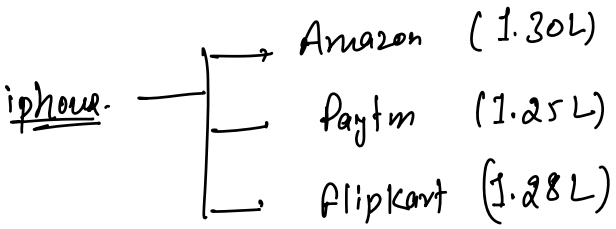
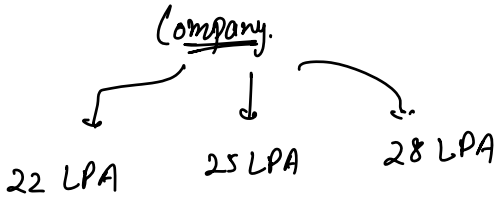
Greedy.

↓

Maximise profit / Minimising loss.





- Work culture
- Working hours
- E.SOP's.

Q: There is a limited time sale going on for toys.

A[i] → sale end time for ith toy.

B[i] → Beauty of ith toy

Time starts with T = 0 & it takes 1 unit of time to buy one toy & toy can only be bought if T < A[i].

Buy toys such that sum of beauty of toys is maximized.

A[] → [3 1 3 2 3]
B[] → [6 5 3 1 9]

T = ∅ 1 2 3

ans → 9 + 6 + 3 = 18.

T =	toy	beauty
∅	1	5
1	4	9
2	0	6
3		
	ans →	20

A[] → [1 2]
B[] → [3 1500]

T = 0 1

ans = 3 + 1500 = 1503.

Greedy → Buy everything → asunding order of time

A[] → [1 3 3 3 5 5 5 8]
B[] → [5 2 7 1 4 3 8 1]

T = ∅ 1 2 3 4 5

B[i] is very less → ignore.

ans = 5 + 7 + 4 + 3 + 8 + 1

= 28.

[Correcting an incorrect step from the past.]

idea → min Heap.

A[] - [1 3 3 3 5 5 5 8]

B[] - [5 2 7 1 4 3 8 1]
(0) (1) (2) (3) (4) (5) (6) (7)

T = 0 1 2 3 4 5 6

5, ~~2~~
7, 4, 1
3, 8

ans = 5 + 7 + 4 + 3 + 8 + 1
= 28.

① Sort them in increasing order of time.

② MinHeap heap;

```
T = 0
for( i = 0; i < N; i++){
    if( T < A[i] ){
        heap.insert( B[i] );
        T++
    }else{
        if( B[i] <= root of min Heap) → ignore.
        else {
            extractMin() from heap
            insert(B[i])
        }
    }
}
```

{ T.C → O(NlogN)
S.C → O(N) }

③ Remove all elements from heap & add them to get ans.

Q: There are N students with their marks.
Teacher has to give them candies such that

a) Every student should have atleast one candy.

b) Student with more marks than neighbours have more candies than them.

Find minimum candies to distribute.

$i \to i-1,\ i+1$

7, 9, 6

A → [1 5 2 1] (0 1 2 3)

1 2 1 1

3 2

ans = 7

A → [4 4 4 4 4] (0 1 2 3 4)

1 1 1 1 1

ans = 5

A → [8 10 6 2] (0 1 2 3)

1 2 2 1

3

ans = 7.

A → [1, 6, 3, 1, 10, 12, 20, 5, 2] (0 1 2 3 4 5 6 7 8)

C → [1 1 2 1 2 3 4 2 1]

3

ans = 19.

Greedy

① $\forall i,\ C[i] \geq 1$

② (a) $\forall i$, if $(A[i] > A[i-1]) \Rightarrow C[i] > C[i-1]$ (left to right)

(b) $\forall i$, if $(A[i] > A[i+1]) \Rightarrow C[i] > C[i+1]$ (right to left)

