Minimum Number of Stops :-

#LeetCode/Medium

an airplane
limit of jump = k

airport → [1 0 1 1 0 1 1 ...]
1 is landable
0 is not.

find minimum possible Jump.

#algorithm/greedy

greedy → Jump maximum then come back to find valid landing position.
if the starting position reached that means airplane can't reach its destination.

unfaithful greedy.

Greedy is not same as Grotesque....

Bag of Tokens

power = [100, 200, 400] budget Power = 150
initial score = 0

able to decrease score & take any power
able to take any power under budget power to increase score.

approach :- sort (powers).

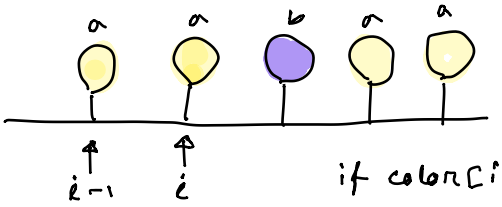
[100, 100, 200, 300, 500]

waste power to buy token from the front

waste score to buy power from the end.
if (score > 0)

Make Rope Colorful :-

color = "aabaa" time needed = [1,2,3,4,1]
find minimum time to get no consequitive element same color.



if color[i-1] != color[i] : Prev = 0

can be simpler & line....

```
{ ans += min(prev, timeneeded[i]);
  prev = max(prev, timeneeded[i]);
```

Break a palindrome :-

catch : have to be lexicographically smallest possible.

"a b c b a" → "a a c b a"

"a a a a" → "a a a b"

approach :- alter first non a to a one; if it's all a change last to b.

Circular Gas Station ✓ return the starting station (unique)

gas = [1 2 3 4 5]
cost = [3 4 5 1 2]

cost[i] = cost to go from i to i+1 gas station.

```
cst = 0 , ans = 0 ;
for ( int i = 0; i < gas.size(); i++){
    cst = cst + gas[i] - cost[i]
    if (cst < 0 ){  cst = 0;
                    ans = i+1 ;}
```

First check if Σgas >= Σcost

Dry Run

gas	1	2	3	4	5
cost	3	4	5	1	2
	0+1-3	0+2-4	0+3-5	0+4-1	3+5-2
cst = 0	-2→0	-2→0	-2→0	3	6
ans = 0	2	3	4	4	4 ✓

Minimum no. of Taps to water the Garden

https://leetcode.com/problems/minimum-number-of-taps-to-open-to-water-a-garden/description/

#LeetCode/Hard

#⭐, #Trick

#Revision

range of tap = [3 4 1 1 0 0]

trick :- generate the left side reach array

i =	0	1	2	3	4	5
Range:	3	4	1	1	0	0
left -	0	0	1	2	4	5
right -	3	5	3	4	4	5
leftreach:	5 ↙	3	4	0	4	5

left = max(0, i - range[i])
right = min(n, i + range[i])

leftreach[left] = max(leftreach[left], right)

now work will be done on this array here, leftreach[0] = 5 mean there exict a tap which starts from 0 & end at 5.

leftreach[1] = 3 means there exist a tap starting from 1 and goes to 3.

leftreach[3] = 0 means there's no tap starting from 3.

Algo :- maxreach=0, tap=0, curr=0

```
for (int i = 0 to n){
    if (i > maxreach) return -1        // not possible
    if (i > curr){ curr = maxreach;
                   tap++; }
    maxreach = max(maxreach, arr[i])
```

Dry Run 1

	i:	0	1	2	3	4	5
Array		5	3	4	0	4	5
maxreach	0	5	5	5	5	5	5
tap	0	0	1	1	1	1	1
curr	0	0	5	5	5	5	5

Dry Run 2 :

	i:	0	1	2	3	4	5
reach		3	3	1	0	0	1
left -		0	0	1	2	4	4
right -		3	4	3	4	4	5
leftReach :		4	3	4	0	5	0

	i:	0	1	2	3	4	5
maxreach	0	4	4	4	4	5	5
curr	0	0	4	4	4	4	5
tap	0	0	1	1	1	1	2

maxreach=0, tap=0, curr=0

```
for (int i = 0 to n){
    if (i > maxreach) return -1        // not possible
    if (i > curr){ curr = maxreach;
                   tap++; }
    maxreach = max(maxreach, arr[i])
```


→ Minimum Replacement to Sort Array

9, 7, 6 ≠ can be broken into 1, 6
2, 5
3, 4

3 3 3 3 4 6 ← this shall be taken

```
no of part = nums[i] / last
if (nums[i] % last ≠ 0) no of part++;
last = nums[i] / no of part
count += no.of parts - 1;
```

Earliest Possible Day of bloom :-

#LeetCode/Hard

plantTime = [2 3 9]

growTime = [2 8 3]

planting has to be done manually to one seed at a time growing can be automatic.

Trick; plant first which takes maximum time to grow ✓✓ dash of class logic.