**DIFFICULTY**

MEDIUM

**PREREQUISITES**

Binary Search, Greedy Algorithm, Geometry

**PROBLEM**

You are given several light sources along a road (road is x-axis). Each light source casts isosceles triangles with their base over the road. We wish to find the highest height at which we can fly a plane from x = L to x = R such that line segment from (L, h) to (R, h) is completely within the light - that is, within one or more of the isosceles triangles cast by the light sources.

**QUICK EXPLANATION**

A useful technique while solving any problem is to identify that the solution w.r.t. one of the variables is monotonic. In this problem there is some height H, above which we cannot find a segment (L, h) to (R, h) satisfying the constraint. But, below this H, for all h, (L, h) to (R, h) will always be in light.

Thus, we can perform a binary search on h to find the largest height which is **valid**.

**EXPLANATION**

If the segment from (L, h) to (R, h) is completely within the light triangles, we call h **valid**.

To check if h is valid we can do the following

function verify(h)

Let A be a list of items, where item is a pair of numbers

for each source of light, s

if height[s] < h then ignore this source

else

base = ((Y[s] - h) \* tan Z[s])

lx = X[s] - base

rx = X[s] + base

A.push(lx, 1)

A.push(rx, -1)

A.push(L, 0)

A.push(R, 0)

So, for each light source, we find the intersection of the light source with a line at height h.

We store these intersection points in a list along with "1" for the points at which a light source is added, and "-1" for the points at which a light source is ended.

Note that we also add L and R to A, but without any delta (that is their pair value is 0).

We know that now the number of light sources illuminating each segment in A are constant. By adding L and R to A, we have ensured that we consider the line segments that start / end at L (or R) separately.

The following snippet shows the complete **verify** procedure.

function verify(h)

Let A be a list of items, where item is a pair of numbers

for each source of light, s

if height[s] < h then ignore this source

else

base = ((Y[s] - h) \* tan Z[s])

lx = X[s] - base

rx = X[s] + base

A.push(lx, 1)

A.push(rx, -1)

A.push(L, 0)

A.push(R, 0)

sort A

Let cnt = 0 be number of active sources of light for the next segment

Let la = -infinity be the endpoint of the last seen segment

for i = 0 to A.size

Let x = A[i].first

Let del = A[i].second

if x > L and

x <= R and

x > la and

cnt == 0

then return "Invalid"

la = x

cnt += del

return "Valid"

We sort A to consider each line segment one by one. This is equivalent to a line sweep algorithm from left to right touching all the intersection points (that we stored in A). For each segment we see if x lies within the line segment. Now, if the line segment from la to x was of positive length, then we know that there is some (or all) of the line segment between la to x which belongs to the region L to R.

If cnt for that segment was 0, then that segment was not lit and we return "Invalid".

Otherwise we update the cnt and la for the next iteration.

If we find that no such dark segment exists, we return "Valid".

Hence we can call **verify** from within a binary search to find the largest possible h.

You need to be careful about precision errors and angle format conversions (degree / radians) for the respective language's math library.