Code Jam to I/O for Women 2017 - Code Jam to I/O for Women

Analysis: Understudies

Understudies: Analysis

Small dataset

There can be at most four roles and eight performers in a Small test case. We could write code to enumerate all of the different ways to pair up the performers, but this can be tricky to get right; however, the limits allow an even easier brute-force solution. It is simple to generate all possible permutations of performers using, for example, Python's `itertools.permutations`. For each permutation, we can pair the first performer in the list with the second, the third with the fourth, and so on, and then check the overall success probability in the way described in the sample case explanations. The maximum probability we encounter is the answer. This method will check equivalent casting decisions multiple times, but it doesn't matter; 8! is under 50 thousand, so your computer should not even break a sweat.

Large dataset

Brute force will not cut it for a musical with up to 40 roles — we must be giving *Cats* and *A Chorus Line* a run for their money! — so we need a more thoughtful strategy. Is it better to pair up reliable performers with other reliable performers, making some roles secure and others risky? Or should we pair our most reliable performers with our least reliable ones, spreading out the risk more evenly across roles? Or is the solution something more complex? Intuition can easily fool us when it comes to probability problems, so it's best to make a mathematical argument.

Suppose that we have paired up our performers in some arbitrary way. Let's consider two of the roles and the four performers. Without loss of generality, let's label the four performers A, B, C, and D, such that their probabilities of becoming unavailable follow the order $P_A \geq P_B \geq P_C \geq P_D$. Our key claim is that we will maximize these four performers' contribution to the overall probability of the show's success by pairing A with D and B with C, if they are not already paired in that way.

Let's prove that pairing A with D is better than pairing A with C. If we pair A with D, the probability that both of these roles will be successfully filled is as follows:

$$(1 - P_AP_D)(1 - P_BP_C) = 1 - P_AP_D - P_BP_C + P_AP_BP_CP_D.$$

If we instead pair A with C, the probability of success is:

$$(1 - P_AP_C)(1 - P_BP_D) = 1 - P_AP_C - P_BP_D + P_AP_BP_CP_D.$$

Subtracting the second quantity from the first, we get:

$$P_AP_C + P_BP_D - P_AP_D - P_BP_C = P_A(P_C - P_D) + P_B(P_D - P_C) = (P_A - P_B)(P_C - P_D).$$

Since we know that $P_A \geq P_B$, and $P_C \geq P_D$, both of the terms in the final expression above must be positive or zero, and so their product is also positive or zero. That means that we are at least as likely to succeed if we pair A with D as we are if we pair A with C. (The contributions from performers other than A, B, C, and D are identical across these two cases, so we don't need to consider them here.) A similar argument shows that pairing A with B cannot possibly be better than pairing A with D.

So we have proven that in any arrangement, for any pair of roles, we should reassign the four performers as necessary to pair up the most reliable and least reliable of the four in one role, and the other two performers in the other. This implies that the most and least reliable performers in the entire cast should be assigned to the same role. (If they are in two different roles, just apply the argument above to that pair of roles.) Similarly, the second-most and second-least reliable performers should be in the same role, and so on. Only the rank orders of the performers' probabilities turn out to matter; the actual values are not important!

This makes our solution very simple: sort the probabilities and then pair up the extremes, and then the remaining extremes, and so on. This is O(**N** log **N**) — it would be linear if not for the sorting step — and it would easily work for values of **N** much larger than 40.