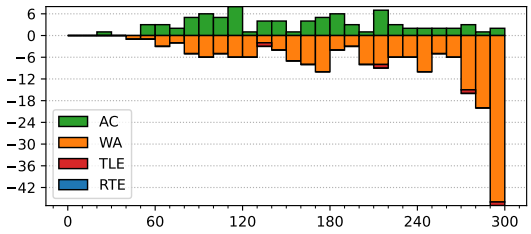
assign weights to those points so that the weighted average of $S_1(v)$ is ≤ 0.5, and the weighted average of $S_2(v)$ is as high as possible (and vice versa).

This is a geometry problem. Note that all the points we can obtain by weighted averages of a set of points is exactly their convex hull; so we're looking for the highest y-variable value in the convex hull intersected with $x \leq 0.5$, which can be determined in $O(n)$ in a number of ways.

Problem D: Carl's Vacation

Problem authors:
Arnav Sastry and the World Finals judges

Solved by 86 teams.
First solved after 21 minutes.
Shortest team solution: 1537 bytes.
Shortest judge solution: 893 bytes.



Clearly, the challenge in this problem is not about running time, but rather about how to represent the three-dimensional geometry problem in a convenient way.

On a plane, the shortest path between two points is always a line segment. So, the path will be a segment from the top of one pyramid to its base, then a segment across the ground to the base of the other pyramid, and then a segment from the base to the top of that pyramid.

We have 16 ways of selecting the faces of the two pyramids we will travel on, so we will check all of them. Now, we have two tilted triangles, each attached to the ground by the base, and we need to get from the top of one triangle to the top of the other using as short of a path as possible. The answer will be the same if we rotate each triangle around its base and flatten it out — so we now just need to find the shortest path between two points on the plane, under the condition that it passes through two specified segments. The shortest path between two points on a plane is just a segment. So, to summarize:

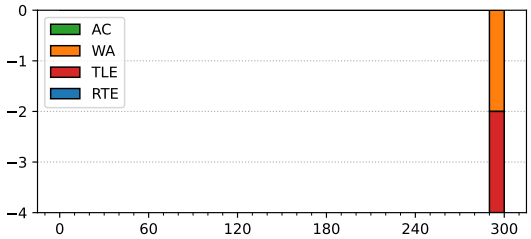
- We iterate over all selections of the faces on both pyramids.
- We rotate the top of the pyramid around the line containing the base of the selected face so that the top of the pyramid lands on the ground.
- We check if the segment between the two rotated tops intersects the two bases, in the right order.
- If yes, we take the distance between the rotated tops as a candidate distance.
- We output the smallest candidate distance.

Another approach is to observe that, for a pair of pyramid faces, we can parameterize the path taken by the ant by two angles θ_1 and θ_2. The minimum of $dist(\theta_1, \theta_2)$ can be found using golden section or ternary search.

Problem E: A Recurring Problem

Problem authors:
Derek Kisman and Matthias Ruhl

Solved by 0 teams.
Shortest judge solution: 2987 bytes.



This was, in the judges' opinion, the hardest problem in this set. It can be solved using fairly complicated dynamic programming.

You can start by thinking about how to get the first few elements of the generated sequence. The first one is actually quite easy: the number of sequences that start with t is the number of ways of writing t as $\sum_{i=1}^{k} c_i a_i$ for some positive integers k, c_i, a_i.

Figuring out the extensions of this is considerably trickier, because now the c_i's and a_i's interact with each other. In general, suppose you want to count how many ways an order-k sequence can start with $(a_{k+1}, a_{k+2}, \ldots, a_{k+\ell})$. If you know the values of c_k and a_k, then you can essentially subtract the contribution of these and try to continue from there; but you still need to remember the original values of $a_k, \ldots, a_{k+\ell-2}$ because these will be make a contribution multiplied by c_{k-1}.

After some thought, a useful function to compute is the following. For a vector $x = (x_1, \ldots, x_\ell)$ of length ℓ and a vector $a = (a_1, \ldots, a_{\ell-1})$ of length $\ell - 1$, define $f(x, a)$ to be the number of possible choices of r and positive integers $a_0, a_{-1}, \ldots, a_{-(r-1)}$ and $c_1, \ldots, c_r$ such that, for all $1 \leq i \leq \ell$ it holds that

$$x_i = \sum_{j=1}^{r} a_{-r+i+j-1} \cdot c_j$$

Note that, if $a = (x_1, \ldots, x_{\ell-1})$, then $f(x, a)$ counts exactly how many generated sequences start with x. Furthermore, we can define a nice recurrence for f:

$$f(x,a) = \begin{cases} 1 & \text{if } x \text{ is identically } 0 \\ 0 & \text{if } x \text{ has any negative entries} \\ \sum_{a_0,c} f(x - c \cdot (a_0, a_1, \ldots, a_{\ell-1}), (a_0, \ldots, a_{\ell-2})) & \text{otherwise} \end{cases}$$

In the sum, a_0 and c range over "all" positive integers, but can clearly be limited to those such that $x - c \cdot (a_0, \ldots, a_{\ell-1})$ is a non-negative vector.

Ok, so implementing this function f (and adding memoization), we have a way of counting the number of sequences with a given start, but this is still very slow, because as written there are lots of dead-end paths explored, that never lead to us reaching the all-zero vector (and therefore do not contribute anything to the count).

From here, there are at least two approaches for making the solution fast enough:

1. One useful idea for computing f faster is that if $f(x, a) > 0$ then $f(x', a') > 0$ where x' and a' are the vectors of length $\ell - 1$ and $\ell - 2$ where we remove the last element. So when computing $f(x, a)$, we can add a pruning step which first computes $f(x', a')$ and checks that this is positive. This improves the running time dramatically and makes the solution fast enough.

   Furthermore, we can relatively easily change f so that we instead count, given $y = (y_1, \ldots, y_\ell)$, the number of generated sequences with a prefix of the form $(y_1, \ldots, y_{\ell-1}, \leq y_\ell)$. With that in hand we can binary search for the next extension.

   One last idea here is that, once we have honed in on a prefix where there are relatively few (say, less than 500 000 different sequences generating that prefix, we can create all of those sequences and then iteratively evolve them until we find the right one.
2. A cleaner idea is to change the function f to not only count the number of sequences with the given prefix, but also all possible extensions, together with their counts. That way, there is no need to binary search for the next entry and we can generate the prefix one step at a time until we are at the right one.