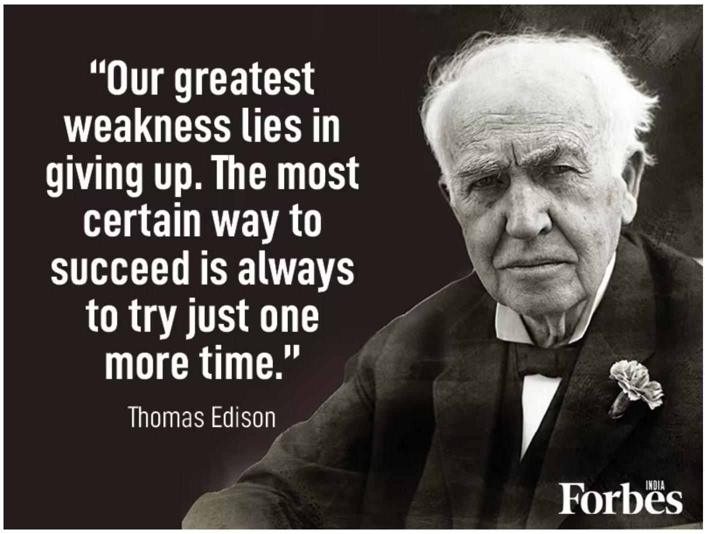
Interview Problems



"Our greatest weakness lies in giving up. The most certain way to succeed is always to try just one more time."

Thomas Edison

INDIA
Forbes

Questions

01. Meeting Rooms

02. Sort nearly sorted arrays

03. Merge k sorted Arrays

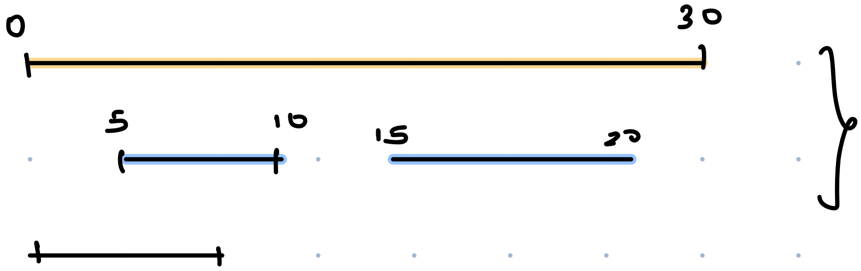
04. Minimum distance Equal pair

05. Minimum window substring

Meeting rooms

Given an array of meeting time intervals where each interval is represented as [start , end] . Your task is to find minimum no. of conference rooms required to schedule all meetings without overlap.

Input : [{0, 30} {5, 10} {15, 20}]



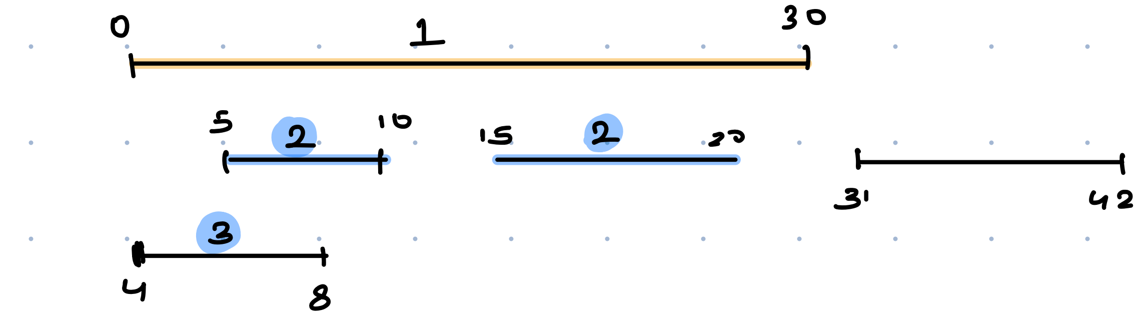
Ans = 2

* Arrays.sort (A , (a, b) → a [0] - b [0])

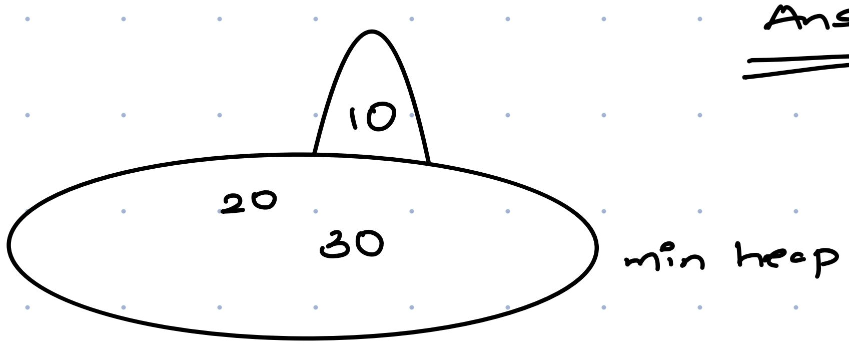
Brute force → We can iterate over intervals and compare the end of one interval and start of next interval. If it overlaps, we will increase count.

