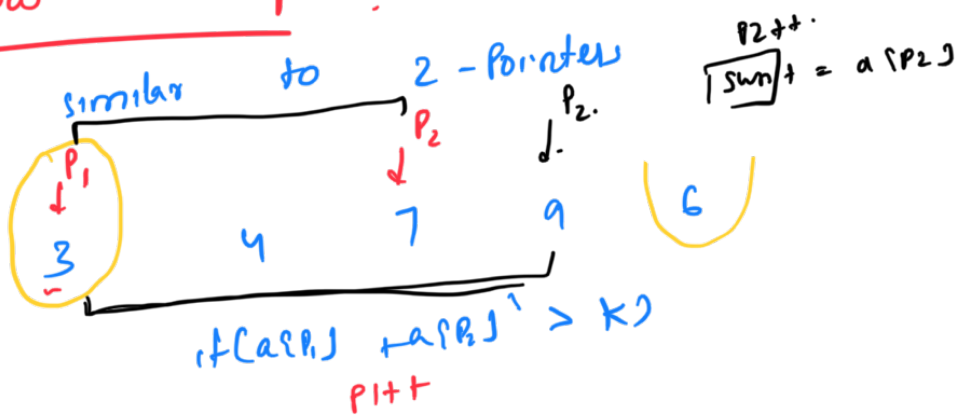
Searching + 2 Pointers

Sliding Window Technique :

→ Almost similar to 2 - Pointers

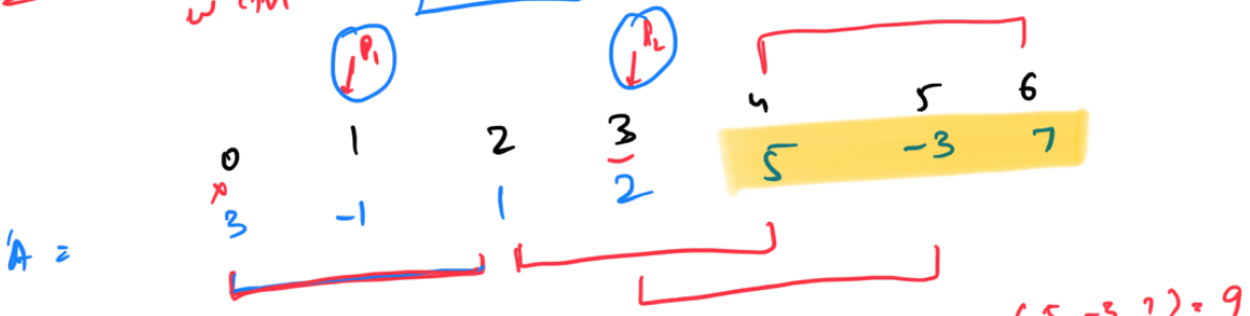


Sliding Window ⇒ Consider all elements between P_1 and P_2.

sum = 14

Question: find a subarray of maximum sum with size K

K = 3



(5, -3, 7) = 9

O(K)

```
for(i= 0; i< K ; i++)
    sum += a[i];
```

sum = 9

```
sum = sum - a[P1];
P1++;
P2++;
sum = sum + a[P2];
```

```
sum += a[i+K]
sum -= a[i];
```

TC: O(K) + O(n-K)

: O(n)

SC: O(1)

Question: Given an array of positive Integers check if there exists a subarray with sum equal to K. K = 15

A: 3 2 5 0 1 8 6 2 10

[1, 8, 6] = 15

Brute Force:

Consider all subarrays, for each subarray compute the up data max.

subarrays: $\frac{n(n+1)}{2}$

T.C for sum of 1 subarray : O(n)

Total T.C: $O(n^3)$

Approach2:

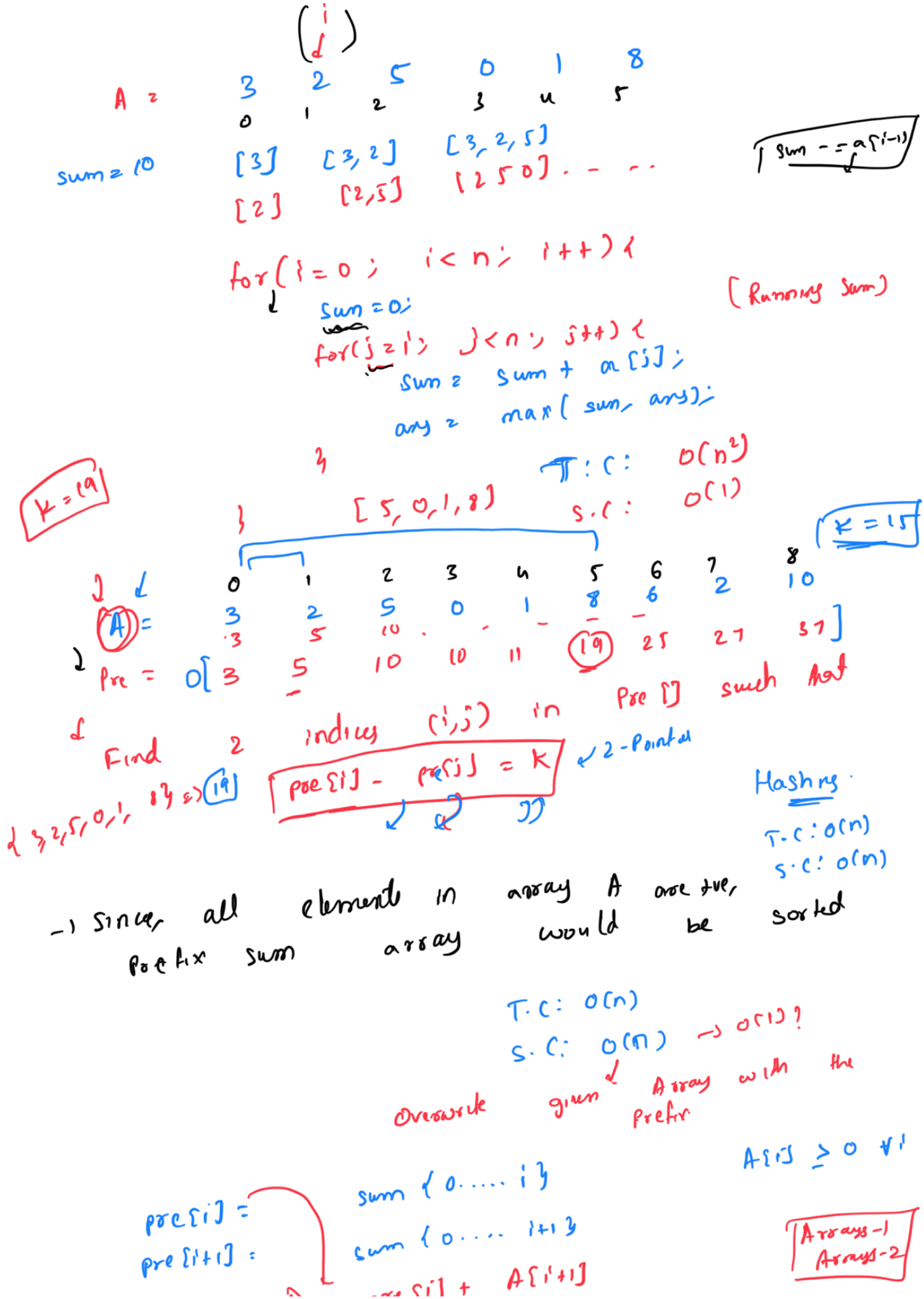
Prefix Sum array:

(i, j) => pre[j] - pre[i-1] : O(1)

T.C: $O(n^2 \times O(1)) = O(n^2)$

S.C: O(n)

$O(n^2)$
$O(1)$


A = 3 2 5 0 1 8
sum = 10
[3] [3,2] [3,2,5] [2,5,0] ...
[2]
Sum -= a[i-1]
for(i=0; i<n; i++){
sum=0;
for(j=i; j<n; j++){
sum = sum + a[j];
ans = max(sum, ans);
}
}
[Running Sum]
T.C: O(n²)
S.C: O(1)
K = 19
[5, 0, 1, 8]
K = 15
A = 3 2 5 0 1 8 6 2 10
Pre = 0 3 5 10 10 11 19 25 27 37
Find 2 indices (i,j) in Pre[] such that
pre[i] - pre[j] = K
2-Pointer
{3,2,5,0,1,8} => 19
Hashing
T.C: O(n)
S.C: O(n)
- Since all elements in array A are +ve, Prefix sum array would be sorted
T.C: O(n)
S.C: O(n) → O(1)?
Overwrite given Array with the Prefix
A[i] ≥ 0 ∀ i
pre[i] = sum{0.....i}
pre[i+1] = sum{0.....i+1}
= pre[i] + A[i+1]
Arrays-1
Arrays-2

pre[i+1] = pre[i] + A[i+1]

pre[i+1] − pre[i] = A[i+1]

A[i+1] ≥ 0

pre[i+1] ≥ pre[i]

Issue: If a subarray [0.... i] has a sum of K, our approach will fail

Array is Sorted

Solution 1:
Add a 0 at the start of prefix

Solution 2:
Do Binary search for K on the prefix array.

