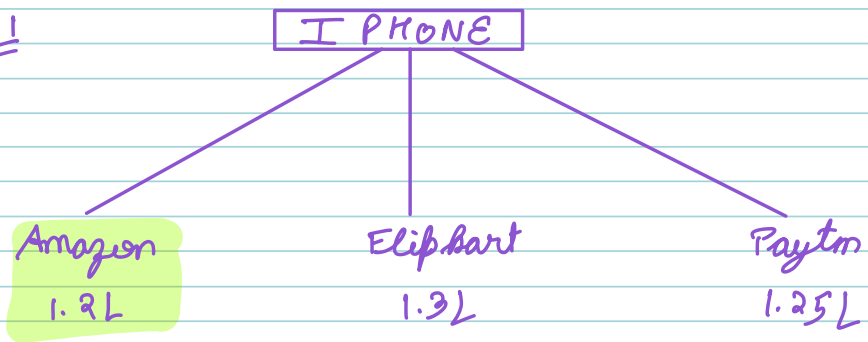
Greedy

Question :- what is Greedy ?

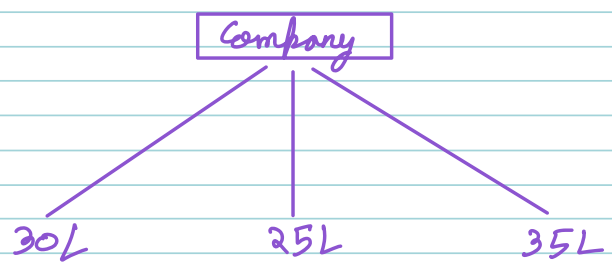
↳ Maximize our profit
OR
Minimize the cost

Eg 1



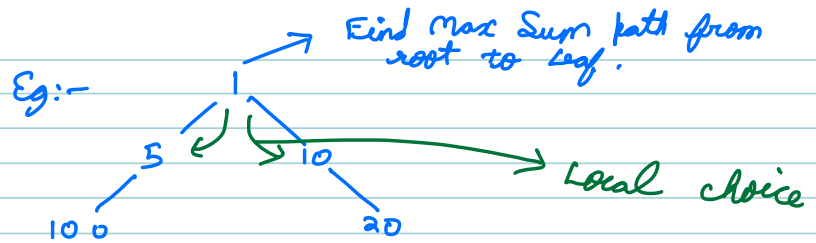
↳ Best Choice → min price

Eg 2



* while being greedy we need to take care of all the factors.

DEFINATION :- It is an approach to solve Optimisation problems by making locally optimal choices



↳ Here Local Choices should contribute toward Global choice.

Question :-

In the recent expansion into grocery delivery, Flipkart faces a crucial challenge in effective inventory management. Each grocery item on the platform carries its own expiration date and profit margin, represented by arrays A[i] (expiration date for the ith item) and B[i] (profit margin for the ith item). To mitigate potential losses due to expiring items, Flipkart is seeking a strategic solution. The objective is to identify a method to strategically promote certain items, ensuring they are sold before their expiration date, thereby maximizing overall profit. Can you assist Flipkart in developing an innovative approach to optimize their grocery inventory and enhance profitability?

A[i] -> expiration time for ith item
B[i] -> profit gained by ith item

Time starts with **T = 0**, and it takes 1 unit of time to sell one item and the item **can only be sold if T < A[i].**

Sell items such that the sum of the **profit by items is maximized.**

Assume it takes 1 unit of time to sell every item. Find max profit.

T < Expiry time.

Ex:-

	0	1	2	3	4
A[] = (Expiry date)	3	1	3	2	3
B[] = (Profit margin)	6	5	3	1	4

T = ∅ X X 3

DRY RUN

move from Left to Right

item		Profit
0	⟶	6
2	⟶	3
4	⟶	9
Profit	⟶	18

* We are not sure if profit is Max or not.

