GREEDY

Good Evening

To do list

Greedy Introduction

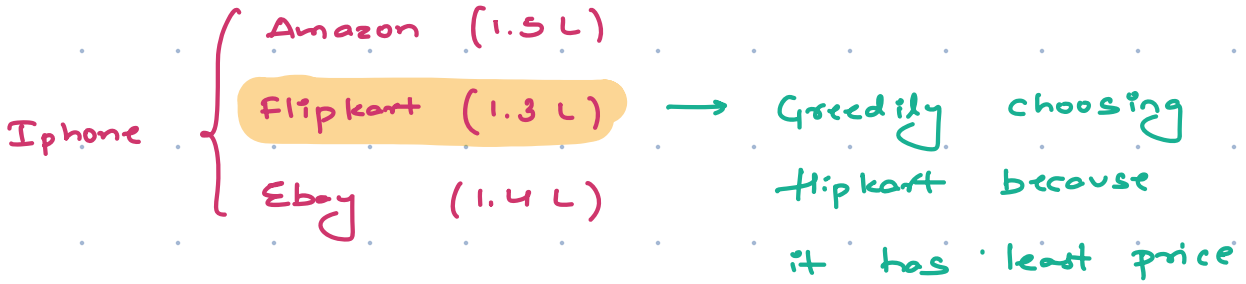
01. Effective Inventory Management

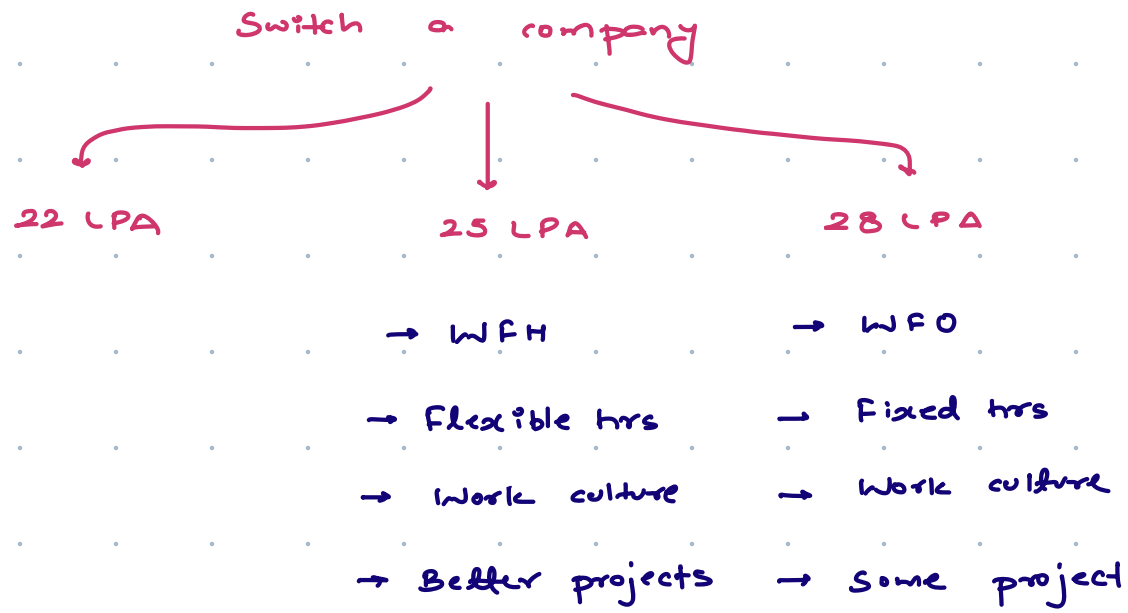
02. Distribute candy

03. Activity Selection

* Greedy → Approach that deals with maximizing profit or minimising the loss

→ Solving a problem by using the best possible option at that time.





Multiple factors involved → Decision making becomes difficult.