Sliding Window (2 Pointers)

1) If a subarray has a sum greater than K, can we reduce the sum by adding more elements?

[Array elements are +ve]

NO!

2) If subarray has sum < K, can we increase sum by adding more elements.

YES!

K = 15

	0	1	2	3	4	5	6	7	8
A =	3	2	5	0	1	8	6	2	10

p1 p2

sum between [p1, p2]

sum = 15 or 15

p2++;

sum of A[P1 : P2] = A[2 : 5]

sum = sum + a[i]

sum = sum - a[P1];

P1++;

T.C:

P1 = 0 ⟶ n } n iterations

P2 = 0 ⟶ n } n iterations

O(n) (n + n)

1) P1 will increase by 1

2) P2 will increase by 1

Binary Search:

Median:

Middle element of an sorted array

A = 1 2 3 5 7

median

N = 5

A = 0 1 2 3 4

$arr[\frac{n}{2}]$

N = 6

A = 0 1 2 3 4 5

1 2 3 4 5 6

Median $\frac{3+4}{2} = 3$

Even length = $\frac{arr[\frac{n}{2}] + arr[\frac{n}{2}-1]}{2}$

$\frac{arr[3] + arr[2]}{2} = \frac{4+3}{2} = 3$

Question: Median of 2 sorted (Distinct elements)

A1 => size M

A2 => size N

M+N is odd

	0	1	2	3	4	5
A1	2	6	11	19	21	30
A2	8	9	15	29	40	

N = 6

N = 5

A = 2 6 8 9 11 15 19 21 29 30 40

0 1 2 3 4 5 6 7 8 9 10

6th element

Index: $(\frac{M+N}{2})+1$ $(\frac{6+5}{2})=5$

(M+N) => odd

Brute Force:

-> Take a new array and throw these 2 element

-> Sort this array $(m+n)\log(m+n)$

-> Return middle element $O(1)$

T.C: $O(m+n)$

S.C: $O(m+n)$

T.C: $(m+n)\log(m+n)$

S.C: $O(m+n)$

A1 = 1 2 3 4 5

A2 = 6 7 8 9 10

count = 0 1 2 3 4 5

arr[] =

Approach2:

A1[6] 2 6 11 19 21 30

A2[5] 8 9 15 29 40

min(15, 19) => median

count = 0 1 2 3 4 (5)

A =

$(m+n)+1^{th}$ element

T.C: $O(m+n)$

S.C: $O(m+n)$

$O(m+n)$

T.C:

S.C: O(1)

A2 6 7 8 9 10
m 1 2 3 4 5

Efficient Approach: (Binary search)

→ B.S on Answer space → Median => belongs to A1 & A2

Answer space: [ ]

↓

low = (min element of the Array)

= Min of both the array

min(A1[0], A2[0])

↓

high = max element of the merged array

= max(A1[n-1], A2[m-1])

Search space: [min(A1[0], A2[0]), max(A1[n-1], A2[m-1])]

(21) (21)

N=6

M=5

A1 = [2 6 11 19 21 30]

A2 = [8 9 15 29 40]

A = [2 6 8 9 11 15 19 21 29 30 40]

0 1 2 3 4 5 6 7 8 9 10

low high

Search space: [2, 40]

$\left(\frac{m+n}{2}\right) = \frac{6+5}{2} = \frac{5}{2}$

low high mid

2 40 21

how do we

Question: Given an [illegible] check if that x a median?

% No. of elements greater than x = No. of elements smaller than x

⇓

x is the Median

1) → No. of elements < Median = $\frac{[m+n]}{2}$ ✓

2) → No. of elements > Median = $\frac{(m+n)}{2}$ ✓

3) → Index of median = $\frac{m+n}{2}$

low = [0, 5]

