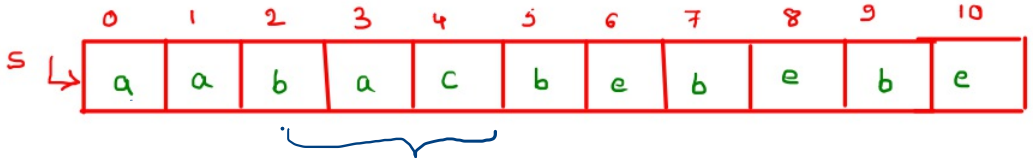
. Find the length of Longest Substring with K unique characters



K = 3

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

2) S.C : $O(n)$

S_1 = b a c → 3

* S_2 = a a b a c → 3

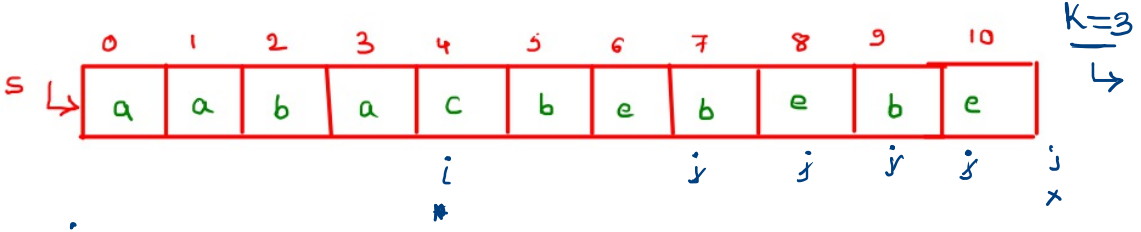
len(S_2) = 3

a - 3
b - 1
c - 1

* of sub-strings

n → $\frac{n \times (n+1)}{2} \Rightarrow O(n^2)$

e b e
↳ e - 2
b - 1
2



- S.W (2 types)
  - ① fixed → K
  - ② variable → *
    + set / HashMap

start = i

end = j

max = 7

hm

key	value
b	1 1 1 1 3
c	1
e	1 1 3

hm.size() → 3

T.C : O(n)

S.C : O(K)

```
function fun(String s, k) // k unique characters
{
        Let hm be a Map
        j=0,i=0,res=-1
    →   while(j<s.length())
        {
                map.put(s.charAt(j), map.getOrDefault(s.charAt(j),0)+1) // update the key and it's value to map
                if(hm.size()==k)
                {
                        res=Math.max(j-i+1,res)
                }
                else if(hm.size()>k)
                {
                    →   while(hm.size()>k)
                        {
                                map.put(s.charAt(i),map.get(s.charAt(i))-1) //it decreases the value of particular key
                                if(map.get(s.charAt(i))==0)
                                        map.remove(s.charAt(i))
                                i++
                        }
                }
                j++
        }
        return res
}
```

Greedy

optimization (maximize / minimize / largest / smallest)

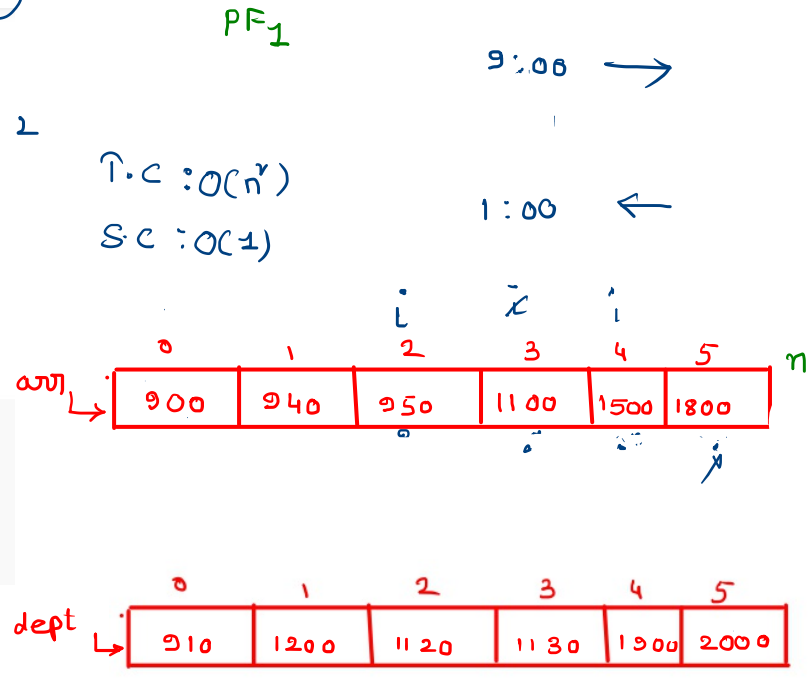
Minimum Number of Platforms Required for a Railway/Bus Station

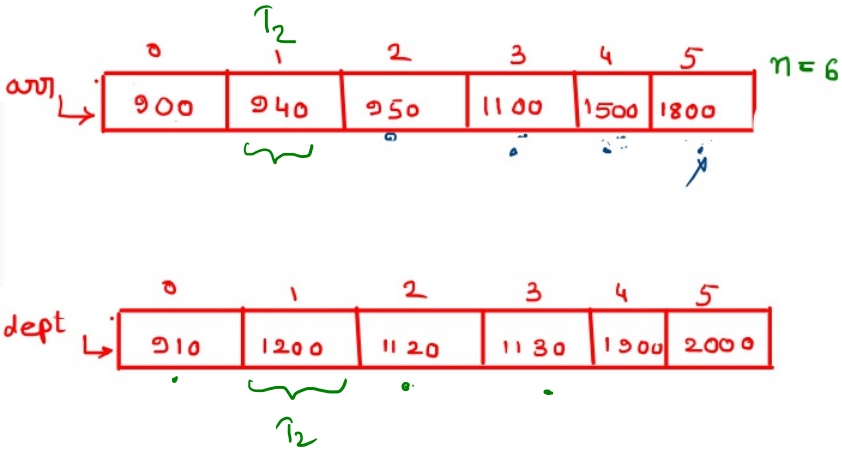
Difficulty Level : Medium • Last Updated : 19 Jan, 2022

