Greedy

Madhan Kumar M S
Abhishek Sharma
Akansh Nirmal
Anuj chandil
Balaji S K
Bhaveshkumar
Burhan
Hemant Kumar
KULDEEP PATIDAR
PREETHAM
Purusharth A
Rajendra
Sanket Giri
Saurabh Ruikar
Shani Jaiswal
sharath r
Shradha Srivastava
Shreya Gupta
Sneha L
Sridhar Hissaria
Subhashini
Sumit Adwani
Suyash Gupta
Vasanth
Vimal Kumar
Yugesh v

AGENDA:

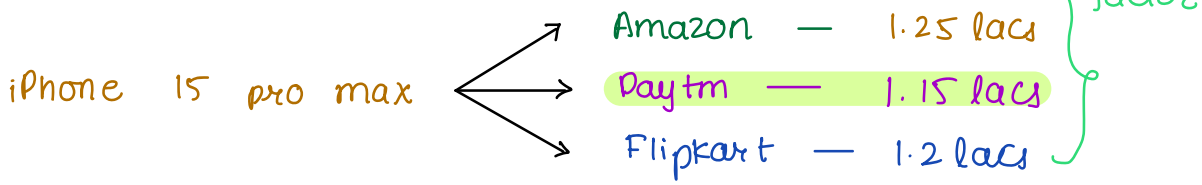
- Inventory Management
- Candy Distribution
- Maximum Jobs

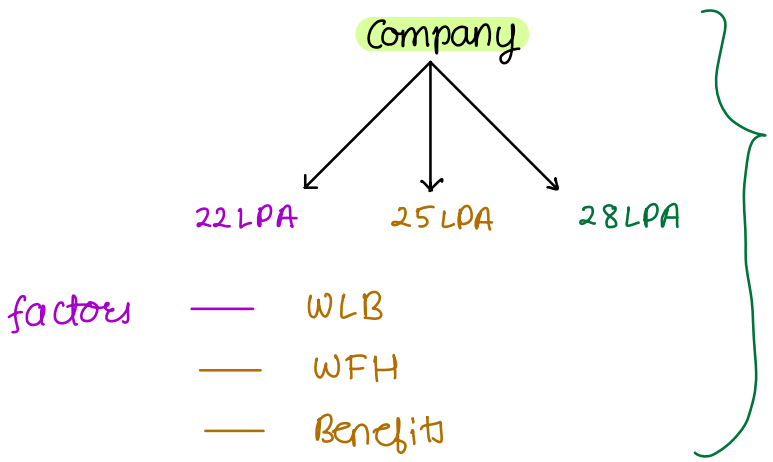
what is greedy ?

Maximising the profit
Minimising the loss
} Based on some factors

Price

Greedy on one factor





Flipkarts 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)
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 maximising overall profit. Can you assist Flipkart in developing an innovative approach to optimize their grocery inventory and enhance profitability?

Selling any item takes a unit of time.

A[] → 3 1 3 2 3 { Expiring }
B[] → 6 5 3 1 9 { Profit }

Always try to pick the item with highest profit.

t = 0 ⟶ 9
t = 1 ⟶ 6 profit = 18
t = 2 ⟶ 3

---

t = 0 ⟶ 5
t = 1 ⟶ 6 profit = 20
t = 2 ⟶ 9
t = 3 ⟶

A = 1 2
B = 3 1500

t = 0 ⟶ 3
t = 1 ⟶ 1500

Idea 2 Pick the items with least expiry time.