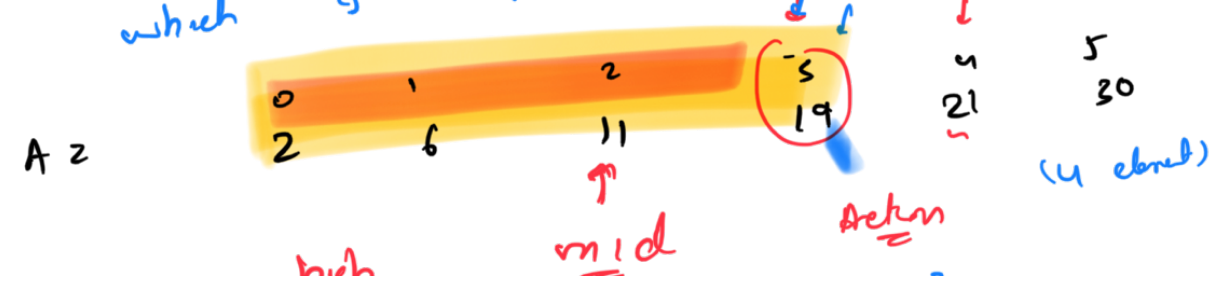
Brute Force:

```
for(i=0; i<N; i++) {
    if( A[i] < x)
        count++;
}
return count
```

T.C: O(n)

Efficient Approach: Binary search?

→ Find the index of largest number which is less than 20



low = 0 → 3 → 3 → 4

high = 5 → 5 → 3 → 3

mid = 2 → 4 → 3

ans = 2
low = mid+1

high = mid-1 (ans+1)

ans = 3
low = mid+1

STOP

[2, 11] < 5

T.C. : O(log n)

N = 6, M = 5

6+5/2 = 5

(log n + log m)
(4) 3 = 7

⑧

A1 : 2 6 11 19 21 30

A2 : 8 9 15 29 40

[21, 40]

Answer Space : [2, 40] =

low	high	mid	No. of elements smaller	Action
2	40	21	7	[2, 40] cannot be Median, high = mid-1
2	20	11	4	[2, 11] cannot be Median, low = mid+1
12	20	16	6	[16, 20] cannot be Median, high = mid-1
12	15	13	5	

A1 = 2 6 11 19 21 30

A2 = 8 9 15 29 40

For : 1) {2, 6, 8, 9, 11} ⇒ [2-11] ⇒ No. of elements smaller than 5 would be less than 5

2) {19, 21, 29, 30, 40} ⇒ [16, 40] ⇒ greater than 5

3) [12, 15]

Largest element would belong to the array.

12 => 5
13 => 5
14 => 5
15 => 5

low = min(A1[0], A2[0]);
high = max(A1[n-1], A2[m-1]);
while (low ≤ high) {
mid = (high + low)/2;
count = 0;
count += BS(A1, mid); → logN
count += BS(A2, mid); → logM
if (count < ((n+m)/2)) (k)
low = mid+1;
}
else if (count > ((n+m)/2)) (k)
high = mid-1;
else {
ans = mid; { Max element in this range }
low = mid+1; → (high = mid)
}

→ update the answer
→ search for better ans

N = 6

A1 = 2 6 11 19 21 30

A2 = 8 9 15 29 40 M = 5

A1 = 2 6 8 9 11 15 19 21 29 30 40
0 1 2 3 4 5 6 7 8 9 10

[12-15]

11 => 4
10 => 4
(11+1)

16 =>
19 => 6
20 => 6

[median+1, high] ≥ 6

[2, 11]

12 => 5

(11+1) = 12

(11+1)

max elem [12-15]

Intermediate element can be

SS: [2, 40]

[2,11] [12,15]

A = 2 6 8 9 11 15 19 21 29 30 40

median ≥ 6

-> one element definitely exists such that no. of elements less and greater than it is equal to $\left(\frac{m+n}{2}\right)$

Belongs to array

$\left(\frac{m+n}{2}\right)$

[x x+1 x+2 (x+3) ? . . . . x+k]

(m+n)

answer

=> For elements in the range, the no. of elements less than it equal to $\left(\frac{m+n}{2}\right)$

No of element < x+3 = $\left\lceil \frac{m+n}{2} \right\rceil$

= $\left\lceil \frac{m+n}{2} \right\rceil$

Search space

at $< \left(\frac{m+n}{2}\right)$ /,

No of elements smaller than $n+i = \left(\frac{m+n}{2}\right)$

Consider $n+3$ to be median.

A: [35, 36, 37, 38, 40, 41, ... 44, ...]

Belongs to the array

→ 40 has be in A1 or A2.

