Buy an Iphone

Amazon	Flipkart	Paytm
1.3 lakhs	1.25L	1.15L

Maximising profit or minimising losses.

Recruiter Hiring for Google

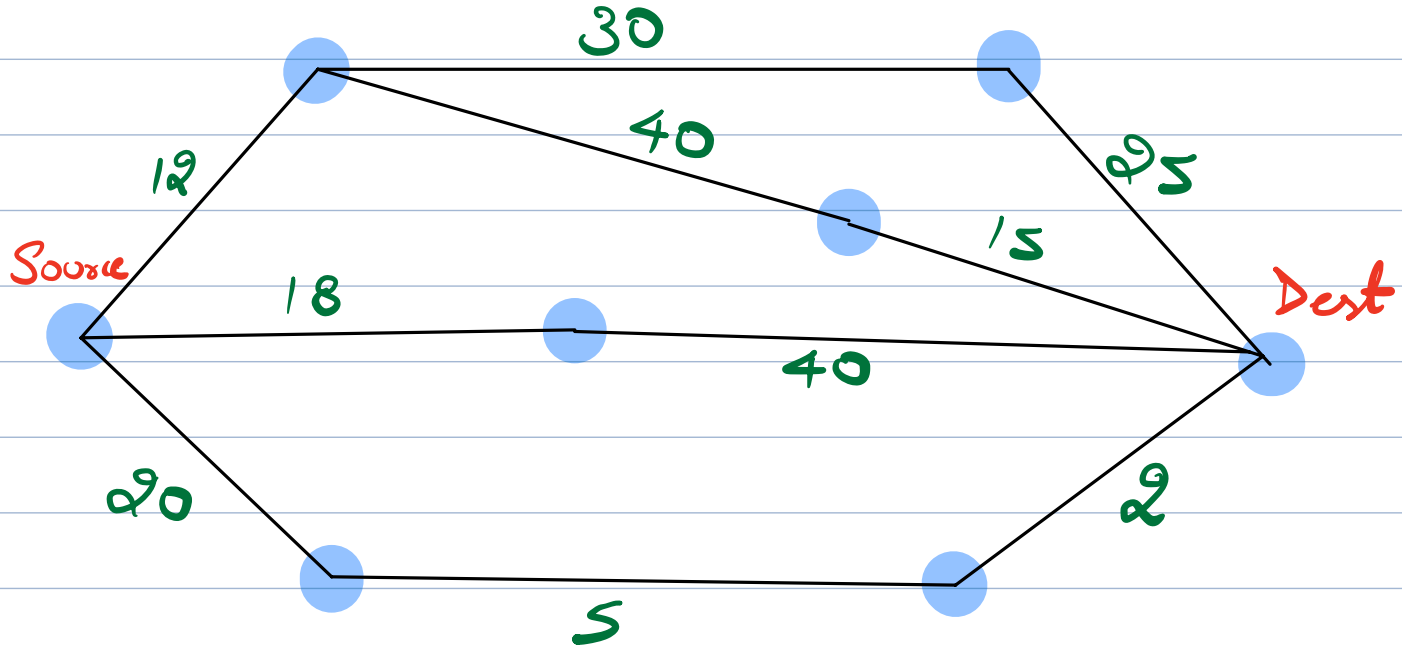
Goal: Minimize overall cost to company
Target: Hire 1 candidate every month.

Jan	Feb	March	Dec.
~~36L~~	35L	18L	
~~45L~~	42L	22L	
~~52L~~	48L	31L	
28L		29L	

Optimising for every month $\Rightarrow$ Optimise the annual hiring.

⇒ Local optimal choices leads to Global optimal solution

⇒ Current choice does not impacts future decisions.



In grocery business, flipkart faces a crucial challenge

Each grocery item has its expiry date, profit margin

⇓

N items ⇒ A ⇒ A[i] denotes the expiry date of ith item.

B ⇒ B[i] denotes profit margin

of ith item

$$A = \begin{matrix} 0 & 1 & 2 & 3 & 4 \\ [3, & 1, & 3, & 2, & 3] \end{matrix}$$

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

Help flipkart implement an algorithm that helps decide which item to promote at a given time maximising overall profit

$(T = 0)$

Const: 1) It takes 1 unit of time to sell 1 product.

2) Only sell an item at $T < A[i]$

$$A = \begin{matrix} \cancel{0} & \cancel{1} & 2 & \cancel{3} & \cancel{4} \\ [3, & 1, & 3, & 2, & 3] \end{matrix}$$

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

Cons 1

Time	Item	Profit
0	4	9
1	0	6
2	2	3
~~3~~		18

Cons 2

Time	Item	Profit
0	1	5
1	0	6
2	4	9
~~3~~		20

Ans

$A = [1, 2]$ (0, 1)

$B = [3, 1500]$ (0, 1)

Time	Item	Profit
0	1	1500
1		1500

Time	Item	Profit
0	0	3
1	1	1500
		1503

	0	1	2	3	4	5	6	7
A =	1	3	3	3	5	5	5	7

	0	1	2	3	4	5	6	7
B =	5	2	7	6	4	3	8	9

Soln

T	Item	Profit
0	0	5
1	~~1~~ 3	~~2~~ 6
2	2	7
3	4	4
4	~~5~~ 6	~~7~~ 8
→ 5	7	9
6		

Min Heap

Code

1) Sort the items on the basis of expiry time. NlogN

```
class Item {
    int expiry;
    int profit;
}
```

```
Item[] items = new Items[N];
```

Add all elements one by one.