A[] → 1 2 3 3 3

B[] → 5✓ 1✓ 6✓ 3✓ 9✓

t → 0 5

t → 1 ~~X~~ ~~3~~ 9.

t → 2 6

A = 1 3 3 3 5 5 5 8

B = ~~5~~ ~~2~~ ~~7~~ 1 ~~4~~ ~~3~~ ~~8~~ ~~1~~

t = 0 → 5

t = 1 → ~~2~~ 8

t = 2 → 7

t = 3 → 4

t = 4 → 3

t = 5 → 1

A 1 1 1

B ~~8~~ (2) (1)

t = 0 → 8

t = 1

A 1 1 1

B ~~1~~ ~~(2)~~ (8)

t = 0 → ~~1~~ ~~2~~ 8

Pseudocode

1> sort based on expiration time → NlogN → AB

minHeap → SC : O(N)

```
for i ---> 0 to N-1 {                          → N times
    exp    = AB[i][0]
    profit = AB[i][1]

    minHeap.insert(profit)
    if ( minHeap.size() > exp ) {              → logN oper.
        minHeap.extractMin()
    }
}
```

print (sum of minHeap) → O(N)

TC : O(NlogN)

SC : O(N)

A	1	1	1
B	1	2	8

Candy Distribution

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

i) Every student have atleast one candy

ii) Student with more marks than one of its neighbour should have more candies

Find the min candies to distribute

	0	1	2	3
A[] =	1	5	2	1
candy	1	3	2	1

ans = 7

QUIZ:

	0	1	2	3	4
A[] =	4	4	4	4	4
candy	1	1	1	1	1

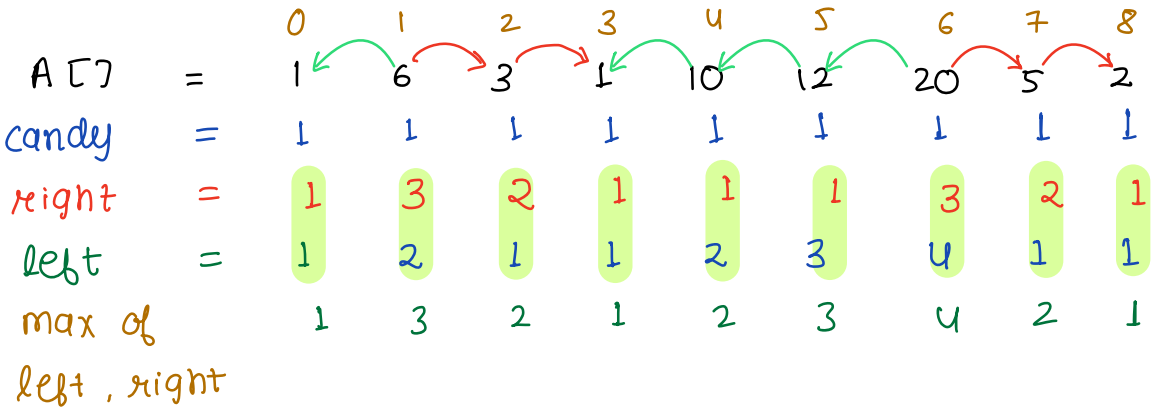
ans = 5

	0	1	2	3
A[] =	8	10	6	2
candy =	1	3	2	1

ans = 7

Step 1 ⟶ Distribute 1 candy to everyone

```
if ( A[i] > A[i-1] ) {
    left = candy[i-1] + 1
}
if ( A[i] > A[i+1] ) {
    right = candy[i+1] + 1
}
```



Pseudocode

candy [1 ..... 1]
n size

TC : O(N)
SC : O(N)

```
for i ⟶ 1 to N-1 {
    if ( A[i] > A[i-1] ) {
        candy[i] = candy[i-1] + 1
    }
}
```

```
for i ⟶ N-2 to 0 {
    if (A[i] > A[i+1]) {
        candy[i] = max ( candy[i],
                         candy[i+1] + 1 )
    }
}
```

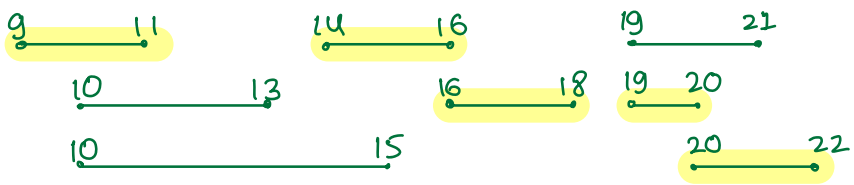
return sum of candy

Break : 22 : 42

Maximum Jobs

[start, end)

