Greedy

Maximize our profit & Minimizing our loss.

Iphone

Amazon (1.3L)

Paytm (1.2L)

flipkart (1.25L)

Considering → Price (min).

Company

45L

30L

48L

→ Job is remote

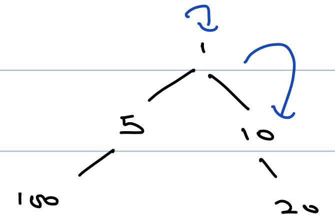
→ work culture

→ kind of Project

→ timings.

Greedy →

it is an approach to solve optimisation problems by making locally optimal choices .


2
1
5
10
100
20

Free Cars

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, and it takes 1 unit of time to buy one toy and the toy can only be bought if T < A[i].

sale end timing

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

sale end time

	0	1	2	3	4
A[ ] =	3	1	3	2	3
B[ ] =	6	5	3	1	9

Beauty.

toy →	Beauty
0 →	6
2 →	3
4 →	9

Total Beauty → 18

$T = \cancel{0}\ \cancel{1}\ \cancel{2}\ 3$

idea :- Buy in the Order of Beauty.

sale end time

	✓ 0	✗ 1	✓ 2	✗ 3	✓ 4
A[ ] =	3	1	3	2	3
B[ ] =	6	5	3	1	9

Beauty.

toy →	Beauty
4 →	9
0 →	6
2 →	3

Total Beauty → 18

$T = \cancel{0}\ \cancel{1}\ \cancel{2}\ 3$

Sale end time

$$\begin{array}{lccccc} & \checkmark & \checkmark & & & \checkmark \\ & 0 & 1 & 2 & 3 & 4 \\ A[\,] = & 3 & 1 & 3 & 2 & 3 \\ B[\,] = & 6 & 5 & 3 & 1 & 9 \end{array}$$

Beauty.

key $\rightarrow$ Beauty

$1 \rightarrow 5$

$4 \rightarrow 9$

$0 \rightarrow 6$

Ans $\rightarrow$ 20

$T = \cancel{0}\ \cancel{1}\ \cancel{2}\ 3$

e.g 2 :-

$$\begin{array}{ccc} & \checkmark & \checkmark \\ & 0 & 1 \\ A = [& 1, & 2] \\ B = [& 3, & 1500] \end{array}$$

key $\longrightarrow$ Beauty

$0 \longrightarrow 3$

$1 \longrightarrow 1500$

Ans $\rightarrow$ 1503

$T = \cancel{0}\ \cancel{1}\ 2$

idea Buy Everything in asc order of time.

Sale end time

$$\begin{array}{lcccccccc} & \checkmark & \times & \checkmark & \times & \checkmark & \checkmark & \checkmark & \checkmark \\ & 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ A \rightarrow [& 1, & 3, & 3, & 3, & 5, & 5, & 5, & 8] \\ B \rightarrow [& 5, & 2, & 7, & 1, & 4, & 3, & 8, & 1] \end{array}$$

Beauty

$T = \cancel{0}\ \cancel{1}\ \cancel{2}\ \cancel{3}\ \cancel{4}\ \cancel{5}\ 6$

min heap

$$\boxed{\begin{array}{ccc} 5 & \cancel{2} & 7 \\ 4 & 3 & 8 \end{array}}$$

idea :- [Correct an incorrect step taken]

① Sort the items in ascending order of time. → $n \log n$

```
minheap mh;
T = 0;
for (i = 0; i < n; i++) {          → nlogn
    if (T < A[i]) {
        mh.insert(B[i]);
        T++;
    }
    else {
        if (B[i] > root of heap)
            extractMin();
            mh.insert(B[i]);
        }  // t--, t++
    }
}
```

③ Remove all the elements from the heap and add them and return

T.C → $O(n \log n)$

S.C → $O(n)$

Candy Distribution

There are N students with their marks. The teacher has to give them candies such that
a) Every student should have at least one candy
b) Students with more marks than any of his/her neighbours have more candies than them.

Find minimum candies to distribute.

e.g 1)

	1	5	2	1	
Cand ->	1	~~1~~ ~~2~~ 3	~~1~~ 2	1	Ans -> 7

e.g 2)

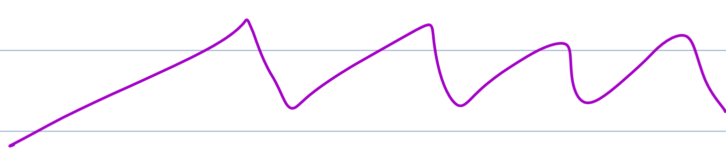
	8	10	6	2	
	1	~~1~~ ~~2~~ 3	~~1~~ 2	1	Ans -> 7.

e.g 3)

	4	4	4	4	
	1	1	1	1	Ans -> 4

e.g 4)

1	6	3	1	10	12	20	5	2	
1	3	2	1	2	3	4	2	1	=> 19 Ans.



arr[] →	1	6	3	1	10	12	20	5	2
	1	~~2~~ 3	~~1~~ 2	1	2	3	4	2	1

$a < b > c$

$x \qquad x+1$

1) ∀ i, c[i] = 1;

