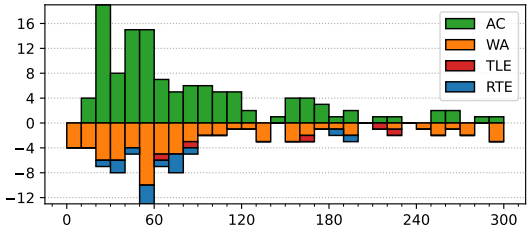
Problem A: Riddle Of The Sphinx

Problem authors:
Martin Kacer and Per Austrin

Solved by 120 teams.
First solved after 10 minutes.
Shortest team solution: 532 bytes.
Shortest judge solution: 347 bytes.



This was an easy interactive problem and there are many ways to solve it. A general observation for this type of problem (which is maybe a bit overkill for the present situation) is the following. Consider the 5×3 matrix

$$A = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \\ a_{41} & a_{42} & a_{43} \\ a_{51} & a_{52} & a_{53} \end{pmatrix}$$

where the i'th row are the three numbers you give in your i'th question. Then, if any 3 rows of A are linearly independent, we can uniquely determine the correct answer. This is the case because if we remove one of the truthful answers, we will have an inconsistent system of equation, but if we remove the lie, then we will have a consistent overdetermined system of equations. In other words we can uniquely identify which answer is the lie, and then we can recover the correct answer using any three of the other answers.

Since we can choose A freely it is nicest to choose it in such a way that the answer is easy to recover, e.g.

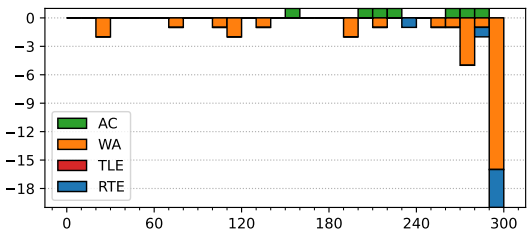
$$A = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 1 & 1 \\ 1 & 2 & 3 \end{pmatrix}$$

Note that we cannot change the last query to $(1, 1, 2)$, because then the last three answers would be linearly dependent, and if one of he first two questions was a lie we would not be able to figure out which one of them was a lie. Similarly, $(0, 1, 2)$ does not work as the last query, as then the second, third and last question are linearly dependent.

Problem B: Schedule

Problem authors:
The World Finals judges and Bob Roos

Solved by 7 teams.
First solved after 153 minutes.
Shortest team solution: 1059 bytes.
Shortest judge solution: 532 bytes.



Suppose we have some schedule with isolation k. Then during the first k weeks, every pair of individuals on different teams meet at least once, and if we repeat the schedule for these first k weeks indefinitely, we get a periodic schedule for arbitrarily many weeks with the same

separation. This means that the problem is equivalent to finding the smallest k such that all people can meet at least once in k weeks. If $k \leq w$ then we take that schedule and repeat it to get a full schedule of w weeks, and if $k > w$ then the answer is `infinity`.

We can view a k-day schedule for one team as a binary string x of length k, with $x_i = 0$ indicating that the first team member comes to work on day i, and $x_i = 1$ indicating that the second team member comes to work on day i. Two binary strings x and y are compatible if for all four combinations $c \in \{00, 01, 10, 11\}$ there is some i such that $x_i y_i = c$. A schedule for n teams is then a list of n binary strings that are pairwise compatible.

We claim such a schedule exists if $n \leq \binom{k-1}{\lceil k/2 \rceil}$. To see this, consider enumerating all binary strings of length k with a 0 in the first bit, followed by some binary string of length $k-1$ with $\lceil k/2 \rceil$ ones. It is not hard to see that any pair of such strings are pairwise compatible: since they both have the same number of ones and are different strings, there must be some index where we see the combinations 01 and 10; since they both start with 0 there is an index where we see the combination 00; and since they both have more than $(k-1)/2$ ones among the last $k-1$ bits, there must be some index where we see the combination 11.

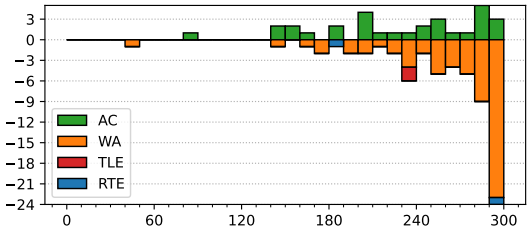
We also claim that such a schedule cannot exist if $n > \binom{k-1}{\lceil k/2 \rceil}$. To see this, note first that without loss of generality we can assume that all strings in a schedule start with 0 (because if we flip 0 and 1 in a string, it remains compatible with the same strings). Also, since there must both be some i such that $x_i y_i = 01$ and some j such that $x_j y_j = 10$, any list of pairwise compatible schedules must be an *antichain* in the Boolean lattice, which by Sperner's Theorem implies that there can be at most $\binom{k-1}{\lceil (k-1)/2 \rceil}$ such schedules. If k is even this matches the claimed bound. If on the other hand k is an odd number, this only gives a bound of $\binom{k-1}{(k-1)/2}$ whereas the claimed bound is $\binom{k-1}{(k+1)/2}$. The last step here, the details of which we leave for the reader to fill in, is to improve the bound using the fact that if we pick some string x with exactly $(k-1)/2$ ones, we cannot also pick the complement of x (by which we mean the string where we keep the first position 0 and flip the remaining $k-1$ bits). Of course, figuring out the details of this is not necessary during the contest: coming up with the construction above, and confidently guessing that it is optimal, is sufficient to solve the problem.

This leads to the following simple algorithm: given n, find the smallest k such that $\binom{k-1}{\lceil k/2 \rceil} \geq n$, enumerate n different binary strings of length k with $\lceil k/2 \rceil$ ones (and starting with zeros), and use these as the schedule.

Problem C: Three Kinds of Dice

Problem authors:
Derek Kisman and Walter Guttmann

Solved by 30 teams.
First solved after 82 minutes.
Shortest team solution: 1303 bytes.
Shortest judge solution: 871 bytes.



Consider a face of die D_3, with value v. Let $S_i(v)$, for $i \in \{1, 2\}$, be the expected value of points die D_i gets if D_3 lands on this face. The expected value die D_1 gets overall is the average of $S_1(v)$ over all faces of D_3.

$S_1(v)$ is simply the number of faces of D_1 larger than v, plus half the number of faces equal to v. The same goes for $S_2(v)$. This means that there are $O(n)$ possible values for the pair $(S_1(v), S_2(v))$, and we can easily calculate all of them in $O(n \log n)$. We are now looking to