__/__/____

BS on answers

(12) Find square root of an number
(return floor value if root not found)

- basically for a given number its sq root will be a number from 0-N any number between that
- so keep 0-N as range & find mid point
- if $mid^2 <= N$ it means this can be the floor value of $\sqrt{N}$ but since $Mid^2 <= N$ we have to check if we have elements that's $= N$ so go right half
- but if $mid^2 > N$ then look on the left half

(13) Find n^{th} root of a number

$\sqrt[n]{m}$

- for a given a number keep the range
- find mid point
- find mid^n
- if $mid^n = m$ then return mid
- else if $mid^n < m$ then look on right
- else $mid^n > m$ look on left
- if not found then return -1

__/__/____

(24) Find smallest divisor lesser than threshold

- for given array elements find the lowest divisor that is lesser than threshold value
- sum total sum of each element by the divisor should be <= threshold
- basically 1st set the range of the binary search low = 1 & high = max element in the array
  (cuz any element divided by greater element always results in 1)
- now find a midpoint
- & divide all array elements with the midpoint & also calculate the sum of each array element division

ex: {1 2 5 9} limit = 8

mid = 5

$\frac{1}{5} + \frac{2}{5} + \frac{5}{5} + \frac{9}{5} = 1 + 1 + 1 + 2$

$= 5 <= 8$ ✓

- now divisor 5 is <= 8 so divisor 5 can be our answer but we look for even smaller divisors i.e <= limit on the left half
- but if sum total is greater than limit we look on right half.

__/__/__

(15) Koko eating bananas

- Same logic as last one
- each element in array represents no of bananas
  ex {10, 20, 30}
  10 bananas
- koko has to complete all piles within time limit
- So we have to find how many banas to eat per hour to complete all piles within given time limit
- So start by eating 1 banana/hour — largest pile size
- Now find the divisor for total bananas that can be eaten per hour to finish all piles within given time

  ex: {10, 20, 30} limit = 12 hrs.

  ~~bananas in pile~~

  $\frac{10}{1} + \frac{20}{1} + \frac{30}{1} = 60$ hrs which is greater than 12

  banana per hour

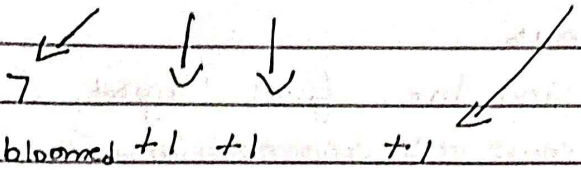
  : 2 - ~~dies~~ bananas per hour
  : 3 -
  : 4 -
  : 5 - takes exactly 12 hrs
  ~~also~~

- $\frac{10}{5} + \frac{20}{5} + \frac{30}{5} = 12$ So eating 5 bananas per hour is the answer

__/__/__

(16) M bouquets

- So basically each element in array represents a day
  ex: {7, 7, 7, 12, 11, 13, 7} M = 2 K = 3



  total 4 flowers bloomed on Day 7
- our task is to find min day where we can make 2 bouquets of 3 flowers i.e using 6 flowers

St 1- find min & max in array (to check from min days how many flowers will be bloomed till max day)

St 2- call a function by sending current day (d) to check how many flowers will be bloomed on current day & if we can make m bouquets with those bloomed no flowers.

St 3- if m number of bouquets can be made on that current day then we look for even smaller days where m bouquets can be made

St 4- and finally return the day where m bouquets can be made. Edge - 1.

__/__/__

(17) Aggressive cows

- So basically every element in array represents stalls
- You have k cows
- our task is find the first largest distance such that all cows are accomodated.

- first sort the array
- at min dist b/w two cows/stalls can be - 1
  at max it can be - (n-1) - a[0]
  highest - lowest
- So from 0 to n-1 we take mid point
- now we have to fill the stalls such that a distance of mid is to be maintained. b/w 2 cows
  ↓
  call function for this (Placable)

- here what happens is
  - we do currentplace - lastplace
    which gives distance b/w the two
  - then if that distance >= mid dist (the min distance to be maintained)
  - then cow will be placed in that currentplace
  - we move to next cow
  - & update lastplace

__/__/___

- now if total cows placed >= cows no
  (i.e if all cows have been placed with given min distance)
  - then return true.

- now in Binary search you look for the next larger element (or next distance) larger so look right
  - else look on left and store current mid as possible answer
- and return answer.