IDEA 1 :- Start Selling in the order of Max profit

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

T = ∅ X X 3

items	Profit
4	9
0	6
2	3
TP ⟶	18

⇒ Max Profit

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

(III) (I) (II)

T = ∅ X X 3

items	Profit
1	5
4	9
0	6
TP ⟶	20

Quiz :- what is the maximum profit we can achieve:-

A[] = [1 2]
B[] = [3 1500]

T=0 X 2

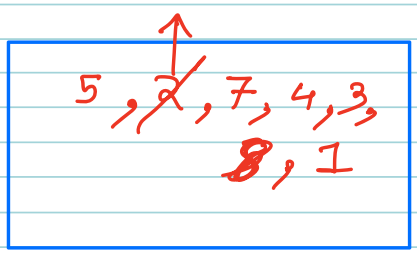
item	Profit
0	3
1	1500
TP →	1503

Conclusion :- selecting max profit first is not helping.

IDEA2 :- Sort everything on the Basis of Time.

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

T= 0 X X 3 4 5 6



Q Can we trade any previous item with the last 3 Expiry?

↳ No, the Profit will be less

- Should we trade anything with second S ?
  - ↳ yes
- Then which item ?
  - ↳ The one with min profit
- which DS should be used ?
  - ↳ Min Heap.
- How we will sort on the basis of time?
  - ↳ By forming pair & then sorting on the Basis of expiry in pair.

PSEUDO CODE

① Sort the pair array on the Basis of time/expiry.

```
Minheap mh;
T=0;
for (i =0; i < n; i++){
    if ( A[i] >T ){
        mh. insert ( B[i]);
        T++;
    } else {
        if ( B[i] > mh. getMin()){
            extractMin();
            mh. insert ( B[i]);
        }
    }
```

}

Profit = (Remove all elements from heap & find sum)

Quiz? :- What is the time & space complexity of this question?

TC - O(N log N)
SC - O(N)

Question 2 :- There are N students with their marks. The teacher has to give them candies such that

a.) Every student should have atleast one candy.

b.) Students with more marks than any of his/her neighbours have more candies than them.

Find minimum candies to distribute.

Ex :- 1 5 2 1

Candies → 1 ~~1~~ ~~1~~ 1 ← Because of Condition one.

~~2~~ 2

3

→ 1 3 2 1

Ans = 7

Ex :- 8 10 6 2

Candies → 1 2 3 2 1 Ans ⇒ 7

Quiz 3 :- what is the minimum no. of candies teacher has to use if the marks are :

[4, 4, 4, 4, 4]

Candies → 1 1 1 1 1 ⇒ Ans = 5

Quiz 4 :- what is the minimum no. of candies teacher has to use if the marks are :

[1, 6, 3, 1, 10, 12, 20, 5, 2]

Candies 1 2 3 2 1 2 3 4 2 1

Ans = 19

IDEA :-

Step 2 → ← Step 3

[1, 6, 3, 1, 10, 12, 20, 5, 2]

Candies :- 1 2 3 2 1 2 3 4 2 1

Step 1 :- Give 1 candy to all.

Step 2 :- Start from Left & Satisfy people.

C++ i, if (A[i] > A[i-1]) { C[i] = C[i-1] + 1; }

Step 3:- Start from right & satisfy people.

```
(∀ i , if (A[i] > A[i+1]) { if ( C[i] <= C[i+1]){
                                C[i] = C[i+1] + 1;
                            }
                          }
```

Step 4:- return Sum(C[]);

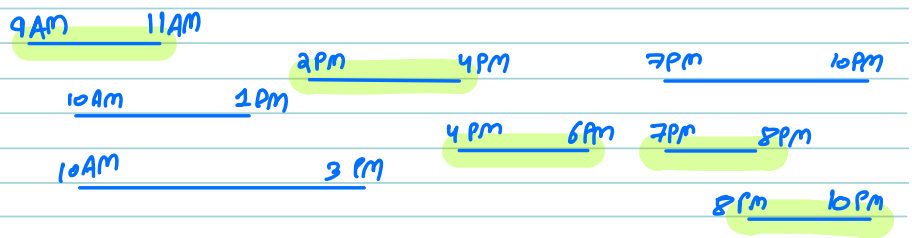
TC → O(N)

SC → O(N)

Question :- Maximum Jobs

Given N jobs with their start & end times. Find the maximum no. of jobs that can be completed if only one job can be done at a time.

