Similar to minimum no of pages to read:

Logic is reversed:

Here we are supposed to find the max distance at which cows can be placed within the given stalls

1. Low = 0(or any min val)
2. High = max(arr)
3. While low <= high:
   1. Mid = low + (high-low)/2
   2. If mid is a valid value: that is we can place cows at this distance apart then:
      1. Update res to mid
      2. Move towards right to find value larger than current(i.e maximize the distance)
   3. Otherwise, distance between the cows is too large, cannot place cows, so reduce distance
      1. Move towards left: right = mid-1
4. Return res

isValid(stalls, noOfStalls, noOfCows, maxDist):

Let cows = 1

For i in range(noOfStalls):

# find a position that is within maxDist from current position

If stall[i] - initialPosition >= maxDist:

Update no of cows(add 1)

Set initialPosition to currentPositon

If cows >= noOfCows:

Return True

Return False

Code:

def isValid(self, stalls,n,k,midMax):

cows = 1

curPosition = stalls[0]

for i in range(1, n):

if stalls[i]-curPosition >= midMax:

cows += 1

curPosition = stalls[i]

if cows >= k:

return True

return False

def solve(self,n,k,stalls):

left = 0

right = max(stalls)

res = -1

stalls = sorted(stalls)

while left<=right:

mid = left + (right-left)//2

if self.isValid(stalls,n,k,mid):

res = mid

# try to increase/maximize the distance between them

left = mid + 1

else:

# decrease distance between them

right = mid -1

return res