Searching - 3

B.S on answer space
↳ Square Root
↳ A^{th} Magical Number
} Easy - Problem

July - 2021 ?
↳ Goldman ?

Question: Aggressive Cows

(2 ≤ K ≤ N)

→ N stalls placed on x - axis ✓
→ K cows (K ≥ 2 and K ≤ N) ✓
→ Place these K cows in N stalls
→ 1 stall → atmost 1 cow.

TASK: Place cows in stalls such that they are as far as possible

1 2 4 6 10

N = 5 , K = 3

	0	1	2	3	4
stalls =	1	2	4	6	10 ✓
	c_1	c_2	c_3		
	c_1			c_2	c_3

5 4

Min_dist

$\boxed{4}$

Maximise distance between any 2 cows

N = 9 , K = 4

	0	1	2	3	4	5	6	7	8
stalls =	2	6	11	14	19	25	30	39	43
	c_1	c_2	c_3	c_4					

Mindist

3

c_1 c_2 c_3 c_4

Best Arrange

c_1 c_2 c_3 c_4 $\boxed{12}$

c_1 c_2

→ Maximise the minimum distance between any 2 cow

$\boxed{Ans = 12}$

13, 14, 15 . . . .

$(100-1)/k-1 =$

N = 6, $\boxed{K=3}$

1 2 3 4 5 100

Min

$\boxed{2}$

c_1 c_2 c_3

$\boxed{100/3 = 33}$

$N = 10^5$

K = 2

Brute Force:

Try all the Possibilities ⇒

$\boxed{K=3}$

prev = ~~3~~ 11

stalls [ ] → [-1, c_1, -1, c_2, -1, c_3] N stalls

1, 3, 7, 11, 20, 30

11 - 3

$\boxed{Back tracking}$ = $\boxed{{}^{N}C_{K}}$ → Combinations

$\frac{N!}{(N-K)!\,K!}$

$\boxed{T.C: \binom{N}{K} \times O(N)}$

Exponential.

$\boxed{N = 10^5}$

$\boxed{90\%}$

Binary Search:

$\boxed{\text{Greedy DP}}$

Search]

Binary

* Maximum / Minimum =>

Search space

lowest possible

Highest possible

$\boxed{[1 \ldots N-1]}$

high =

N = 7, k = 7

k = 3, 4, 5

$\boxed{k = 2}$ spc

stalls = 1 3 5 6 10 11 17

c1 c2 c3 c4 c5 c6 c7

2 2 1 2 1 6

(max - min) = $\boxed{16}$

MAX = (max - min)

k = 6

1 2 3 4 5 6

c1 c2 c3 c4 c5 c6

low = 1

Search space:

low = 1

high = stalls[n-1] - stalls[0]

Simpler Problem:

Check if we can place k cows where any 2 cows are placed atleast D distance apart

$\boxed{5 \text{ cows}}$

k = 4, D = $\boxed{8 \text{ units}}$

stalls = 2 6 11 14 19 25 30 39 43

c4 c5

Stalls:

c_1 c_2 c_3 c_4

Can I place 4 cows such that every 2 cows are atleast 7 units apart?

6, 5, 4, 3, 2, 1

K = 4, $\boxed{D = 15}$? 2

Stalls = 2 6 11 14 19 25 30 39 43

c_1 c_2 c_3

Can we place cows such that any 2 cows are 16 units apart?

16, 17, 18 ...

```
bool check(Stalls[], N, K, D) {
    recent_placed = stalls[0];
    cows_placed = 1

    for(i = 1; i < N; i++) {
        if( stalls[i] - recent_placed >= D ) {
            recent_placed = stalls[i];
            cows_placed ++;
            if(cows_placed == K) return true
        }
    }
    if( cows_placed >= K) return true
    else false;
}
```

T.C: $\boxed{O(n)}$

K = 3

stalls = 1 2 6 11 17 20

Stalls -

c_1 c_2 c_3 17

c_1 c_2 c_3 9

$\{x_1, x_2 ..\}$ $K \to \boxed{O(K)}$ $({}^{N}C_K) \times O(K)$

$K = 2,$ $D = 17$

c_1

$C_1 =$ [1 2 3 4 5 6 7 8 20]

cows_placed = 2

O(N)

O(n)

$\boxed{2^K}$

c_1 c_2 c_3

0 1 2 3 4 5

$\boxed{N = 6, K = 3}$

$\boxed{{}^{N}C_K} \times \boxed{O(K)}$

{1, 3, 5} => O(K)

(max-min)

K = 4

Stalls = 2 6 11 14 19 25 30 39 43

c_1 c_2 c_3 c_4

Ans = 12.

