IPhone 13

Nike Ad- -   ← - -

1 2 10 ✓ 5 2 5 ✓ ←

$\boxed{A[i] + A[i] = max}$ → select max 2 elements & add. $10+5=\underline{\underline{15}}$

TC=O(N) SC=O(1)

Q. There is a limited time sale of toys,

A[i] → Sale end time for ith toy

B[i] → Beauty of ith toy.

It takes one unit of time to get a toy & toy can only be taken if current time < A[i]. Time starts from 0.

Select toys to maximize sum of beauty.

correcting a incorrect step from past. ✓

$$\begin{array}{lccccc} & 1 & 2 & 3 & 4 & 5 \\ A \rightarrow & [3 & 1 & 3 & 2 & 3] \\ B \rightarrow & [6 & 5 & 3 & 1 & 9] \end{array}$$

Time 0 → B[2] = 5

1 → B[1] = 6

2 → B[5] = 9

3 → x

20 ✓ (Ans)

B → [3 10]

A → [1 2]

Time 0 → 3

1 → 10

13 ✓

Sort wrt A[i] i.e. sale end time.

$$\begin{array}{lccccccc} & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ A \rightarrow & [1 & 3 & 3 & 3 & 5 & 5 & 5] \\ B \rightarrow & [5 & \cancel{2}\,5 & 7 & 1 & 4 & 3 & 8] \end{array}$$

important to consider ∵ beauty is high!

can be ignored ∵ beauty is very less.

B[i]

Time 0 → 5

1 → ~~2~~ 8

2 → 7

3 → 4

4 → 3

5 → x

Sum = 27 ✓

⇓

Use min heap to deselect min beauty toy.

Brute → Try all subsets → $TC = O(2^n)$

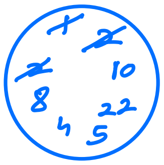
```
sort wrt A[i];   t=0;
for(i=0; i<n; i++){
    if(A[i] > t){
        insert B[i] in min heap;
        t++;
    }else{
  →     if(root of heap < B[i]){
            delete root & insert B[i];
        }
    }
}
return sum of elements in heap. ✓
```

A→[1 1 3 3 3 4 6 6 6]

B→[1 2 10 2 22 4 10 5 8]

0 ✗ 1 ✗ 2 ✓ 3 ✗ 4 ✓ 5 ✓ 6 ✓ 7 ✓ 8 ✓

T⇒0→~~1~~ ~~2~~ 8
1→10
2→~~2~~ 22
3→4
4→10
5→5
6



Min Heap

Ans = 8+10+22+4+5

TC=O(Nlog(N)) ✓

SC=O(N)

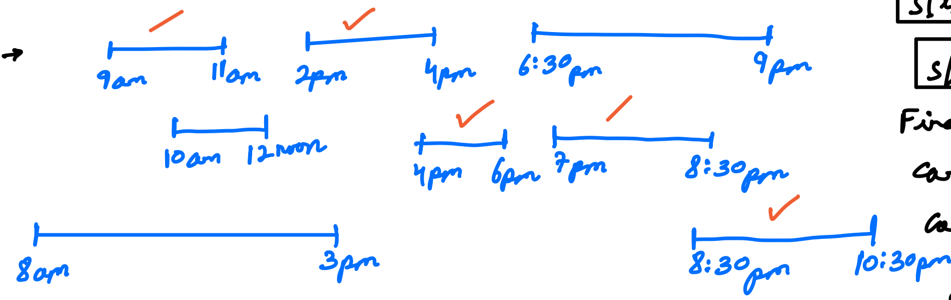
$s[i] < E[i] \quad \forall i$

$s[i+1] \geq E[i]$

Find max jobs that can be done, if we can only do one job at a time.

Given→St & End ∀ jobs.

Q→



Ans = 5

1) N=1   Ans = 1 ✓

2) Assume for N-1 }→ Finish (N-1) jobs as early as possible ⇒ max the possibility of doing Nth job.

3) Solve for N

Try best possibility via induction. ✓

Steps → 1) Sort based on end time. ✓
2) Select the jobs from left to right.

St→[8 12 1 3 4 5 13]
End→[11 14 5 4 5 6 19]

⇓

1 2 3 4 5 6 7

St→[3 1 4 5 8 12 14]

End→[4 5 5 6 11 14 20]

✓ ✗ ✓ ✓ ✓ ✓ ✗

[15 16 17]
[16 17 18]

```
sort wrt End[i];   ans=0;  lastEnd=0;
for(i=0; i<n; i++){
    if(start[i] >= lastEnd){
        ans++;  lastEnd = End[i]; =14
    }
}
return ans;
```

ans = ~~1~~ ~~2~~ ~~3~~ 4
5

TC=O(Nlog(N))   SC=O(1)

Q→ N students in a class.

Every student has marks from previous test in array A.

Distribute candies s.t →

1) Every student has atleast one candy. ✓

2) student with higher marks have more candies than neighbours.

$i \to i-1$, $i \to i+1$ ✓

✓ if $A[i] > A[i-1] \Rightarrow C[i] > C[i-1]$

✓ & $A[i] > A[i+1] \Rightarrow C[i] > C[i+1]$

Find min candies to distribute.

	1	2	3	4	5
A →	1	5	2	1	6
candies →	1	3	2	1	2

Ans = 1+3+2+1+2 = 9

Solve for one condition at a time. ✓

	1	2	3	4	5	6	7	8	9
A →	2	6	3	1	10	12	20	5	2
						3	4	1	1
✓ C →	1	2	1	1	2	3	4	2	1 ←
→	1	3	2	1	2				

✓

Ans = 1+3+2+1+2+3+4+2+1
= 6+10+3 = 19 ✓

Condition 1 → $A[i] > A[i-1] \Rightarrow \boxed{C[i] > C[i-1]} \Rightarrow \boxed{C[i] = C[i-1]+1}$ ✓

else → $C[i] = 1;$

Condition 2 → $A[i] > A[i+1] \Rightarrow \boxed{C[i] > C[i+1]}$ ✓ $\Rightarrow C[i] = \max(C[i], C[i+1]+1)$ ✓

else → $C[i]$ is unchanged. ✓

TC = O(N) SC = O(N)

Q→ Some seats are arranged in a row. N people are sitting on some seats.

x → occupied
0 → empty seat

In one move, a person can move to adjacent seat. Find min moves to bring all people together.

[x 0 0 0 x x 0 0 x]

3 + 2 = 5 (Ans)

3 + 3 + 5 = 11

Try small inputs in increasing order. ✓

N=1 [0 0 0 - - . x 0 0 0 - -] Ans = 0

N=2 [- - - x - - x - -] Ans = Either one can move to other.

N=3 [- - x 0 0 0 x 0 0 0 0 0 x - - -]

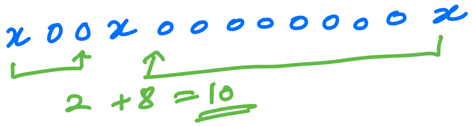
8 + 5 = 13

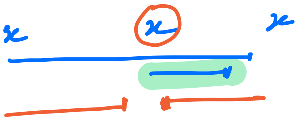
3 + 8 = 11

3 + 5 = 8

✓ ✓

1) Try to bring people to a place where atleast one person is sitting. ✓


x o o x o o o o o o o o x
2 + 8 = 10


x
x
x