58) Given N tasks, k workers & time taken for each task find min time in which we can complete all tasks.

No Sorting

Note1: A single worker can only do continous set of tasks

Note2: All workers start their assigned tasks at same time

Note3: A task can only be assigned to a single person

Ex: N = 15, k = 3

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 3 | 5 | 1 | 7 | 8 | 2 | 5 | 3 | 10 | 1 | 4 | 7 | 5 | 4 | 6 |

16    18    37 → max of all them → 37

0-5 : 26    6-10 : 23    11-14 : 22 → 26

0-4 : 24    5-10 : 25    11-14 : 22 → 25 min time

Ex: k = 2

| 0 | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 1 | 1 | 6 |

→ ans = 6 ✓

Ex: k = 2

1 1 1 1 100 → avg = 104/2 → 52 ✓

100

→ ans = 100

k = 2

101 102 103 | 104 105

$P_1$    $P_2$

Ex: k = 1

| 0 | 1 | 2 | 3 |
|---|---|---|---|
| 3 | 6 | 2 | 8 |

19

k = 10 : 1 1 1 1 1 1 1 1 | 3 5

$P_1$    $P_2$

Ex:

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| N=15 : | 3 | 5 | 1 | 7 | 8 | 2 | 5 | 3 | 10 | 1 | 4 | 7 | 5 | 4 | 6 |

$\underbrace{3\ 5\ 1\ 7}_{16}\ \underbrace{8\ 2\ 5\ 3}_{18}\ \underbrace{10\ 1\ 4\ 7}_{22}\ \underbrace{5\ 4\ 6}_{15}$ } Time 22

k= 4

[0-3 : 16] [4-6 : 15] [7-9 : 14] [10-12 : 16] Task rem

Can we finish : 22 23 24 25 . . . ———→

l-randomly gave a number T

~~11~~ ~~12~~ ~~13~~ ~~14~~ ~~15~~ ~~16~~ : Can we finish in k=4 ?

## Binary Search

1) Target : mintime

2) Search : [max(arr) Totalsum]

- sum/k
- 1
- min
- max

3) Divide or not ?

| lo | hi | mid | |
|---|---|---|---|
| 10 | 71 | 40 | ans = 40, left h = mid-1 |
| 10 | 39 | 24 | ans = 24, left h = mid-1 |
| 10 | 23 | 16 | goto right l = mid+1 |
| 17 | 23 | 20 ? | goto right l = mid+1 |
| 21 | 23 | 22 | ans = 22 goto left |
| 21 | 21 | 21 } | goto right l = mid+1 |
| 22 | 21 | {break} | |

ans = 22

// Pseudo Code

10:35 break

lo = (max(arr[])) hi = (sum of arr[])

ans = hi

```
while( l <= hi ){

        m = ( lo + hi )/2

        if( check( m, arr[], N, k ) )
            // Means we can finish in m time
            ans = m;
            h = m-1
        }
        else{
            l = m+1
        }
}
return ans;
```

given array, size, workers

$l = max(arr[])$

$hi = sum\ of\ arr[]$

n = nuber of Element n

which we are we are

doing n = [hi - l + 1]

TC: $\log_2^n * O(N)$

TC: $N \log_2^n$

SC: $O(1)$

```
// bool check( int time, int arr[], int N, int k){
    s=0, c=0
    i=0; i<N; i++){
        s = s + arr[i]
        if( s > time) {c++; s = arr[i]}
    }
    if( s != 0) { c++ }
    if( c <= k) {return True}
    else {return False}
}
```

TC: O(N)

SC: O(1)

Ex:

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
| N = 15 : | 3 | 5 | 1 | 7 | 8 | 2 | 5 | 3 | 10 | 1 | 4 | 7 | 5 | 4 | 6 |

time = 20

I am giving?