$f(35) = \frac{m+n}{2}$

$f(36) = \frac{m+n}{2}$

$f(40) = \left(\frac{m+n}{2}\right)$

$F(41) = F(40) + 1$

T.C:

Iterations = Log (Length of search space)

Search space: [min(A), max(A)]

= max(A1[n-1], A2[m-1]) − min(A1[0], A2[0])

T.C per iteration: log N + log M

T.T.C: $\log(\max(A_1, A_2) - \min(A_1, A_2)) * [\log M + \log N]$

Question: Element at index k in the sorted array

of size $\left(\frac{m+n}{2}\right)$ with k

Question: $(n+m)$ can be even

A1 = 3 5 7 9

A2 = 1 2 6 8 11 12

Median = $\frac{A\left[\frac{n+m}{2}\right] + A\left[\frac{n+m}{2} - 1\right]}{2}$

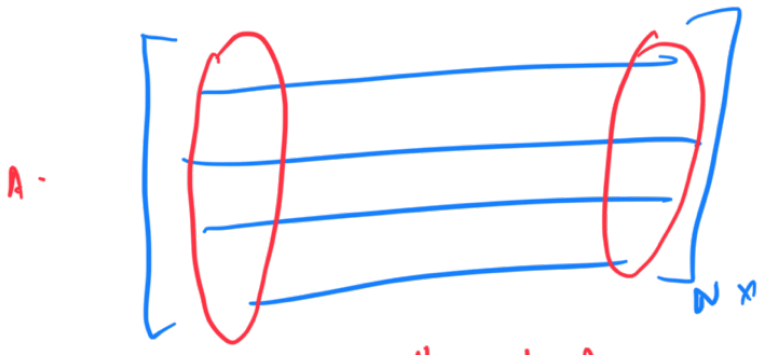
$k = \left(\frac{n+m}{2}\right)$

$k = \left\lceil \frac{n+m}{2} - 1 \right\rceil$

$\lfloor floor\left(\frac{x1+x2}{2}\right) \rfloor$

Question:

We have N sorted arrays of size M



B.S on N arrays

$\log M + \log M \ldots N$ times $= N \log m$

low = min(1st column)

max = max(last column)

```
while (low <= high) {
    mid = (high + low)/2;
    // B.S on all N Rows to count no. of element smaller than mid.
    if (count < n.m/2)
        low = mid+1;
    if (count > n.m/2)  high = mid-1
```

else { ans = mid; low = mid+1; }

}

Total No. of element = n × m

K = $\frac{n \times m}{2}$

T.C: log(max − min) × N log M

→ (N+M is odd) for find K^{th} element.

→ % we want in the Merged

log(min(n,m))