TC : $O(n^2)$

SC : $O(1)$



Ans= 3



cnt = 1

4 ≥ 30 ✗ cnt = 2

5 ≥ 8 ✗ cnt = 3

15 ≥ 8 ✓ cnt= 3

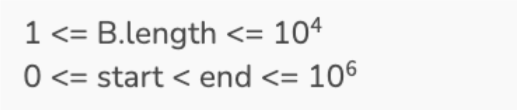
31 ≥ 10 cnt = 3

Ans = 3

Tc : nlogn

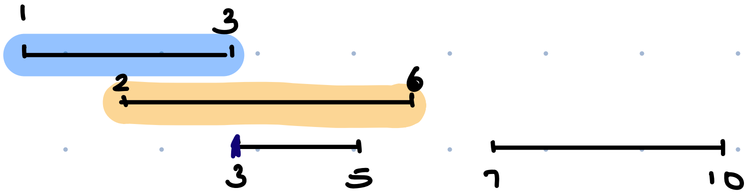
sc : O(n)

* Prefix Array Approach



$1 <= B.length <= 10^4$

$0 <= start < end <= 10^6$



pf[ ] =

0	1	2	3	4	5	6	7	8	9	10
	1	1	-1 +1		-1	-1	+1			-1

prefix sum 0 1 2 2 2 1 0 1 1 1 0 → Ans = 2

```
pf[ ] = new int [10^6];
for (i=0; i<N; i++) {
    s = A[i] [0]
    e = A[i] [1]
    pf[s] ++;
    pf [e] --;
}
```

Tc : O(N + Range)

Sc : O(Range)

```
ans = 0
for ( i=1 ; i< 10^6 ; i++) {
    pf [i] = pf [i-1] + pf [i]
    ans = Math. max ( ans, pf [i] );
}
```

Sort nearly sorted Array

Q Given a nearly sorted array. Sort the entire array

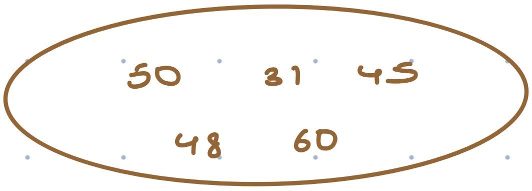
Every element is shifted from its correct position by atmost k steps

	0	1	2	3	4	5	6	7	8	
A[ ] = {	13	22	31	45	11	48 ↑	20	60	50 }	k = 4

Sorted A[ ] = { 11 13 20 22 31 45 48 50 60 }

0	1	2	3	4	5	6	7	8
11	13	20	22	31	45	48	50	60

Soln → Arrays. sort (A)



K+1 size

min heap

Ans = 11 13 20 22 31 45 48 50 60

Code → By yourself

* Merge K - sorted Arrays

AL = {
0 { 2, 3, 5, 11 }
1 { 1, 5, 7, 9 }
2 { 0, 2, 4 }
3 { 3, 4, 5, 6, 7, 8 }
}

Ans = { 0, 1, 2, 3, 3, 4, 5, 5, 5 . . . . }

Solution → Use two pointer technique similar to Merge two sorted Arrays (K−1) times

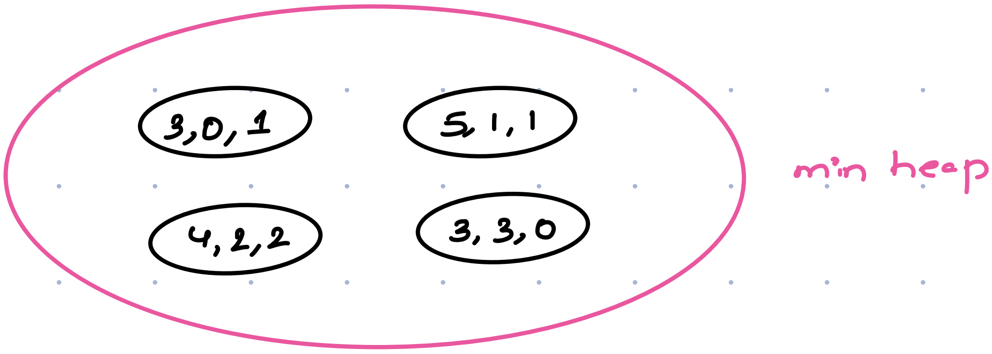
⇒ let's say each array is of length N

$$TC = 2N + 3N + 4N + \dots + KN$$

$$= N\left(\frac{K(K+1)}{2} - 1\right) \approx O(K^2N)$$

* Better Solution

⇒ We can maintain a min heap of K size, remove minimum value from heap and add to ans, add next element of removed array to heap , do this until all element are exhausted