S = ~~8~~ ~~9~~ ~~24~~ 16    S = ~~8~~ ~~10~~ ~~15~~ ~~18~~    S = ~~10~~ ~~11~~ 15    S = ~~7~~ ~~12~~ 16 22

t1    t1    t1    t1

S = 6

Take leftout

(Not valid)

(return false)

t1

Ex:

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
| N = 15 : | 3 | 5 | 1 | 7 | 8 | 2 | 5 | 3 | 10 | 1 | 4 | 7 | 5 | 4 | 6 |

time = 30

t1    31    S = 5    t1    S = 15

assign to a worker: t1

t1

c = 3

3 < 4

→ N ⇒ $C \log_2 N$

$lo = \max()$    $hi = \text{sum}()$

$x = (hi - lo + 1)$

$\log_2 x$

28) Given N cows & M stalls, all M stalls are on x-axis at different locations. Place all N cows such a way min distance between any 2 cows is maximized

Note1: In a stall only 1 cow can be present

Note2: All cows have to be placed. {stall[] sorted}

Ex1:

| | 0 | 1 | 2 | 3 | 4 | min |
|---|---|---|---|---|---|---|
| Stalls = 5 | 1 | 2 | 4 | 8 | 9 | |

Cows = 3

$C_1 \overset{1}{—} C_2 \overset{2}{—} C_3$ : 1

$C_1 \overset{3}{————} C_2 \overset{5}{————} C_3$ : 3

$C_1 \underset{7}{————————} C_2 \underset{1}{—} C_3$ : 1

ans = 3

Ex2:

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | min |
|---|---|---|---|---|---|---|---|---|---|---|
| Stalls = 9 | 2 | 6 | 11 | 14 | 19 | 25 | 30 | 39 | 43 | |

Cows = 4

$C_1 \overset{4}{—} C_2 \overset{5}{—} C_3 \overset{3}{—} C_4$ 3

$C_1 \overset{9}{———} C_2 \overset{14}{—————} C_3 \overset{18}{—————} C_4$ 9

$C_1 \overset{12}{—————} C_2 \overset{16}{—————} C_3 \overset{13}{———} C_4$ 12

Stalls = 9
cows = 4

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| 3 | 8 | 12 | 18 | 25 | 30 | 35 | 41 | 49 |

$C_1$ — 15 — $C_2$ — 17 — $C_3$ — 14 — $C_4$    mi

min → 9

Atleast 20 distance between 2 cows

* ~~21~~ ~~22~~ ~~23~~ ~~24~~ ~~25~~ . —

$\boxed{7 = 8}$

Atleast 8 distance between 2 cows ✓        ans = ______

4 ✓  5 ✓  6 ✓  7 ✓  8 ✓

Target : maximize

Search Space : [min adj diff , last - first]

Divide 2 half & discard

BS

| lo | hi | mid | |
|---|---|---|---|
| | | | goto left |
| 4 | 46 | 25 | h = mid-1 |
| 4 | 24 | 14 | ans = 14<br>l = mid+1 |
| 15 | 24 | 19 | h = mid-1 |
| 15 | 18 | 16 | h = mid-1 |
| 15 | 15 | 15 | h = mid-1 |
| 15 | 14 | {break} | |

Ex:

3 Stall

| | | |
|---|---|---|
| 0 | 1 | 2 |
| 3 | 4 | 8 |
| $C_1$ | | $C_2$ |

2 cows

ans = 5

4 Stall

4 Cows

| | | | |
|---|---|---|---|
| 0 | 1 | 2 | 3 |
| 3 | 9 | 11 | 18 |
| $C_1$ | $C_2$ | $C_3$ | $C_4$ |

6 2 7

ans = 2

12 - 3 = 9

25 - 12 = 13

Stalls = 9

cows = 4

dist = 8

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| 3 | 8 | 12 | 18 | 25 | 30 | 35 | 41 | 49 |
| $C_1$ | | $C_2$ | | $C_3$ | | $C_4$ | | |
| +1 | | +1 | | +1 | | +1 | | |

// Pseudo Code :

lo = ( min adj diff )    hi = ( last - first )    ans = —

$n = (hi - lo + 1)$

```
while ( lo <= hi) {
    mid = (lo+hi)/2
    if ( check(mid, ar[], N, C))
        ans = mid;  {we can atleast place at mid distance}
        l = mid+1
    }
    else {
        h = mid-1
    }
}
return ans;
```

min distance we need to place them

stall[]

no: of cows we need to place

$\log_2^n * [N]$ : TC

SC: O(1)

```
bool check( dis, ar[], N, cows)
    lastcow = A[0], c=1
    i = 1; i < N; i++) {
        if( A[i] - lastcow >= dis) {
            // ↳ we can place here
            lastcow = A[i]   c++
            if (c == cows) {return True}
        }
    }
    return false
}
```

TC: O(N)

$TC \rightarrow \left\{ \begin{array}{l} \rightarrow \text{Allocate Books} \\ \text{Painter partition} \end{array} \right\}$

(Bay rdan)