Ex :-



Total Jobs → 5

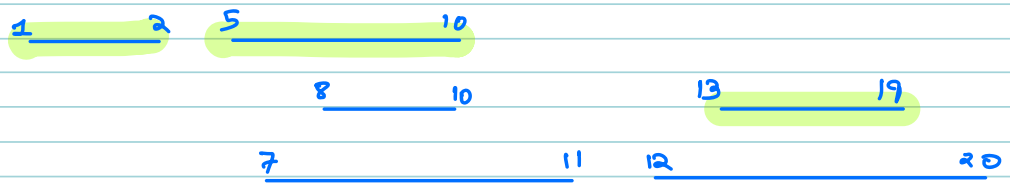
GREEDY IDEAS

1. Shortest duration job.
2. Earliest Start time.
3. Earliest End time.

Quiz :- Max Jobs that can be completed by 1 person given Start & End time :-

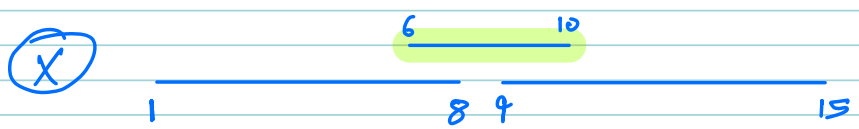
$S = [1, 5, 8, 7, 12, 13]$

$E = [2, 10, 10, 11, 20, 19]$

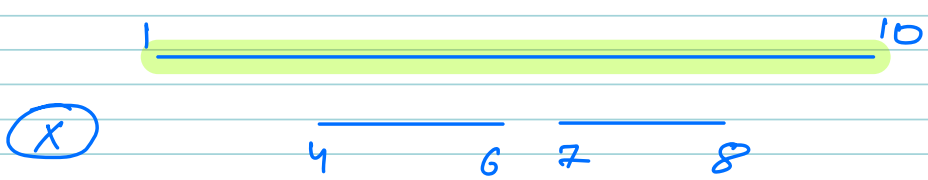


Ans = 3

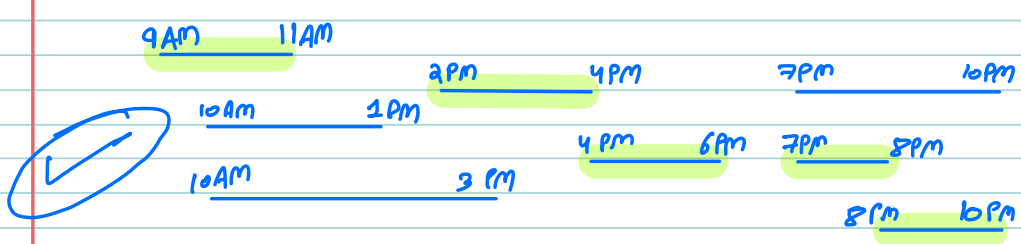
IDEA 1 :- Shortest Duration job :-



IDEA 2 :- Earliest Start time.



IDEA 3 :- Earliest end time



Ans = 5

Q. Why this is working?

↳ Earliest end time ⇒ start early + has shortest duration

PSEUDO CODE

```
        0  1  2  3
S = [ 1  5  8  7 ]
E = [ 2  10 10 11 ]
```

```
class Pair {
    int s;
    int e;
    Pair (x, y) {
        s = x;
        e = y;
    }
}
int Solve ( int[] S , int[] e ){
    int n = s.length;
    Pair[] a = new int[n];
    for ( i = 0; i < n; i++){
        a[i] = new pair( S[i], e[i]);
    }
    Arrays.Sort (a, (Pair a, Pair b) → {a.e - b.e} );
```

comparator OR Comparable ←

```
    PrevEndTime = a[0].e;
    ans = 1;
    for( i=1; i<n; i++){
        Pair P = a[i];
        if( P.s >= PrevEndTime){
            ans++;
            prevEndTime = P.e;
        }
    }
    return ans;
}
```

TC → O(n logn)

SC → O(n)