triplet = val, li, i

Ans = {0 , 1 , 2 , 2 , ... }

```
lists = {  0  {2, 3, 5, 11}
           1  {1, 5, 7, 9}
           2  {0, 2, 4}
               0  1  2
           3  {3, 4, 5, 6, 7, 8}  }
```

```
class triplet implements Comparable <I>{

    int val;
    int li;
    int di;

    triplet (int v, int li, int di){
        this.li = li:
        this.val = v;
        this.di = di;
    }

    public int compareTo (triplet Other){
        if (this.val < Other.val) return -1;
        else if (this.val > Other.val) return 1
        else return 0;
    }
}
```

```
PriorityQueue<triplet> pq = new PriorityQueue<>();

for( i=0; i< lists.size(); i++){
    triplet tp = new triplet(lists.get(i).get(0), i, 0);
    pq.add(tp);
}

while (pq.size() >0){
    triplet tp = pq.remove();
    ans.add(tp.val);
    if( tp.di + 1 < lists.get(tp.li).size()){
        int v= lists.get(tp.li). get (tp.di+1);
        int l= tp.li
        int d = tp.di + 1
        pq.add(new triplet(v,l,d));
    }
}
```

Total ele = k*n

TC: $O(k \underset{\uparrow}{n} \log k)$

SC: $O(k)$

m

* Minimum Distance Equal pair

Given an array A, find a pair of indices (i, j) such that A[i] == A[j] & absolute difference |i - j| is minimised.

Basically, find two equal elements in array that are closest to each other & return the distance.

$$\begin{array}{lcccccc} A[\,] = & \{7 & 1 & 3 & 4 & 1 & 7\} \\ & 0 & 1 & 2 & 3 & 4 & 5 \end{array}$$

Ans = 4-1 = $\underline{\underline{3}}$

* Hashmap → To get distance b/w two same elements.

$$\begin{array}{lcccccc} A[\,] = & \{7 & 1 & 3 & 1 & 1 & 7\} \\ & 0 & 1 & 2 & 3 & 4 & \end{array}$$

Ans = 4-3 = 1

```
ans = ∞
for (i=0; i<N; i++){
    if (hm.containsKey(A[i])){
        ans= Min(ans, i - hm.get(A[i]));
    }
    hm.put(A[i], i);
}
```

TC: O(N)
SC: O(N)

```
if (ans == ∞) return -1
return ans;
```

* Minimum Window Substring

Given two strings str & t, find the minimum window in str which contains all characters of t (including duplicates). If no. such window exists, return an empty string // uppercase characters

str = " A D O B E C O D E B A N C "
t = " A B C " } Ans = " B A N C "

str = " A D B E C D E B A N C "
0 1 2 3 4 5 6 7 8 9 10
t = " A B C B " } Ans = B E C D E B A

Brute force → ∀ all substrings of str, check if it contains all the character of t.

01. Store freq of t in a freq-t array 4

02. Store freq of substring of s = t in a freq-s array.

Q3. $\forall$ char freq $\underbrace{\text{(substring)}}_{\text{can be an potential ans}} \geq$ freq(t)

Dynamic sliding window to shrink the window

										l ↓			r ↓	
str =	" a	d	o	b	e	c	o	d	e	b	a	n	c	"
	0	1	2	3	4	5	6	7	8	9	10	11	12	

t = " a b c "

