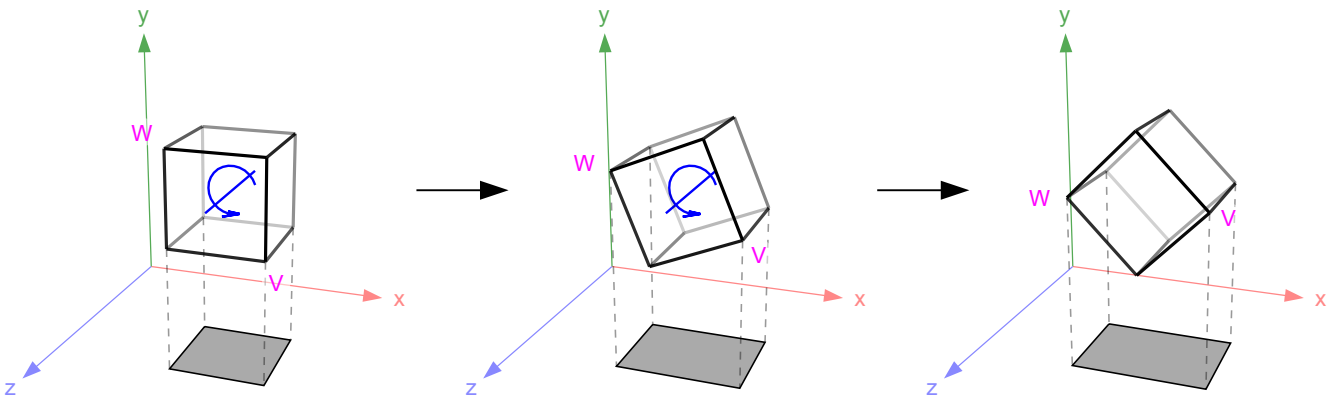
Code Jam 2018 - Qualification Round

Analysis: Cubic UFO

This is an ad hoc geometry problem with many different solutions.

Test Set 1 (Visible)

Suppose the cube is initially axis-aligned. Let us rotate it about the z-axis by angle t, from +x towards +y, and study the shadow:



Key observations:

- The shadow is a rectangle aligned to x- and z-axes, starting out as a square for t = 0.
- For $0 \le t \le \pi/4$ (45 degrees): z-length = 1 always, and x-length = $V_x - W_x$, where V_x and W_x are x-components of vertices V and W in the figure. Therefore this is really a 2-D problem; we can ignore z!
- The shadow area is **A** = $1 \times (V_x - W_x) = 2\,V_x$, since $W_x = -V_x$.

For this setup, maximal area is attached when $t = \pi/4$, which corresponds to $V_x = \sqrt{(½)}$, resulting in **A** = $\sqrt{2} \approx 1.414214$. This exceeds all Test Set 1 inputs, so the setup is sound.

Next, we find the shadow area **A** as a function of angle t. From basic geometry, $V_x = \sqrt{(½)} \times \cos(t - \pi/4)$. Therefore **A** = $2V_x = \sqrt{2} \times \cos(t - \pi/4)$, for $0 \le t \le \pi/4$.

Given **A**, naively we'd invert the formula and get t as sum of $\pi/4$ and $\cos^{-1}(\mathbf{A} / \sqrt{2})$. However, to satisfy $0 \le t \le \pi/4$, we need the negative branch of $\cos^{-1}$! Therefore the inverse is:

$t = \pi/4 - |\cos^{-1}(\mathbf{A} / \sqrt{2})|$.

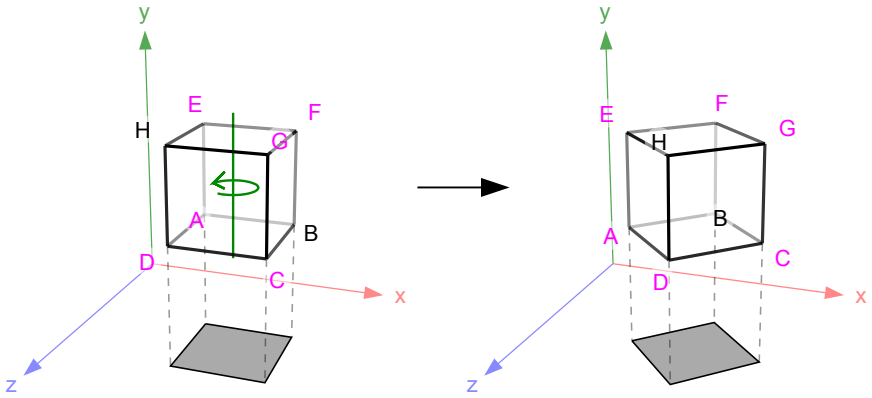
Once t is obtained, the final outputs are the centers of three non-parallel faces. One such face (invariant for all t) is (0, 0, ½). The other two can be obtained from rotating (½, 0), and (0, ½) by angle t, and assigning z to 0. Using the rotation formula yields (½ cos(t), ½ sin(t), 0) and (−½ sin(t), ½ cos(t), 0).