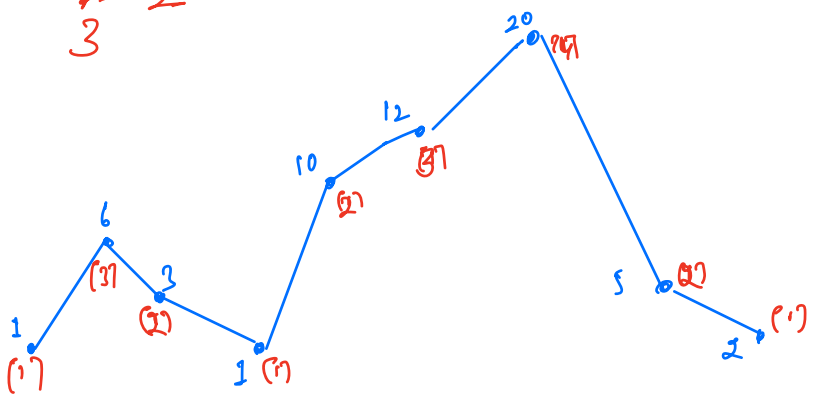
$C[i] = C[i-1] + 1$ if $(C[i] \leq C[i+1])$

$C[i] = C[i+1] + 1$

A → [1, 6, 3, 1, 10, 12, 20, 5, 2]
0 1 2 3 4 5 6 7 8

C → [1 1 1 1 1 1 1 1 1]
2 2 2 3 4
3

ans → 19.



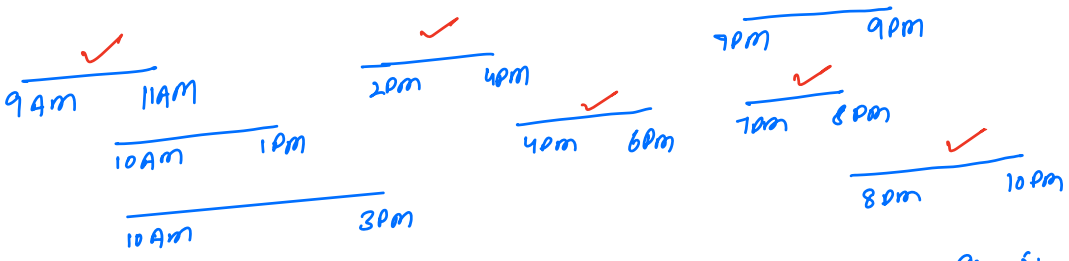
T.C → O(N)
S.C → O(N)

Break → 11:38 → 11:45.

Q: Given N jobs with their start & end-time. Flipkart

find the max jobs that can be completed if only one job can be done at a time.

$S[i] \geq E[i-1]$



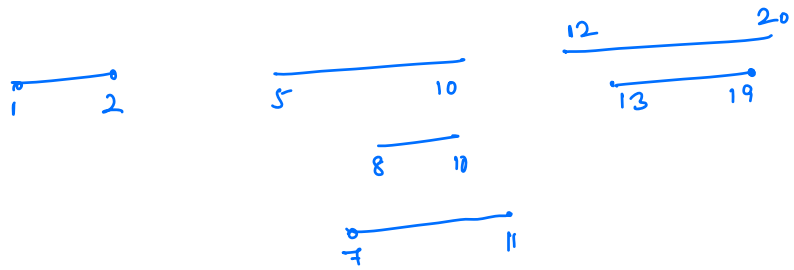
ans = 5.

S : [1 5 8 7 12 13]

E : [2 10 10 11 20 19]

0 1 2 3 4 5

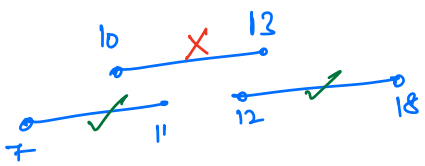
ans = 3.



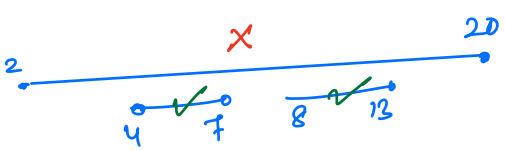
Greedy

① Duration

② Start time.



ans = 2.



ans = 2.

start early + minimum duration ⇒ end early.

② End-time.

S = [1 5 8 7 12 13]

E = [2 10 10 11 20 19]
0 1 2 3 4 5

↓ sort on end-time.

S = [1 5 8 7 13 12]

E = [2 10 10 11 19 20]

ans = ~~1~~ 2 3 , lastEnd = ~~2~~ ~~10~~ 19

Pseudo-code

① Sort on the basis of End-time.

② ans = 1, lastEnd = E[0]

```
for( i: 1 ; i < N ; i++){
    if( S[i] >= lastEnd ){
        ans ++
        lastEnd = E[i]
    }
}
return ans;
```

T.C → O(NlogN)
S.C → O(1)

Q) N people are sitting randomly on M seats placed in a line. $[N \le M]$

X → occupied

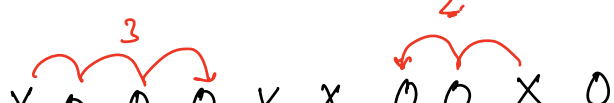
O → empty

i-1 ← i → i+1

In one move a person can move to any adjacent empty seat.