→ Highest Value in that row

A1 =
A2 =

A = 2 6 8 9 11 16 19 21 29 30 40

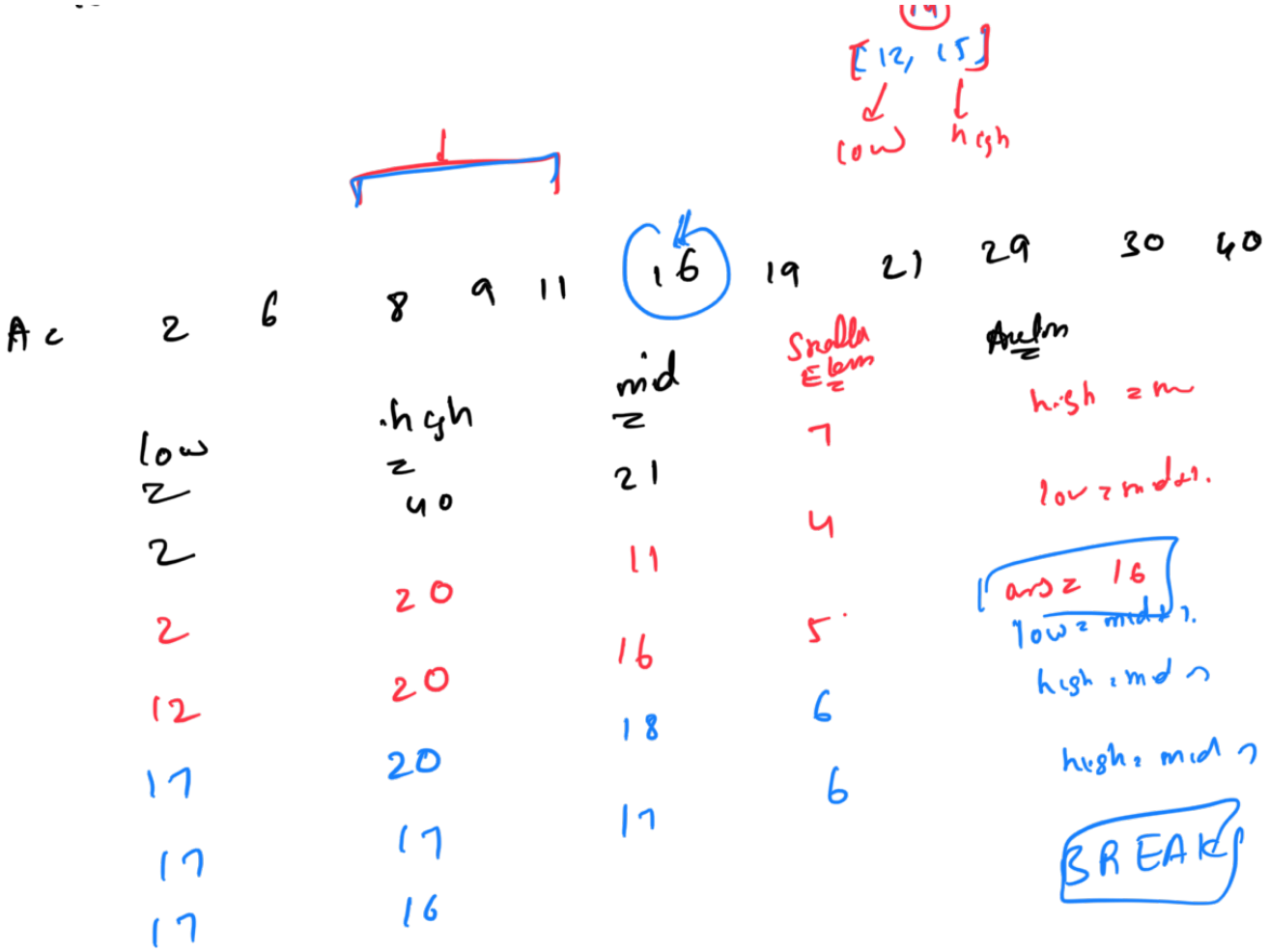
0 1 2 3 4 5 6 7 8 9 10

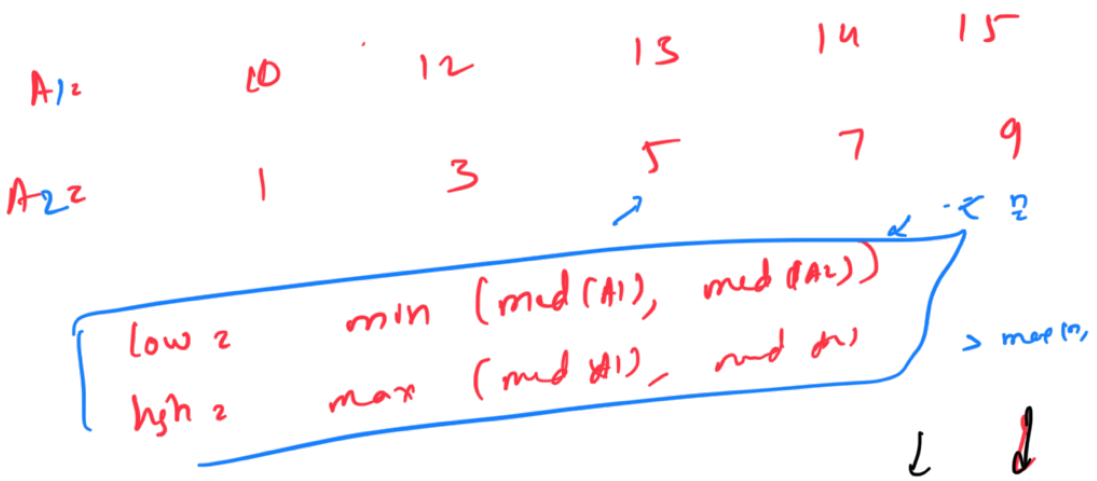
13 [12-15] ≥ 7

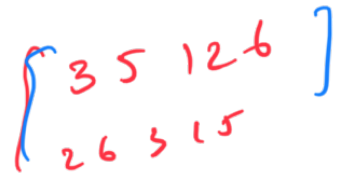
22 [21, 40] (log n + log m)

