→ $x = 102$, $x+1 = 103$
sum digits(x) = 3, sod(x+1) = 1

→ $x = 109$, $x+1 = 110$
sod(x) = 10, sod(x+1) = 2

x, sod(x) = d; $x+1$, sum sod(x+1) = (d+1)

$x = 117$, $x+1 = 118$
sod(x) = 9, sod(x) = 10

$x = 99$, $x+1 = 100$
sod(x) = 18

$x = 115$, sod(x) = 7 ✗
$x+1 = 116$, sod(x+1) = 8 ✓

$x = 100$, $x+1 = 99$
sod(x) = 1, sod(x+1) = 18

x, sod(x) = d, $x-1$, sod(x-1) = d-1

for sod(for n>1)
{ for s=0;
while(n>0){
s+= n%10
n = n/10
}
return s;
} $\log_{10} n$

108) Good Integer

→ Given A & B, find all good Integer less ≤ A

a number k is said to be good Integer if

k>0, (k - sod(k)) ≥ B

Solution:

BF:
for(for k = 1; k<=A; k++) {
if(k - sod(k) >= B){
cnt++;
}
}
return cnt;

TC: $O(A)$ × sum of digits() → $\log N$

A = 30, B = 10

is 20 good Integer
20 - (2) ≥ 10 ✓
20 21 22 23 ...
T T T T T

If k is a good Integer → let's say k+1
k - sod(k) >= B
Assume sod(k) = d
k - d >= B

k+1 - sod(k+1)
k+1 - d+1
k - d >= B
k+1 is also a good Integer

is 15 good Integer
15 - (6) < 10
12 13 14 15
F F F F

If k is not a good Integer → let's go for k-1
k - sod(k) < B
Assume sod(k) = d
k - d < B

k-1 - sod(k-1)
↳ with value of this = d-1

k-1 - (d-1)
k-1 - d+1
k - d < B

If k is not a good Integer
k+1 is also not a good Integer

Combine both above things
↳ b
k
[F F F F F F T T T T T T T]
→ Idea BS:
a, b
How many a-b+1

28) Given 2 sorted arrays, find median

What's property of median

[left Array] ≤ [Right array]

If total elements = 16
[left Array 8] ≤ [Right array = 8]
Median: [Max of left Array + Min of Right Array] / 2

If total elements = 15
[left Array = 8] [Right Array = 7]
Median: [Max of left]

2 arrays: N & M

If (N+M) is even
[left Array size: (N+M)/2, (N+M)/2] [Right Array: (N+M)/2]

If (N+M) is odd
[left Array size: (N+M+1)/2] [Right Array (N+M)/2]

For Given N M
Left Array size: (N+M+1)/2
Right Array size: (N+M)/2

Ex:
A[]: 3 5 8 11 15 18
B[]: 2 7 10 13 19 23 25

Observation
N+M = 15
Left Array = 7
Right Array = 8

Thought
Abhiram
Piyush
Rathish
Vaitha

In your left[]
→ it will contain few elements of A
→ it will contain few elements of B

noA = 1, noB = 6, 7
3 <= 23 ✗ → Increase size of A
Elements on left <= Elements on Right ✗

noA = 2, noB = 5
3 7 <= 19 ✗ → Increase size of A

noA = 3, noB = 4
3 5 8 <= 23 ✗ → Increase size of A

noA = 4, noB = 3
3 5 8 11 <= 13 ✗
2 7 10 <= 11 ✗
Left A, Left B, Right A, Right B
11 <= 13
10 <= 11
Max of left Array (11)

TC: log(min(N, M)) ✓

SC: O(1)

// A.size() < B.size()
int median(vector<int> A, vector<int> B){
// for low & high, [BS on size of A {How many elements of A should be on left part}]
int M = A.size(), N = B.size();
int lo = 0, hi = A.size();
while(lo <= hi){
int Asize = (lo+hi)/2 // elements of A should be on left part
int Bsize = (N+M+1)/2 - Asize // Elements of B should be on left part
double LeftA = A[Asize-1];
double LeftB = B[Bsize-1];
double RightA = A[Asize]
double RightB = B[Bsize]
// we need to over each condition
If [LeftA > RightB]{
// A size we have to decrease
high = mid-1
}
else if [LeftB > RightA]{
// increase A size
low = mid+1
}
else{
// we have valid partition
If ((N+M)%2 == 0){
// median [left[] max & right() min]/2
return [max(LeftA, LeftB) + min(RightA, RightB)]/2
left() maximum, Right() minimum
}
else{
// median of left() max
return [max(LeftA, LeftB)]
}
}
}

If (A.size() == 0){
LeftA = INT_MIN
else
LeftA = A[Asize-1]
If (Asize == M){
RightA = INT_MAX
else
RightA = A[Asize]

If (B.size() == 0){
LeftB = INT_MIN
else
LeftB = B[B.size()-1]
If (Bsize == N){
RightB = INT_MAX
else
RightB = B[Bsize]

Edge Case:
A[] = 10 Element →
B[] = 3 Elements
Left Arr: 7
Right Arr: 6
→ we apply BS on size of A
Asize = 3
Bsize = 4
This is not possible

→ To avoid all this we should Apply BS on smaller array size

O(1)

max() {
If (A.size() > B.size())
return median(B, A)
else return median(A, B)
}

Hint For Greed Sub Arrays

0 1 2 3 4 5 6 7 8 9
3 7 10 11 12 15 20 25 30 37 40

B = 15

[0 0]
[0 1]
[0 2]
[0 3]
[0 4]

i — j → [Total subarrays] → [(j-i+1)/2]

→ How many odd subarrays are there starting from i

[i, i+2] i+2x ≤ j
[i, i+2] 2x ≤ j-i
[i, i+2x] x ≤ [(j-i)/2]
[i, i+2x] x = [(j-i)/2]

Total odd subarrays → (x+1)

No: of odd subarrays [(j-i)/2]

40) Given a array length(n), count of subarrays with atleast k unique elements] Enclosing Strings

K = 4

0 1 2 3 4 5 6 7 8 9 10 11
[X X 3 1 2 5 2 5 6 1 1 2]
P_1 P_2 P_3 P_4

		Count	+117
$P_1 = 0$	$P_2 = [5, 11]$	7	<1, 1>
$P_1 = 1$	$P_2 = [5, 11]$	7	<2, 2>
$P_1 = 2$	$P_2 = [5, 11]$	7	~~<3, 2>~~ <5, 2>
$P_1 = 3$	$P_2 = [8, 11]$	4	<5, 1>

↳ All Ending [8 11]
[a b] → b-a+1
→ 11-8+1 → 4

$P_1 = 3$ $P_2 = 8$
$P_1 = 3$ $P_2 = 9$
$P_1 = 3$ $P_2 = 10$
$P_1 = 3$ $P_2 = 11$