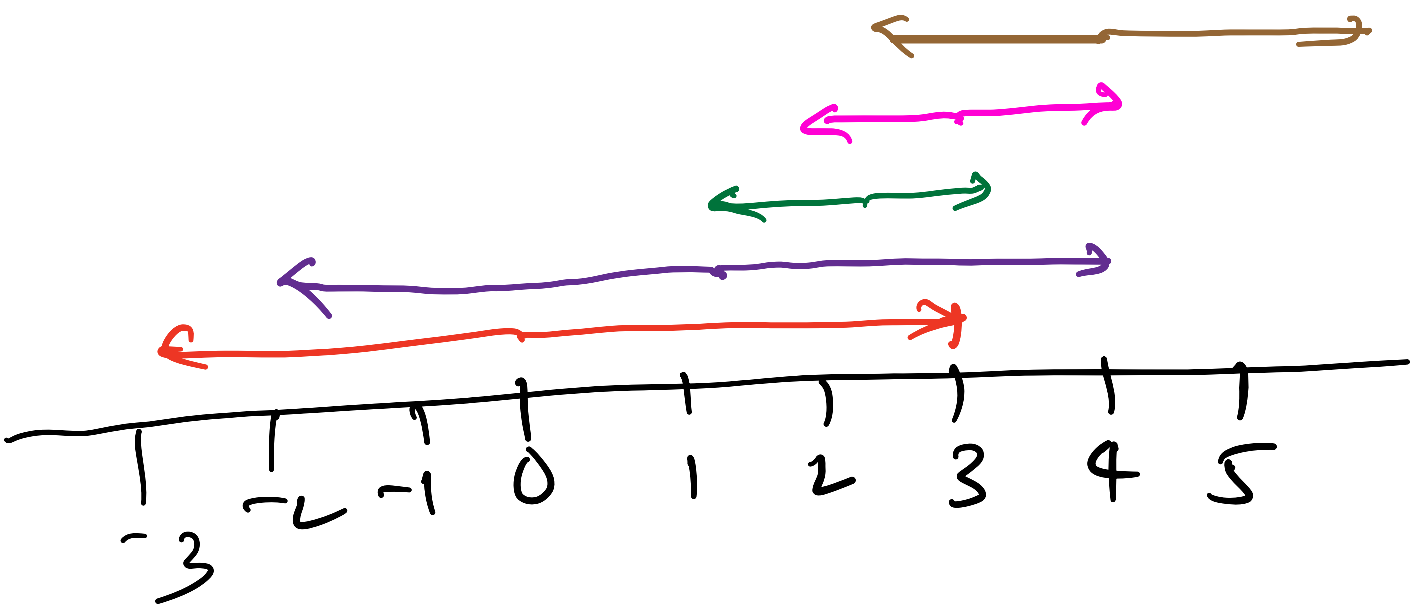
LC 1326

Minimum # of taps to open to water a garden

Intuition

Consider the below example

[3, 3, 1, 1, 2, 0]



From the above diagram we can think that tap purple opened at '0'th position covers maximum.

So, at '0' we greedily pick the tap that covers maximum range.

once we reach point '4'. It is evident that we have inspected the garden and all the points are covered.

However, After reaching '4' we don't have any tap to open to further inspect the remaining area.

But, if we look closely, the tap Brown had already been opened at '2' covers till 5.

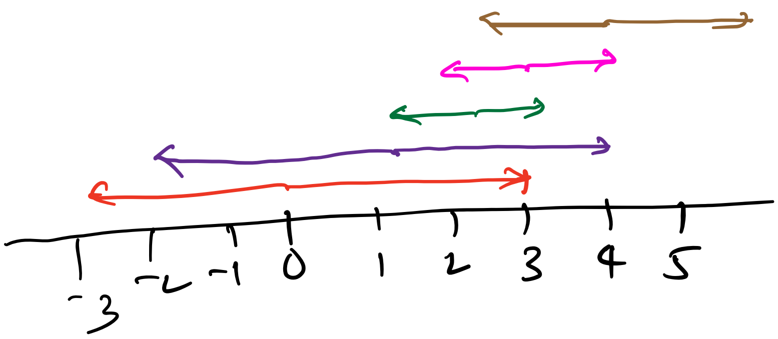
* So, the point is, while inspecting current wet area. if we encounter any tap that covers

beyond our maximum area under inspection, we can safely say that at some point, another tap opens which will not block our inspection once the area under inspection is complete.