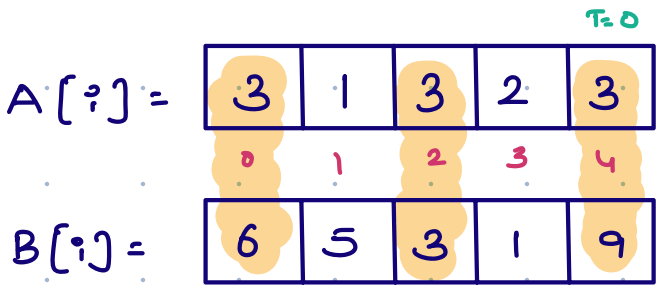
2. Problem 1 Flipkart's Challenge in Effective Inventory Management

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.**

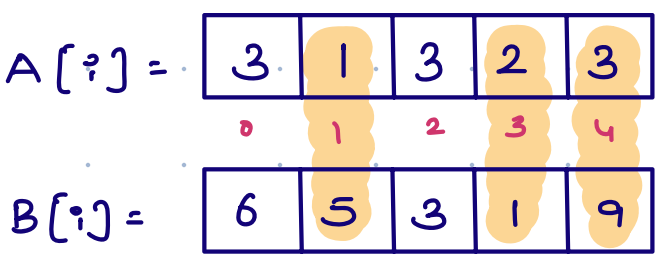


Idea 1 → sell the item with maximum profit

profit = 9 + 6 + 3 ⇒ 18

T = Ø ✗ ✗ 3

Idea 2 → sell the product with minimum expiry



T = Ø ✗ ✗ 3

profit = 0 + 5 + 1 + 9 ⇒ 15

Idea 3 → Selling all the items will help in attaining maximum profit.

→ Sell the items with minimum end time

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

T = ∅ ✗ ✗ 3

profit : 5 + 6 + 9

→ 20

A[] = { 1 2 }

B[] = { 3 1500 }

profit = 1503

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

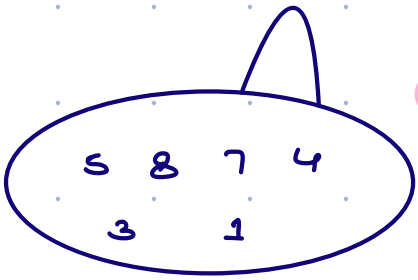
T = ∅ ✗ ✗ ✗ ✗ ✗ 6

profit = 0 + 5 + 2 + 7 + 4 + 3 + 8 + 1 ⇒ 28

→ Correct the incorrect product that we have sold in past.

* Use min heap

						↓		
A[]=	1	3	3	3	5	5	5	8
B[]=	5	2	7	1	4	3	8	1



T= ✗ ✗ ✗ ✓ ✗ ✗ 6

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

= 28

C[] =

Exp	profit
1	5
3	2
3	7
3	1

* Code →

01. Sort Array on the basis of end time

02. PriorityQueue<I> pq = new PriorityQueue // min heap

```
Time = 0
for (i=0; i<n; i++){
    if (Time < c[i][0]){
        pq.add(c[i][1]);
        Time = Time + 1;
    }
    else{
        if (c[i][1] > pq.peek()){
            pq.remove();
            pq.add(c[i][1]);
        }
    }
}
```

TC: O(nlogn)

SC: O(n)

03. Empty priority queue & add it to answer.

Chocolate Distribution

Given N students marks, assign chocolate to all students in such a way that :-

01. Each student gets atleast one chocolate

02. if (A[i] > A[i-1]) → chocolate assigned to i^{th} student should be more than $(i-1)^{th}$ student.

03. if (A[i] > A[i+1]) ⟶ chocolate assigned to i^{th} student should be more than $(i+1)^{th}$ student.

Calculate min no. of chocolates to assign to all N students.