Find minimum moves to make all people sit together.

S = X O O O X X O O X O

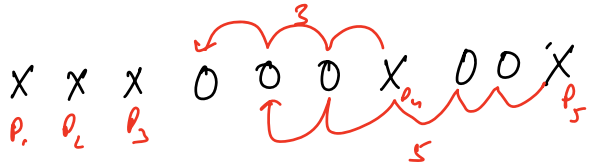


ans = 5.

S = X O O O X O O O X

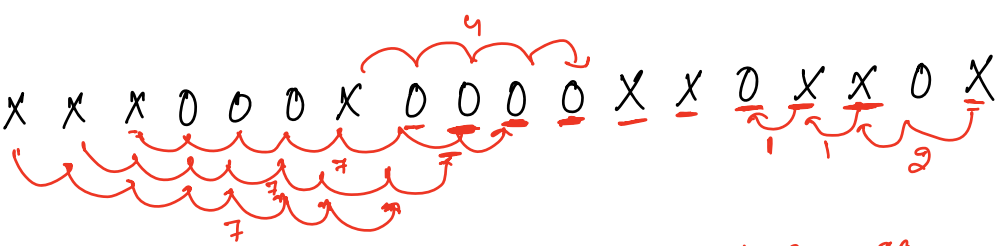
ans = 6.

S = O X X X O O O X O O X



ans = 8.

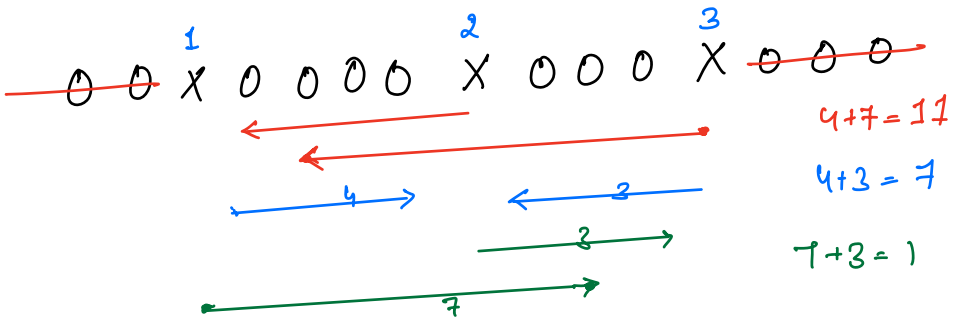
S: O X X X O O O X O O O O X X O X X O X



✓ 29

28

7+7+7+4+1+1+2 = 29.

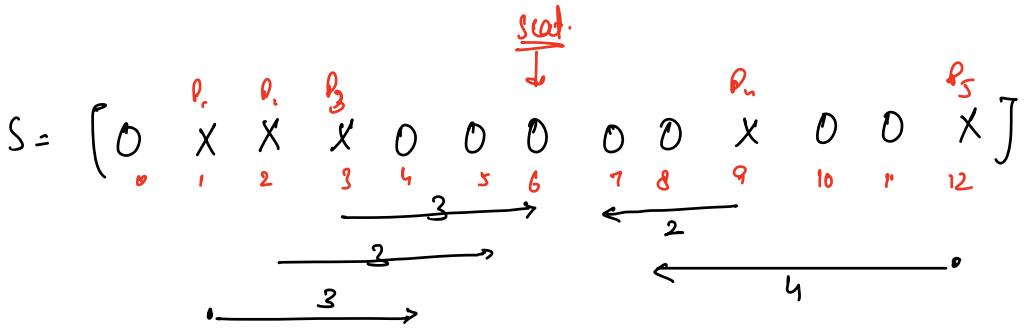


4+7 = 11

4+3 = 7

7+3 = 1

[middle person]



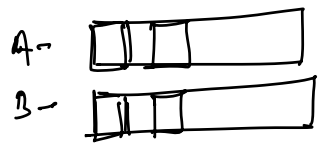
middle person → 5 + 7 = 12

middle seat → 15.

subarray containing max no. of X's

#todo.

1. Pair {
int si
int ei
}



. Pair [ ] arr = new Pair [N];

Arrays.sort (arr, new Comparator ( ) {

public int compare (Pair a, Pair b) {

return a.ei - b.ei

}

})