So as long as we encounter a maximum range that goes beyond our range of inspection we can safely increment tap count and update area of inspection.

Algorithm

[3, 3, 1, 1, 2, 0]



Left

min (0, i - range[i])

Right

max (n, i + range[i])

i	left	right
0	0	3
1	0	4
2	1	3
3	2	4
4	2	5
5	0	5

* Better ranges

for every left, if it has maximum right not that in can array.

~~3~~ 4 3 ~~4~~ 5

start_end = [$\cancel{0}$, $\cancel{0}$, $\cancel{0}$, 0, 0, 0]

put all Zeros initially.

iterate through 'n'

n = 0 ;

left = 0 , right = 3

max(0, 3) = 3

start_end [left] = 3

n = 1

left = 0 , right = 4

max (3, 4) = 4

Similarly fill the remaining.

The point is, at each start,

if we find a betterend,

update start end [start] = better one.

= max (startend [start]

, currentend

)

start_end = [4, 3, 5, 0, 0, 0]

Filling start_end array is first part of the problem.
at any point index of start_end gives start range and it's corresponding start_end[i] gives end range.

Part - II

initialize max_range = 0
allowable range = 0
taps = 0

Traverse the range 0, N+1

if the inspector goes to a point where no taps are currently open & no taps were opened during the inspection that probably give them path forward

Cond 1

then we can say that no matter how many taps we open we can't water whole garden.

i > max_range.

return -1.

Cond 2

during traversal,

If we encounter a range i.e greater than current range, safely open the tap and update the current_range → new_allowable range.

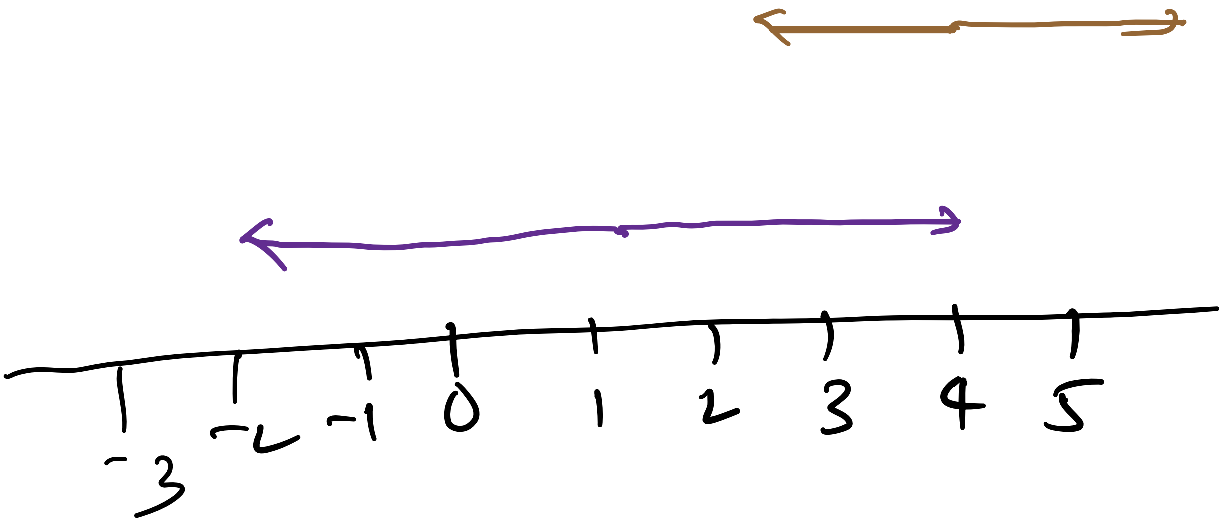
i > allowable_range.

taps++

allowable_range = max_range.

Finally,

max_range = max(max_range, StartEnd(i))



From the simplified image,

@0
AR=0
MR=0
J=0

@1
AR=4
MR=4
J=1

@2
AR=4
MR=5 (Jump extended at 2

but we don't increment jump yet.

We increment at AR's end

@ 3

AR = 4

MR = 5

T = 1

@ 4

AR = 4

MR = 5

T = 1

@ 5

i> AR but MR we do have

so taps ++

AR = 5

MR = 5

} and so on.