Search space: [1, 43-2] => [1, 41]

Go Right

mid. Right

Search Space: [1 2 3 . . 8 . . 9 10, 11, 12, 13, 14 . 20 . . . . 40, 41

Go left.

ans = 9

if check(mid) == true
ans = mid;
low = mid+1.

check (mid) == false
high = mid-1
left

mid

mid

12

40, 41

Search Space : 1 2 3 4 . . . . 11, 12, 13, 14, 15 . . .

check(D)
check(mid) = = true {
ans = mid
low = mid+1
}

check(mid) = = false
high = mid-1

int Aggressive Cows (Stalls [ ], k) {

low = 1
high = stalls [n-1] - stalls [0];

while (low <= high) {

mid = (high + low) / 2;

if (check(mid) = = true) {
ans = mid;
low = mid+1;
}

else if (check(mid) = = false) {
high = mid - 1;
}

}

Hard

O(n)

return ans;
}

Sorted

logn.

log N

Search : [0, n-1]

Search Space : [1, stalls[n-1] - stalls[0]]

Size of Search Space :

size = O(stalls[n-1] - stall[0])

Iterations in B-S => Log (stalls [n-1] - stalls [0]) x O(n)

... stalls[0]];

T.C: o(n . log(s tals ...

Question : Painters Partition Problem

1) → there are N boards each with a length
2) → We have K painters
3) → A painter can paint only continuous boards
4) → Painting 1 unit of boards takes 1s
→ Find the minimum time to get all the boards painted

[Not sorted]

N = 4, K = 2.

	B1	B2	B3	B4	maxTime
	10	20	30	40	
	P1	P2	P2	P2.	90
	P1	P1	P2	P2	70
	P1	P1	P1	P2.	$\boxed{60}$

P1: 10(s)
P2: 20+30+40 = 90(s)

P1: 30
P2: 70

P1: 60
P2: 40

<60

50 . . , 58, 59,

N = 4, K = 4 K = 1

10	20	30	$\boxed{40}$
P1	P2	P3	P4

10(s)?

Search Space:

high : sum(boards)

low : max(boards) →

[0, 1]

min(board)

N = 4, K = 4

10 20 30 40

P1 P2 P3 P4

P1 : 10
P2 : 20
P3 : 30
$\boxed{P4 : 40}$

<u>Simpler Question</u> :

Given N boards and K painters, check if we can paint all the boards where each painter paints for almost T time.

N = 4, K = 2, T = 80

boards =	10	20	30	40
	P1	P1	P1	40

P1 = 10 + 20 + 30

P2 = 40

Can we paint all the boards with 2 painters where every painter paints for almost 90(s)

T = 81, 82, 83 8

N = 4, $\boxed{k = 2}$, $\boxed{T = 50}$?

		P1	P1	
boards =	10	20	30	40
	P1	P1	P2.	P3

$\forall$ T < 50

$\boxed{T = 49, 48, 47 \ldots}$ $\boxed{k = 2}$

P1 = 10 + 20

P2 = 30

P3 = 40

... with 2 painters when

Can we paint all boards ... every painter paints for atmost 40(s)

$N = 4, \quad K = 2$

$\boxed{Ans = 60}$

boards = 10 20 30 40

[40, 100]

low = 40

high = 100

ans = mid
high = mid-1;

mid → Right

40 41 42 . . . . 58, 59, 60, 61, 62 . . . . . . 99, 99, 100

check(mid) = False.

```
int paintersPartition (boards[], N, K){
    low = max(boards);
    high = sum(boards);
    while (low ≤ high){
        mid = (high + low)/2;
        if(check(mid) == true){
            ans = mid;
            high = mid-1;
        }
        else {
            low = mid+1;
        }
    }
    return Ans;
}
```

O(N)

SS: [max(boards), sum(boards)]

O(N × log(sum(boards) - max(boards)))