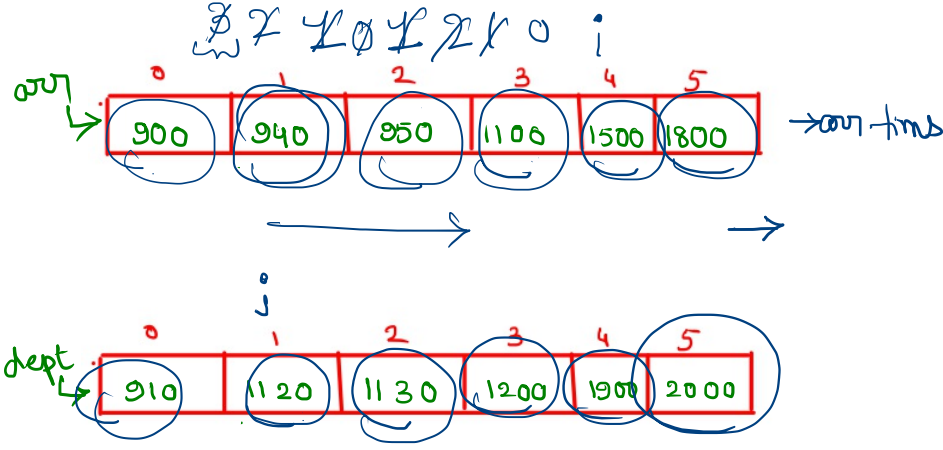
Given the arrival and departure times of all trains that reach a railway station, the task is to find the minimum number of platforms required for the railway station so that no train waits.
We are given two arrays that represent the arrival and departure times of trains that stop.

Examples:

Input: *arr[] = {9:00, 9:40, 9:50, 11:00, 15:00, 18:00}*
dep[] = {9:10, 12:00, 11:20, 11:30, 19:00, 20:00}
Output: *3*
Explanation: *There are at-most three trains at a time (time between 9:40 to 12:00)*







res = 1

2

3

PF

1

2

3

i = 01

j = 0

→ arr[i] ≤ dep[j]

950 ≤ 1120

1100 ≤ 1120

→ arr[i] > dep[j]

```
function fun(arr[],dept[],n)
{
        pf_count=1, res=1
        i=1, j=0
        while(i<n && j<n)
        {
                if(arr[i]<=dept[j])
                {
                        pf_count++
                        i++
                }
                else if(arr[i]>dept[j])
                        pf_count--
                        j++
                if(pf_count>res)
                        res=pf_count
        }
        return res
}
```

→ T.C ⇒ $n\log n + n$ ⇒ $O(n\log n)$

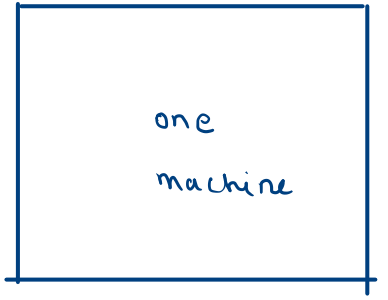
S.C ⇒ $O(1)$

objective → max profit

* Job scheduling with Dead lines

let $J_i \rightarrow$ 1 Hour

Xerox - Shop



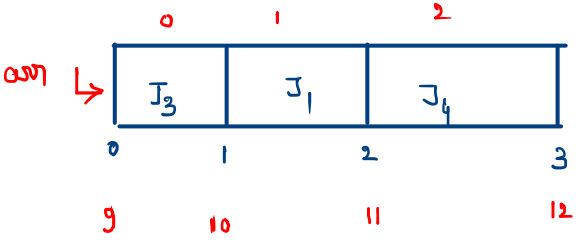
9:00 AM →

11:00 AM

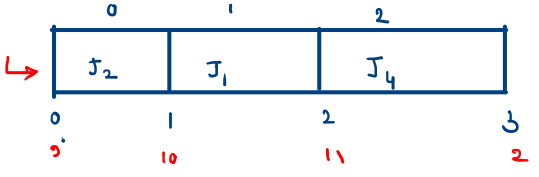
12:00 AM

	J_1 ✓	J_2	J_3 ✓	J_4	J_5
P :	20	15	10	5	1
D :	2	2	1	3	3

→ C_1 $10+20+5=35$



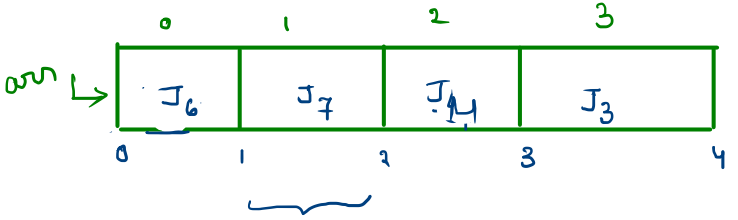
→ C_2 $20+15+5=40$



Ex 2 :-

Greedy About profit

	J_1	J_2	J_3	J_4	J_5	J_6	J_7
P:	3	5	20	18	1	6	30
D:	1	3	4	3	2	1	2



$6 + 30 + 18 + 20 = 74$