Greedy

TABLE OF CONTENTS

iphone → single parameter (price) ✓

shoes → multiple parameters ←

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 maximised.

$$\begin{array}{lcccccc} & & 0 & 1 & 2 & 3 & 4 \\ A[\ ] & \rightarrow & [\ 3 & 1 & 3 & 2 & 3\ ] \\ B[\ ] & \rightarrow & [\ \textcircled{6} & 5 & \textcircled{3} & 1 & \textcircled{9}\ ] \\ T \rightarrow & & 1 & & 2 & & 0 \end{array}$$

$profit = 9 + 6 + 3 = \underline{18}$

$$\begin{array}{lcccccc} & & 0 & 1 & 2 & 3 & 4 \\ A[\ ] & \rightarrow & [\ 3 & 1 & 3 & 2 & 3\ ] \\ B[\ ] & \rightarrow & [\ 6 & 5 & 3 & 1 & 9\ ] \\ T \rightarrow & & 2 & 0 & & & 1 \end{array}$$

$profit = 5 + 9 + 6 = \underline{20}$

$$\begin{array}{lcc} & 0 & 1 \\ A = [& 1 & 2\ ] \\ T \rightarrow & 0 & 1 \\ B = [& 3 & 1500\ ] \end{array}$$

$profit = \underline{1503}$

Best case → Sell everything.

Sort wrt expiry (A[i], B[i]).

```
         ✓   ✓   ✓        ✓    ✓         ✓
         0   1   2   3    4    5    6    7
A = [    1   3   3   3    5    5    5    8 ]
B = [    5   2   7   1    4    3    8    1 ]
T →    0   1   2        3    4         5
```

high profit → replace it with lowest profit selected item. (lowest profit selected → minHeap)

profit < all selected items profit ⇒ ignore it.

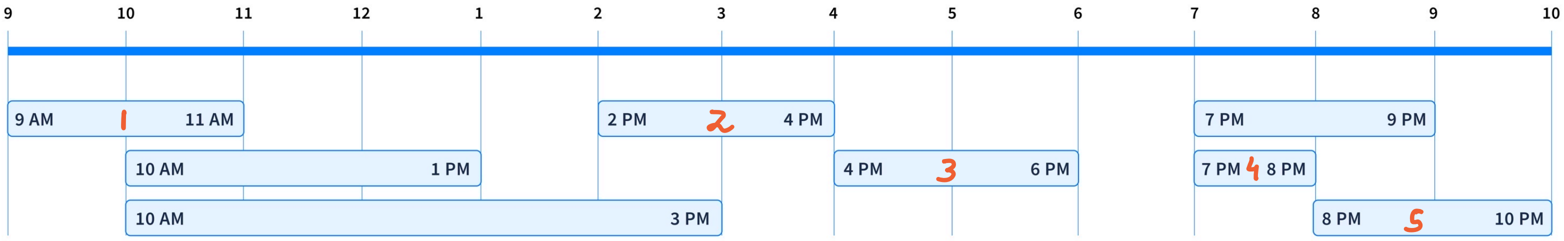
ans = 5 + 7 + 4 + 3 + 8 + 1 = 28

```
// Sort (A, B) pair wrt A[i].
  T = 0            ans = 0
  for i → 0 to (N-1) {
      if ( T < A[i]) {        // Sell item
          ans += B[i]              T++
          heap.insert (B[i])
      } else if ( B[i] > heap.root) {    // Replace item
          ans -= heap.extractMin ()
          ans += B[i]
          heap.insert (B[i])
      }
  }
```

heap.root → min selected profit

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

< Question > : Given N jobs with their start and end-time. Find the max jobs that can be completed if only one job can be done at a time.



Ans = $\underline{5}$

$\boxed{S[i] >= E[i-1]}$

Quiz :

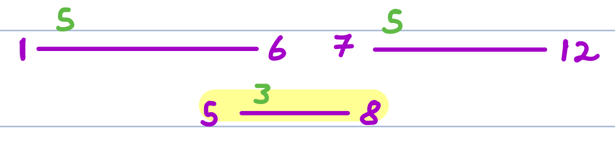
✓ ✓ ✓

S = [1 5 8 7 12 13]

E = [2 10 10 11 20 19]

