondering about how many committees have certain age makeups, the natural partition is to fix the number of people in each age bracket and count the committees with exactly that number. The size of this universe is the number of subsets of size 4 from a set of size 32, $\binom{32}{4}$.

Letting $x/y/z$ represent the number of people in their 20s, 30s, and 40s, respectively, on the committee, we have the 15 partition sets 0/0/4, 0/1/3, 0/2/2, 0/3/1, 0/4/0, 1/0/3, 1/1/2, 1/2/1, 1/3/0, 2/0/2, 2/1/1, 2/2/0, 3/0/1, 3/1/0, 4/0/0.

Different questions about the same natural partition lend themselves to different approaches. If we want to know the number of committees with at least one member in his or her 20s, there are 10 partition sets meeting that criterion and 5 that don't, so counting the ones that don't (working by complement) is less work: 0/0/4 is $\binom{10}{4}$, 0/1/3 is $12\binom{10}{3}$, 0/2/2 is $\binom{12}{2}\binom{10}{2}$, 0/3/1 is $\binom{12}{3}10$, and 0/4/0 is $\binom{12}{4}$. Therefore we want $\binom{32}{4} - \binom{10}{4} - 12\binom{10}{3} - \binom{12}{2}\binom{10}{2} - 10\binom{12}{3} - \binom{12}{4}$.

On the other hand, if we wanted the number of ways to form committees with at least one member of each age bracket, we have only three partition sets meeting that condition, and so working directly is more efficient: 1/1/2 is $10 \cdot 12\binom{10}{2}$, 1/2/1 is $10\binom{12}{2}10$, and 2/1/1 is $\binom{10}{2}12 \cdot 10$. We want their sum, $240\binom{10}{2} + 100\binom{12}{2}$.

- Key words to look for to consider partitions are “at most”, “at least”, “(no) more than”, and “(no) fewer than”. Situations which aren't numerical partitions but might be easier done by complement are often signaled by the English description being the property *not* satisfied, such as “bitstrings that are not palindromes”. This is not a sharp line; to count “letter strings that are not all vowels”, you could think of the set of strings as partitioning into subsets with fixed numbers of vowels, or you could think of “all vowels” and “not all vowels” as complements without subdividing further. The computation would be the same in each case.