$\frac{N+M}{2}$

low	high	mid	No. of elements smaller than 21	Action
2	40	21	7	Discard [21,40] 5 high = mid − 1
2	20	11	4	Discard [2,11] low = mid + 1
12	20	16	6	high = mid − 1
[12	15]	13	5	ans = 13 low = mid+1
14	15	14	5	ans = 14 low = mid+1
15	15	15	5	ans = 15 low = mid+1
16	15			





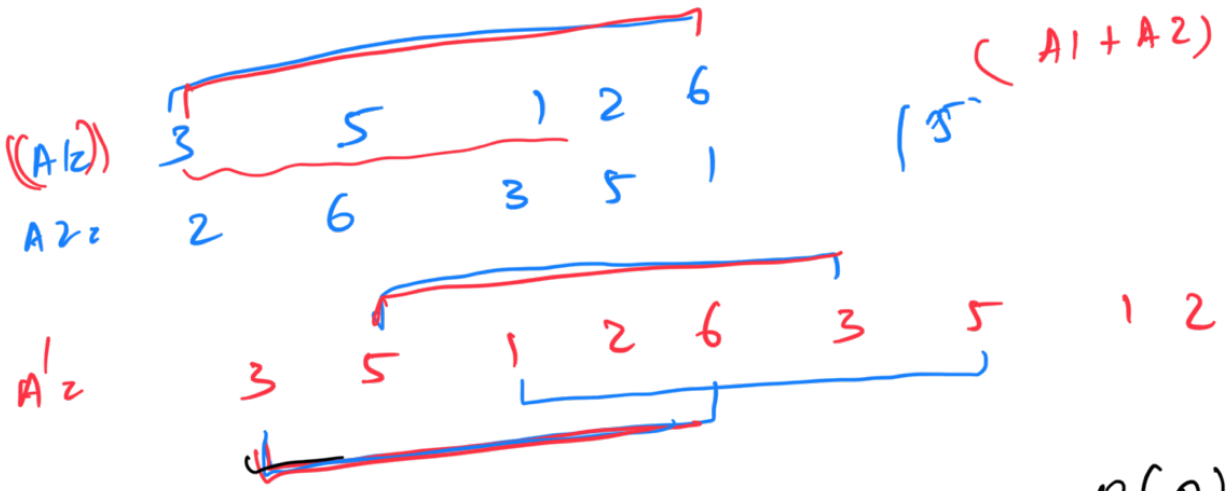


A1 = "35 | 126"

A2 = "26315"

A1 : 3 3 1 5 5 5

A2 : 5 5 5 3 3 1



Prefix Sum Hash :

O(n) T.C.

Z - Algo

KMP

Kadane's Algorithm

A1: 5 7 8 9 10

A2: 1 2 3 4 (6)

low : min (Med1, Med2) : min (8, 3) : 3

high : max (3 8) : 8

[3, 8]

---

q : 1 0 0 1

0 1 0 0

Ans 3

10 ⁽¹¹⁾

0 1 0 1 Base 3 : {0, 1, 2}

11 ⇒ 0 1 0 (2)

3 2 1 0

$0 \cdot 3^3 + 1 \cdot 3^2 + 0 \cdot 3^1 + 1 \cdot 3^0$

$1 \times 2^4 + 0 \cdot 2^3 + 1 \cdot 2^2 + 1 \cdot 2^1 + 0 \cdot 2^0$

1 0 1 1 0 ⇒

4 3 2 1 0

3 6 7 9

Base 10: [0-9]

⇒ Smallest 'K' such that 111 ... = N

$K^P = N$

1 1 1 1 1 1 1 = $K^0 + K^1 + K^2 + \dots$ for (i = 64 ; P >= ... ; i--){

P 3 2 1 0

P = 64 | P = 63 | ... P = 62

$K^0 + K^1 + K^2 + K^3 + \dots K^{64} = N$

$N = 10^{18} = 2^{64}$

Base = 2

P = 64

$\frac{a(r^n - 1)}{r - 1} = \frac{(K^{65} - 1)}{K - 1} = N$

long long int

2 (64)