Given N jobs with their **start** & **end** time
Find the maximum jobs that can be completed
if only one job can be done at a time

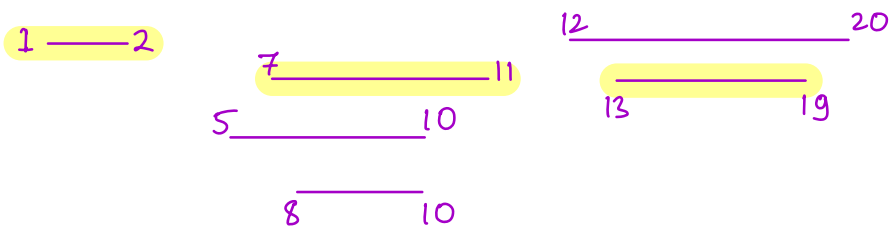


ans = 5

QUIZ

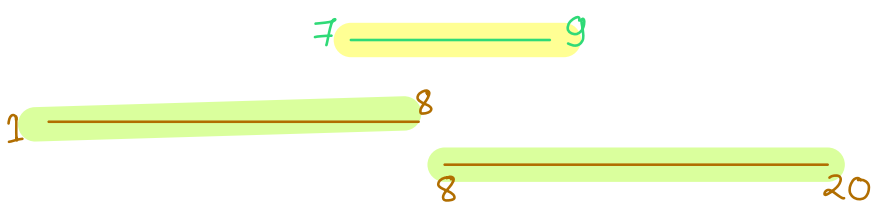
S []	=	1	5	8	7	12	13
E []	=	2	10	10	11	20	19

ans = 3



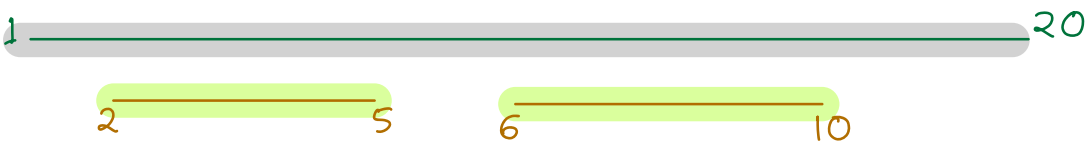
Ideas

Idea 1 ⟶ Pick jobs with smaller duration



Picking the smallest job first doesn't work

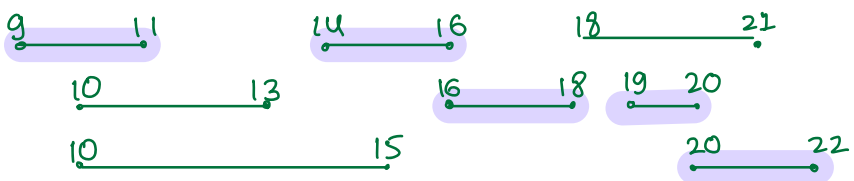
Idea 2 ⟶ Sort based on starting time



The above approach doesn't work ∵ any = 2
pick 2 — 5
6 — 10

start + duration ? = end

Idea 3 ⟶ Sort based on ending time



S =	9	10	10	14	16	19	18	20
E =	11	13	15	16	18	20	21	22

last end time = ~~11~~ ~~16~~ ~~18~~ ~~20~~ 22

Pseudocode

S =	9	10	10	14	16	19	18	20
E =	11	13	15	16	18	20	21	22

```
Sort  S,E  based  on  end  time
jobs = 0
last = -∞
for  i ⟶ 0 to N-1 {
        start, end = SE[i][0], SE[i][1]
        if (start >= last) {
              jobs++
              last = end
        }
}
```

SC : O(N)

TC : O(N log N)

```
print(jobs)
```