$N = 4, \quad K = 2$

$\boxed{T = 70}$

check

20 30 40

boards = 10 20 ...
P1 P1 P1

low = [40, 100]

check(70) = true
check(54) =

low	high	mid	Action
40	100	70	ans = 70, high = mid - 1
40	69	54	low = mid + 1
55	69	62	check(62) = true, ans = 62, high = mid - 1
55	61	58	check(58) = false, low = mid + 1
59	61	60	check(60) = true, ans = 60, high = mid - 1
59	59	59	check(59) = false, low = mid + 1
60	59		

$\boxed{ans = 60}$

a, b

$$\boxed{a \times b = gcd(a, b) \times lcm(a, b)} \rightarrow \text{Resources}$$

$$lcm(a, b) = \frac{a \times b}{\boxed{gcd(a, b)}} \Rightarrow O(\log(n))$$

Euclidean
↓
Maths: gcd
↓
Euc

Maximum / Minimum

log(a, b)

→ Whenever we want to find max/min, in most of the case there will be a solution using Binary Search it is the cannot guarantee ...

→ all ... optimal sol...
most

check funt

low, high)

check(mid) → DP / Greedy

Euclidean

Research Paper

max, min ⇒ Greedy, DP

Binary Search

Greedy, DP

$(a, b)_2$ ⇒ Euclidean Asked

^nC_k, ^nP_k

A.P

^nC_k, ^nP_k

30-40%

Extra Search

N = 3

2^N subsets

A = {1, 2, 3}

N = {1} {2} {3} {1,2} {1,3} {2,3} { } {1,2,3}

8 subsets

{(2) (2) 2 ...} 2 .. 2 . 2 . . . 2 2^N

N = 5

1 3 5 10 12 = 2^5

Very Very Compl

Recursion / Back tracking

Recursion Call Stack

k

Iteration

Stack

^nC_k, ^nP_k

30–45

Back track =>

All possibilities => Back track

Brute

K = 2

True

K = 3 4, ...

K = 2

N = 9

Stalls = 2 6 11 14 19 25 30 49 45

c_1 c_2 c_3 c_4 c_5 c_6 c_7 c_8 c_9

high = Stalls[n-1] – Stalls[0]

45–2 = 41

< 41

low = Min difference between any 2 consecutive elements

O(n) > (max - min)

low = 2

N = 4, K = 2,

3

low = 1

A = 100 101 102 103

low = 100

low = 1

[1 2 3 4 5 6]

F... Cs(...)] + O(N)

T.C: O(N

Recursion.: Can we break a bigger problem into smaller problem

O(N)

N = 8, K = 4

Recursion

minDist = 3

	0	1	2	3	4	5	6	7	8
stalls =	2	6	8	14	19	25	30	39	43
	C1	C2	C2.	C3	C4				
	C1		C2.	C3	C4				

Recursion / B

Exponential

Question: Aggressive cows

```
bool check(int stalls[], int N, int k, int minDist){
      int recent_placed = stalls[0];
      cows_placed = 1;

      for(int i=1;i<N;i++){
            if(stalls[i] - last_placed >= minDist){
                  recent_placed = stalls[i];
                  cows_placed++;
            }
      }
      if(cows_placed >= k)
            return True;
      else
            return False;
}
```

```
int AggresiveCows(int stalls[], int n, int k){
      low = 1, high = stalls[n-1]-stalls[0];
      int ans = 0;
      while(low <= high){
            mid = (high + low) / 2;

            if(check(stalls, n, k, mid) == true){
                  ans = mid;
                  low = mid + 1;
            }
            else
                  high = mid - 1;
      }
return ans;
}
```

Question: Painter's Partition problem

```
bool check(int boards[], int n, int k, int maxTime){
      int painters = 1;
      int time = 0;

      for(int i=0;i<N;i++){
            time = time + boards[i];

            // Recently added board cross max time
            if(time > maxTime){
                  // Assign it to next painter
                  time = boards[i];
                  painters++;
            }
      }

      if(painters > k)
            return False;
      return True;
}
```

```
int paintersPartition(int boards[], int n, int k){
      lo = max(boards), high = sum(boards);
      int ans;

      while(low <= high){
            mid = (high + low)/2;
            if(check(boards, n, k, mid) == true){
                  ans = mid;
                  high = mid - 1;
            }
            else
                  low = mid + 1
      }
return ans;
}
```