$(K-1)(\dots\dots\dots)$ (100)^65

$(K-1)$

5^65

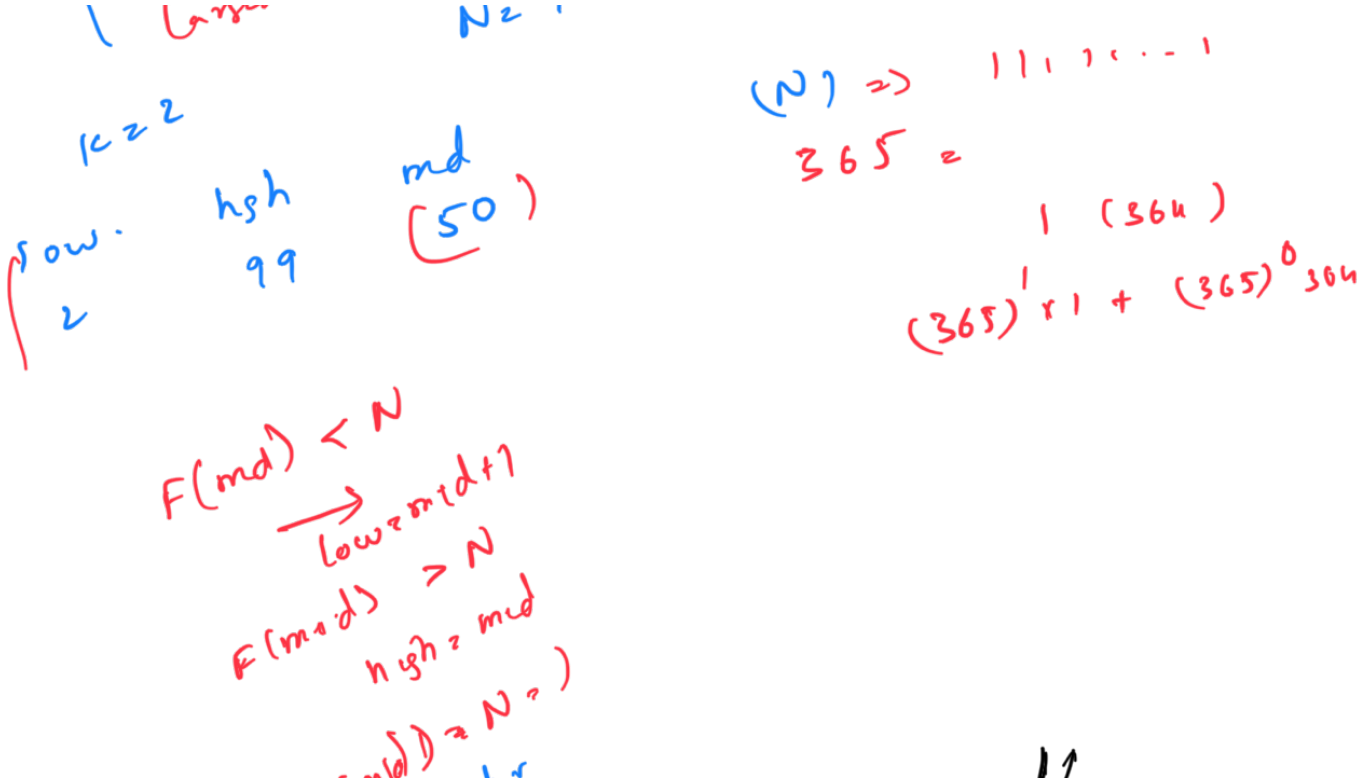
$(K^{65} - 1) \geq N \cdot (K-1)$

Smallest value of K = 2

Largest value of K = N - 1

100

long long



Question: Subarray with sum equal to K Implementation1

```
subarraySum(int a[], int n, int k){
      int p1 = 0, p2 = 0;
      int sum = arr[0];

      while(p2 < n){
            while(sum < k and p2 <= n-2){
                  p2++;
                  sum += a[p2];
            }
            while(sum > k){
                  sum -= a[p1];
                  p1++;
            }
            if(sum == k)
                  return True;
      }
      return False;
}
```

Question: Subarray with sum equal to K Implementation2

```
subarraySum(int a[], int n, int k){
      int p1 = 0;
      int sum = 0;
      for(int p2 = 0; p2 <= n-1; p2++){
            sum += a[p2];

            while(sum > k){
                  sum -= arr[p1];
                  p1++;
            }
            if(sum == k)
                  return True;
      }

      return False;
}
```

Question: Median of 2 sorted arrays

```
int median(int a[], int n, int b[], int m){
       low = min(a[0], b[0]), high = max(a[n-1], b[n-1]);
       ans;
       while(low <= high){
             mid = (high + low)/2
             x = 0;
             x = x + countLess(a, n, mid);    // Can be done using BS
             x = x + countLess(b, m, mid);
             k = (m+n)/2;
             if(x < k)
                   low = mid + 1;
             else if (x > k)
                   high = mid -1;
             else{
                   ans = mid;
                   low = mid+1;
             }
       }
}
```