Test Set 2 (Hidden)

Our solution hinges on the following two crucial observations:

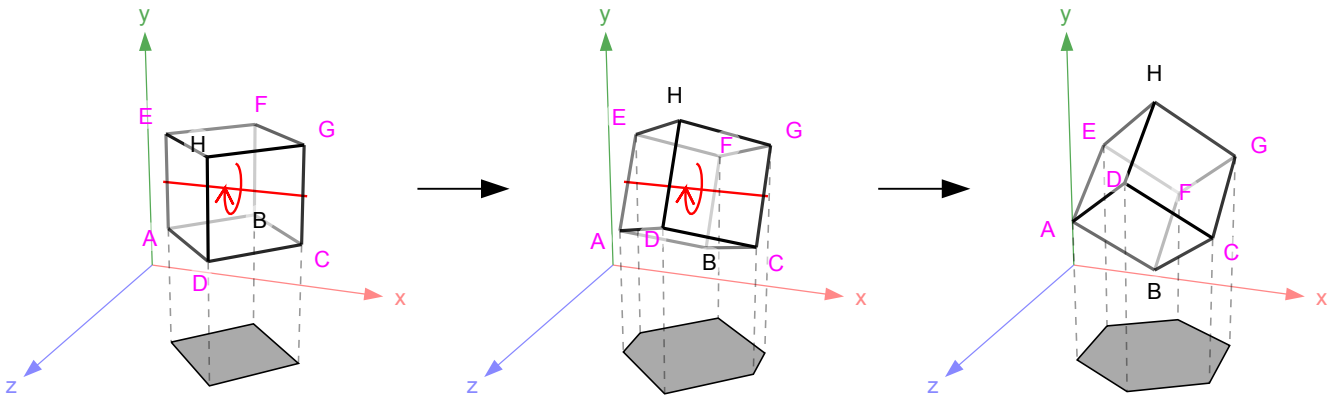
- The cube will cast the smallest possible shadow, which has a square shape, when one of its faces is parallel to the xz-plane.
- The cube will cast the largest possible shadow, which has the shape of a regular hexagon, when one of its vertices is on the y-axis.

To simplify the computations, let's rotate the cube about the y-axis by 45 degrees. (The direction of the rotation does not matter, since the cube would end up in the same orientation either way.) After that, the cube will look like this:



It might not be immediately clear why this simplifies our life, but it will make sense soon!

According to the above observations, the cube currently has the smallest possible shadow. To maximize that shadow, we can rotate the cube about the x-axis from +z towards +y, and bring the vertex H (from our diagram above) onto the y-axis. A useful property of this rotation is that the area of the shadow consistently increases throughout this rotation. Since we start with the smallest possible shadow and continuously rotate until we get the largest possible shadow, we achieve every possible shadow area at some point during this rotation. So, we can use binary search to figure out the exact angle by which we need to rotate the cube about the x-axis to achieve the desired area.



However, two questions remain:

- If we rotate the cube about the x-axis by a certain angle, what will be the coordinates of the vertices of the cube?
- Given the coordinates of the vertices, how can we calculate the area of the shadow?

Rotating a cube about the x-axis

Notice that, since we are rotating the cube about the x-axis, the x-coordinates of the points will remain the same; only the y- and z- coordinates will change. So, instead of performing rotations in 3-D, we will project the point onto the yz-plane (the x = 0 plane) and perform the rotations in 2-D.

For example, suppose that we want to rotate point P = (P_x, P_y, P_z) about the x-axis from +z towards +x by angle t. First, we project the point onto the yz-plane, where it will have coordinates (0, P_y, P_z). We will ignore the x component and treat the point as (P_y, P_z) on a 2-D plane.

Now, rotation about the x-axis by angle t in the indicated direction is equivalent to rotating (P_y, P_z) about (0, 0) by angle t in a clockwise direction. The resulting 2-D point will be

$(P_y', P_z') = (P_y \times \cos(t) + P_z \times \sin(t), -P_y \times \sin(t) + P_z \times \cos(t))$,

which in 3-D becomes (P_x, P_y', P_z'). We can get this from the rotation formula, or the complex expression $(P_y + iP_z) \times e^{-it}$.

Shadow area

As H approaches the y-axis, the shadow on the y = −3 plane takes the shape of a convex hexagon. More specifically, the vertices of the hexagon are the projection of points D, C, G, F, E, and A onto the y = −3 plane. For a point P with coordinates (P_x, P_y, P_z), the coordinates of its projection onto the y = −3 plane are (P_x, −3, P_z).

Now, to find the area of the shadow, we can treat the six projected vertices as if they were on a 2-D plane, with only their x- and z-coordinates. By construction, these already form a convex hull with vertices properly oriented (otherwise one would need to explicitly compute the convex hull, although shortcuts exist for the special case). This enables us to apply the standard formula to compute area of a convex polygon. Note that area computation can also be simplified by using symmetry with respect to the z-axis, and apply the trapezoid area formula instead of the general convex polygon.

Once we have a set of coordinates that produces the desired area, we can compute the coordinates of the face-centers of any three non-pairwise-parallel faces, and we have solved the problem. With the setup above, these are simply ½(A + C), ½(C + F), and ½(F + A).

Other approaches

There are many other ways to solve this problem:

- Instead of binary searching, the angle of rotation can be solved directly. In fact, final coordinates can be computed using a closed form without using trig functions!
- We could avoid computing the area of the shadow by using this amazing cube shadow theorem.
- Instead of doing two rotations (one about the y-axis and another about the x-axis), we could rotate the cube about line connecting points (0, 0, 0) and (1, 0, 1), or some other similar axis, to bring a vertex onto the y-axis. The general rotation formula can be found here.