A[] = { 1 5 2 1 6 }

chocolate = { 1 1 1 1 1 } ✗

chocolate = { 1 5 2 1 6 } ✓ → 15 chocolates

chocolate = { 1 3 2 1 2 } → 9 chocolates

A[ ] = { 4 4 4 4 4 }

chocolate = { 1 1 1 1 1 } → 5 chocolates

A[ ] = { 3 100 60 }

chocolate = { 1 2 1 } → 4 chocolates

Idea →

if (A[i] > A[i-1]) A[i] = A[i-1] + 1

if (A[i] > A[i+1]) A[i] = A[i+1] + 1

A[ ] = { 3 6 2 8 10 }

left = { 1 2 1 2 3 }

right = { 1 2 1 1 1 }

---

chocolate = 1 + 2 + 1 + 2 + 3 ⇒ 9 chocolates

A[ ] = { 3 6 9 11 7 2 }

left = 1 2 3 4 1 1

right = 1 1 1 3 2 1

---

chocolates = 1 2 3 4 2 1 ⇒ 13 chocolates

A[] = { 1 6 3 1 10 12 20 5 2 }

left = 1 2 1 1 2 3 4 1 1

right = 1 3 2 1 1 1 3 2 1

---

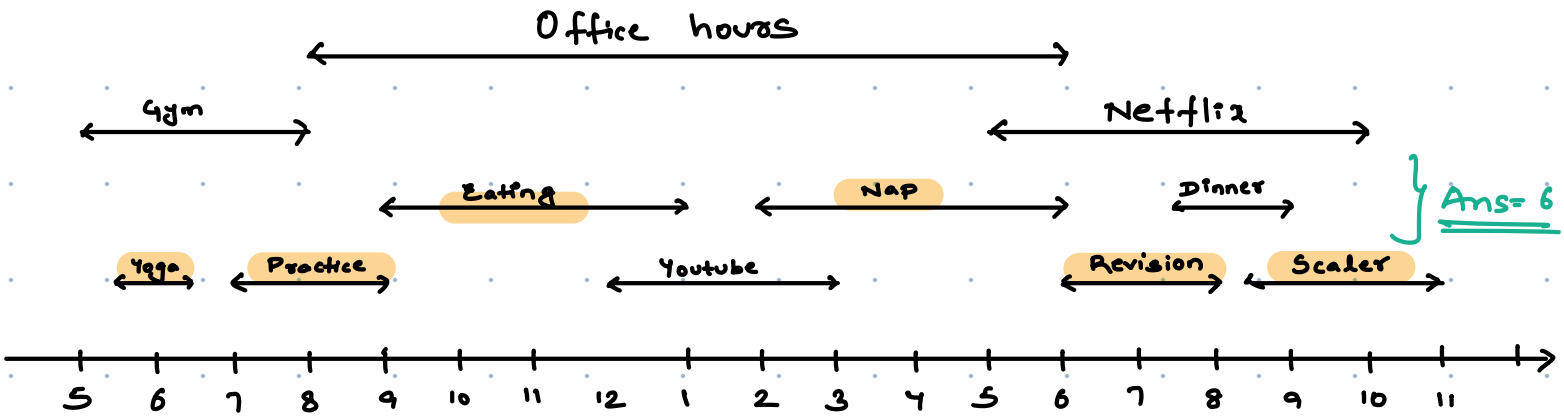
chocolate = 1 + 3 + 2 + 1 + 2 + 3 + 4 + 2 + 1 = 19 chocolate

Code

10:14 pm → 10:24 pm

Activity Selection

Flipkart
Microsoft

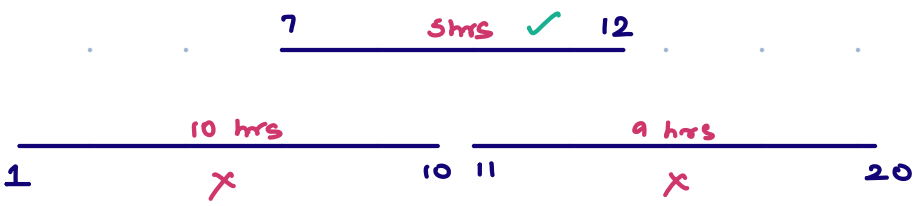


Q **Max tasks** which one can do.

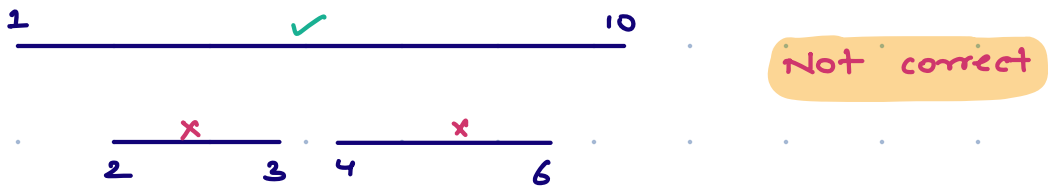
Note 1 → Once we start, we need to finish task.