2) ∀ i, if (A[i] > A[i-1]) { c[i] = c[i-1]+1 }   → 1 to n

3) ∀ i, if (A[i] > A[i+1]) → {
   if (c[i] < c[i+1]+1) {
   c[i] = c[i+1]+1
   }
   }

   ↓ n to 0

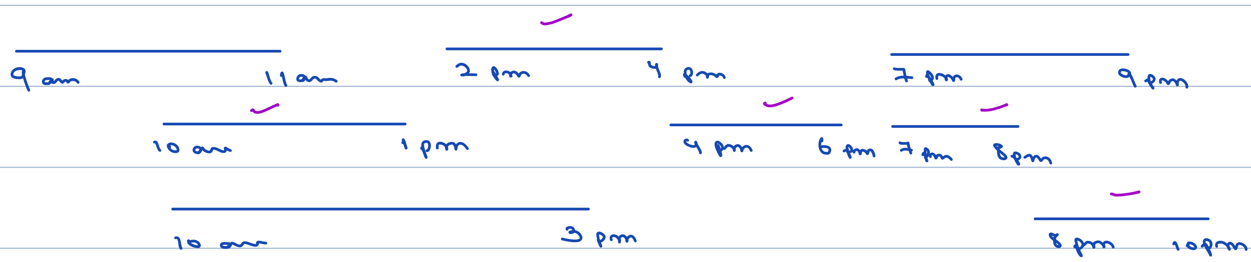
4) return sum(c[]);

T.C → O(n)

S.C → O(n)

Ques Maxm, Jobs :-

Given N jobs with their start and end times. Find the maximum number of jobs that can be completed if only one job can be done at a time.

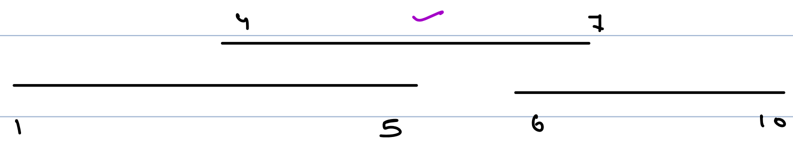


Total jobs = 5

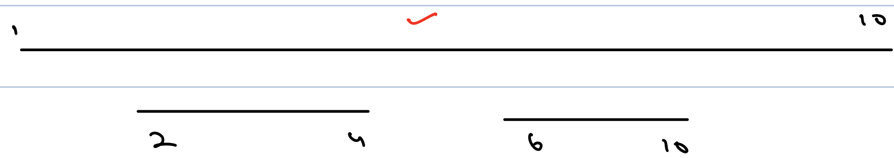
Greedy Ideas :-

1) Shortest Duration jobs.
2) Shortest Start time.
3) Shortest End time.

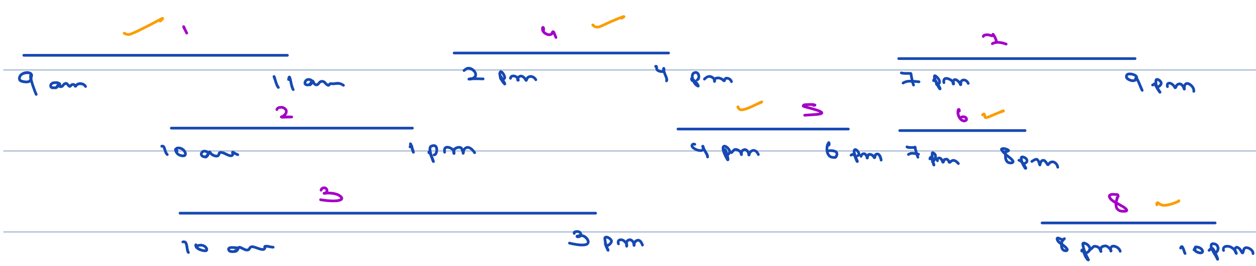
1) Shortest Duration Job :- ✗



2) Shortest Start time ✗



3) Shortest End time :- → Start early + shorter duration



Ans = 5.

PriorityQueue<Pair> pq = new ____ ( );

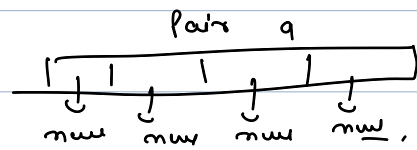
pq. add (new Pair ( ));

pq.getMin( );

→ comparable

```
class Pair {
    int s;
    int e;
    Pair (x, y) {
        s = x;
        e = y;
    }
}
```

	0	1	2	3
s =	[1	5	8	7]
e =	[2	10	10	11]




```
int solve (int[] s, int[] e) {

    int n = s.len;  → 4
    Pair a[] = new Pair[n];

    for (i = 0; i < n; i++) {
        a[i] = new Pair(s[i], e[i]);
    }

    Arrays.sort (a, (Pair u, Pair v)
                  -> (u.e - v.e));

    prevJobEnded = a[0].E;

    ans = 1;

    for (i = 1; i < n; i++) {
        Pair p = a[i];
        if (p.s >= prevJobEnded) {
            ans++;
            prevJobEnded = p.e;
        }
    }

    return ans;
}
```

T.C → O(n log n)

S.C → O(n)