freq_t

a → 1

b → 1

c → 1

freq_s

a → 1

d → ~~1~~ ~~2~~ 1

o → ~~1~~ ~~2~~ 1

b → ~~1~~ ~~2~~ 1

e → ~~1~~ ~~2~~ 1

c → ~~1~~ 0

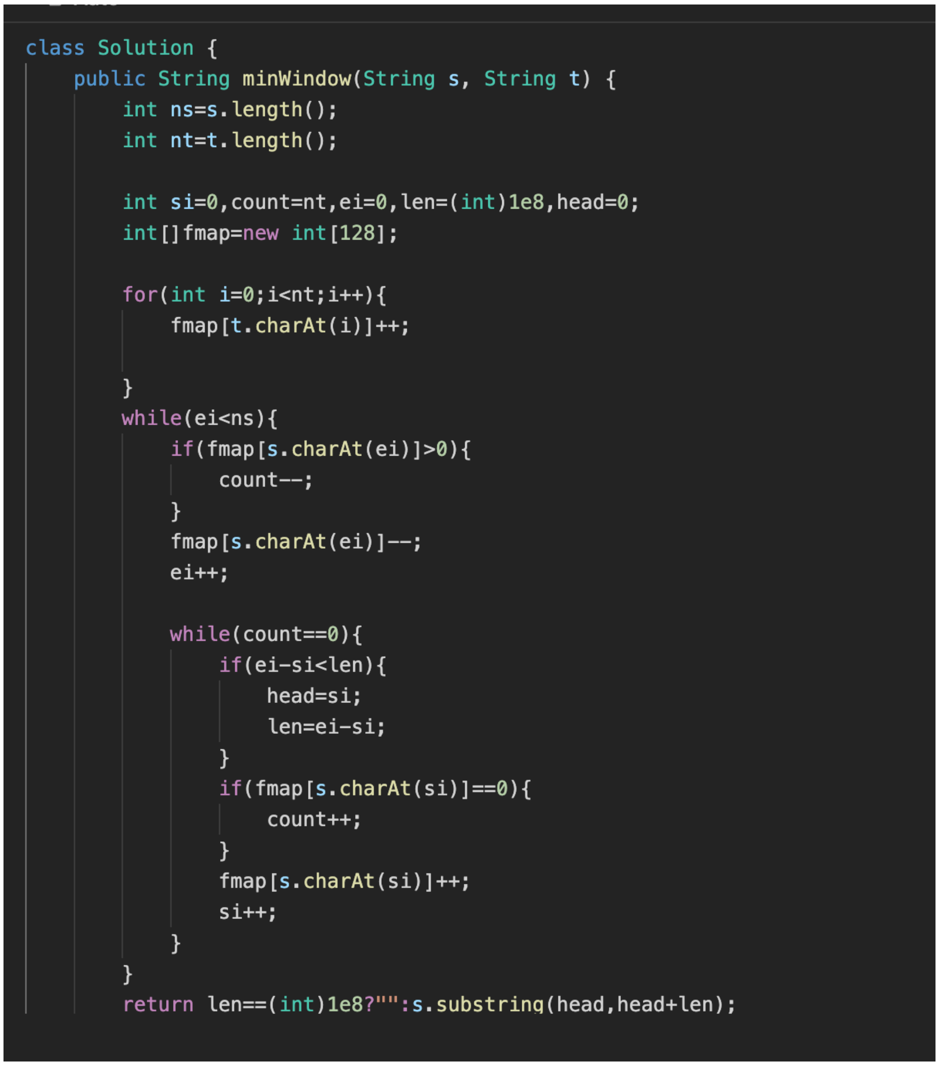
len = r − l + 1

= 6

```
* freq_t = new int (26)

  for ( i=0; i<m: i++ ){
  |   freq_t [ t(i) - 'a' ] ++;
  }

* freq_s = new int [26]:

  ans = 0

  l = 0          int st = -1
  r = 0
```

```
while (r < N){
    if ( check(freqs, freqt )){
        if (r-l+1 < ans ){
            ans = r-l+1
            st = l
            freqs [str[l] - 'a'] --;   l++;
        }
    }
    else {
        freqs[str[r] - 'a'] ++;
        r++;
    }
}
return str.substring(st, st + ans);
```

check if $\underbrace{freqs[i] \geq freqt[i]}_{\text{for all 26 characters}}$



```java
class Solution {
    public String minWindow(String s, String t) {
        int ns=s.length();
        int nt=t.length();

        int si=0,count=nt,ei=0,len=(int)1e8,head=0;
        int[]fmap=new int[128];

        for(int i=0;i<nt;i++){
            fmap[t.charAt(i)]++;

        }
        while(ei<ns){
            if(fmap[s.charAt(ei)]>0){
                count--;
            }
            fmap[s.charAt(ei)]--;
            ei++;

            while(count==0){
                if(ei-si<len){
                    head=si;
                    len=ei-si;
                }
                if(fmap[s.charAt(si)]==0){
                    count++;
                }
                fmap[s.charAt(si)]++;
                si++;
            }
        }
        return len==(int)1e8?"":s.substring(head,head+len);
```