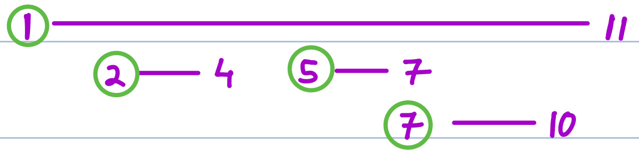
Ans = $\underline{3}$

1. Sort on the basis of duration →



Ans = ~~1~~ → $\underline{2}$

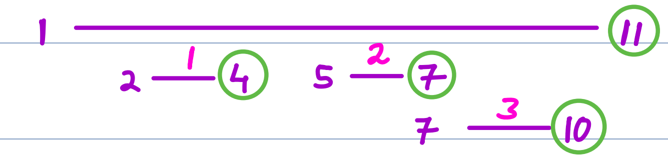
2. Sort on the basis of start time →



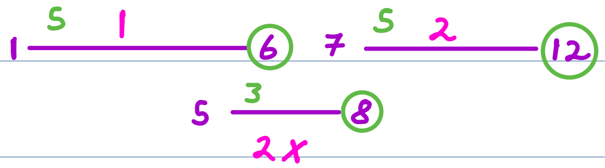
Ans = ~~1~~ → $\underline{3}$

3. Sort on the basis of end time →

start early + short duration ⇒ ending early



Ans = 3



Ans = 2

```
// sort wrt end time in asc. order
  ans = 1          end = E[0]
 for i → 1 to (N-1) {
      if (S[i] >= end) {
            ans++
           end = E[i]
      }
 } return ans
```

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

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

a. Every student have at least one candy

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

Find ~~maximum~~ min candies to distribute. $(i-1) \leftarrow i \rightarrow (i+1)$

$$\begin{array}{cccccc} & & 0 & 1 & 2 & 3 \\ \textbf{1.} & A[\,] \rightarrow & [\,1 & 5 & 2 & 1\,] \\ & c \rightarrow & 1 & 3 & 2 & 1 \end{array}$$

Ans = 7

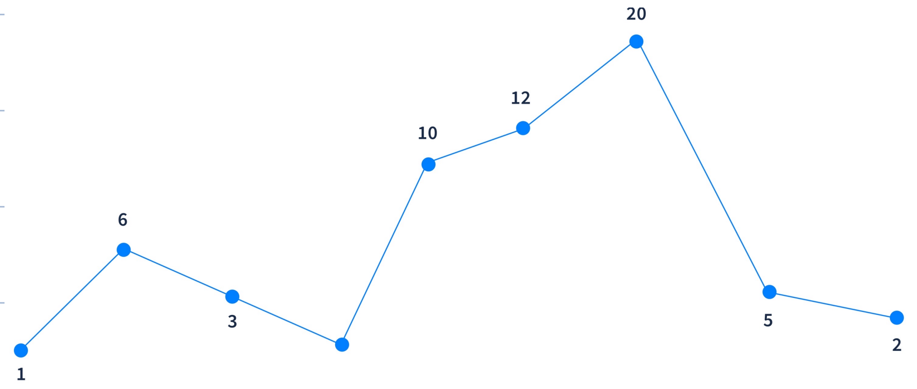
$$\begin{array}{ccccccc} & & 0 & 1 & 2 & 3 & 4 \\ \textbf{2.} & A[\,] \rightarrow & [\,4 & 4 & 4 & 4 & 4\,] \\ & c \rightarrow & 1 & 1 & 1 & 1 & 1 \end{array}$$

Ans = 5

$$\begin{array}{cccccc} & & 0 & 1 & 2 & 3 \\ \textbf{3.} & A[\,] \rightarrow & [\,8 & 10 & 6 & 2\,] \\ & c \rightarrow & 1 & 3 & 2 & 1 \end{array}$$

Ans = 7

$$\begin{array}{ccccccccccc} & 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ A = & [\,1 & 6 & 3 & 1 & 10 & 12 & 20 & 5 & 2\,] \end{array}$$



	0	1	2	3	4	5	6	7	8
A[ ] →	[1	6	3	1	10	12	20	5	2]
c →	1	~~1~~ ~~2~~ 3	~~1~~ 2	1	~~1~~ 2	~~1~~ 3	~~1~~ 4	~~1~~ 2	1

1) ∀ i, c[i] = 1

2 → a) if (A[i] > A[i-1]) ⇒ $\boxed{c[i] > c[i-1]}$

c[i] = c[i-1] + 1

b) if (A[i] > A[i+1]) ⇒ $\boxed{c[i] > c[i+1]}$

c[i] = max (c[i], c[i+1] + 1)

```
∀i, c[i] = 1                ans = 0

for i → 1 to (N-1) {
    if (A[i] > A[i-1])      c[i] = c[i-1] + 1
}

for i → (N-2) to 0 {
    if ( A[i] > A[i+1])     c[i] = max (c[i], c[i+1] + 1)
    ans += c[i]
}

ans += c[N-1]               TC = O(N)     SC = O(N)

return ans
```