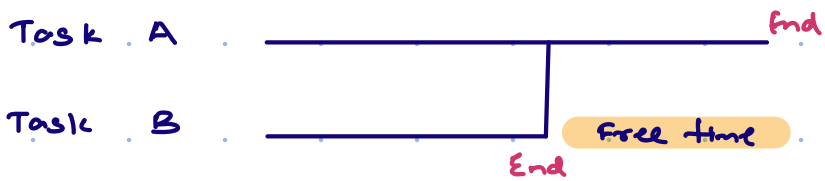
Note 2 → Overlapping tasks are not allowed at any given time.

Idea 1 → Activities with less duration



Idea 2 → Sort on basis of start time

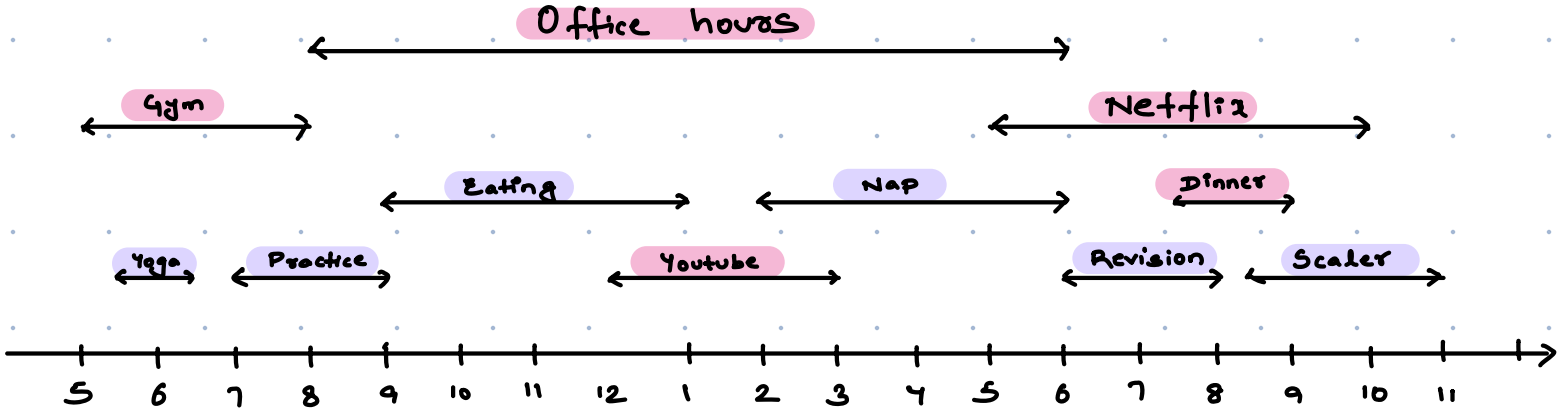




S = { 1 5 8 7 12 13 }

E = { 2 10 10 11 20 19 }

Ans = 3



Approach → Choose task which is finishing first, so that we have more time for next task

st end → sort 2d arr based on end time → st end

st	end
5	8
5:30	6:30
7	9
9	13
8	18
12	15
14	18
18	20
19:30	21

st	end	Last activity
5:30	6:30	6:30
5	8	
7	9	9
9	13	13
12	15	
8	18	
14	18	18
18	20	20
19:30	21	

Ans = 5

01. Sort the jobs on the basis of end time

```
count = 1
lastEndtime = c[0][1]

for (i=1; i<n; i++){
    if (c[i][0] ≥ lastEndtime){
        count = count + 1;
        lastEndtime = c[i][1];
    }
}
return count;
```

TC: O(nlogn)
SC: O(n)

Incase, you have to create a 2D array

* Famous question
  - → Meeting Rooms 1, 2, 3
  - → Minimum platforms
  - → Job scheduling
  - → Activity selection
  - → Gas Station
  - → Candy distribution

→ Fractional knapsack

→ Minimum no. of arrows to burst balloon.

→ Light bulb switch

Duplicate numbers

01. HM

02. Sort the array

03 Modify the given array.