Use comparator to sort.

```
MinHeap< int > heap;
T=0;
```

```
for (i= 0; i<N; i++ ){
```

N

```
if ( T < items[i].expiry ) {
    heap.insert(items[i].profit);
    T++;
} else {
    if (items[i].profit > root of Heap) {
        heap.extractMin();
        heap.insert(items[i].profit);
    }
}
```

logN

⇒ Iterate over Heap & calculate sum of all items.

$$T.C. = O(N \log N)$$

$$S.C. = O(N)$$

There are N students in a class. After their exam results, teacher wants to distribute candies to them.

Const: 1) Every student must have atleast 1 candy.

2) Students with more marks should be given more candies than the neighbours.

Find the minimum candies required.

$$A = \begin{matrix} 0 & 1 & 2 & 3 & 4 \\ [1, & 5, & 2, & 1, & 10] \end{matrix}$$

$$Candy = [1, 2, 2, 1, 2]$$

Min # Candies = 9

Quiz $A = \begin{matrix} 0 & 1 & 2 & 3 & 4 \\ [4, & 4, & 4, & 4, & 4] \end{matrix}$

C: [1, 1, 1, 1, 1]

Ans = 5

$A = [\underset{0}{8}, \underset{1}{10}, \underset{2}{6}, \underset{3}{2}]$

C: [1, 2, 2, 1]

Ans = 7

Quiz $A = [\underset{0}{1}, \underset{1}{6}, \underset{2}{3}, \underset{3}{1}, \underset{4}{10}, \underset{5}{12}, \underset{6}{20}, \underset{7}{5}, \underset{8}{2}]$

$\Rightarrow$ CL: 1, 2, 1, 1, 2, 3, 4, 1, 1

CR: 1, 3, 2, 1, 1, 1, 3, 2, 1

Ans: {1, 3, 2, 1, 2, 3, 4, 2, 1}

```
if (A[i] > A[i-1]) {
    CL[i] = CL[i-1] + 1
}
```

Code

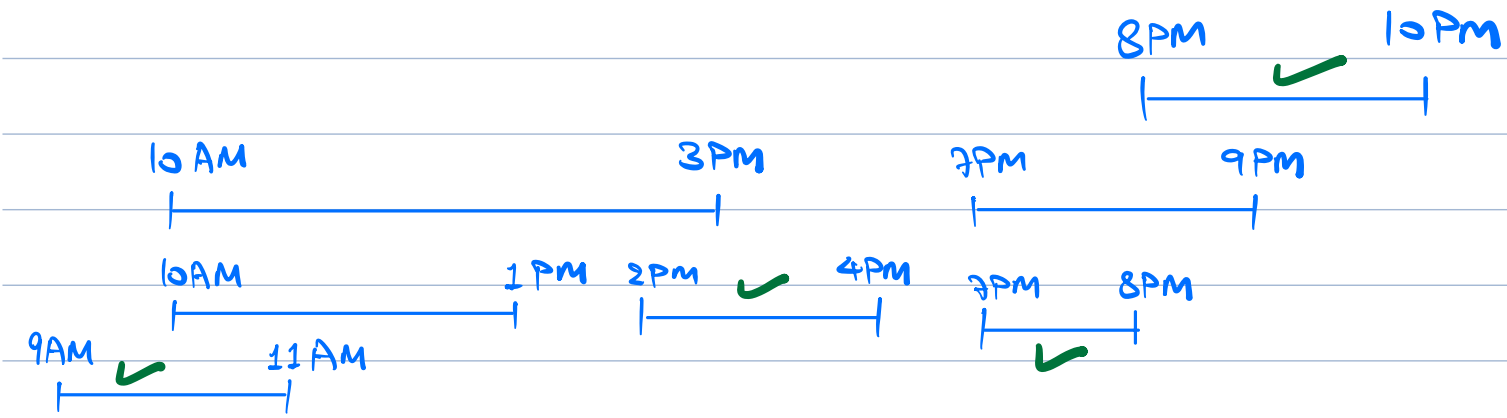
```
CL[N]          CL[0] = 1;
CR[N]          CR[N-1] = 1;

for (i=1; i<N; i++){

        if (A[i] > A[i-1]){
            CL[i] = CL[i-1] + 1;
        } else {
            CL[i] = 1;
        }
}

for (i=N-2; i>=0; i--){

        if (A[i] > A[i+1]){
            CR[i] = CR[i+1] + 1;
        } else {
            CR[i] = 1;
        }
}
ans = 0;
for (i=0; i<N; i++){
        ans = ans + max(CL[i], CR[i]);
}
```

T.C. = O(N)

S.C. = O(N)

Given N jobs ⇒ Start & End time.

Find the maximum no. of jobs that can be completed if only one job can be done at a time.

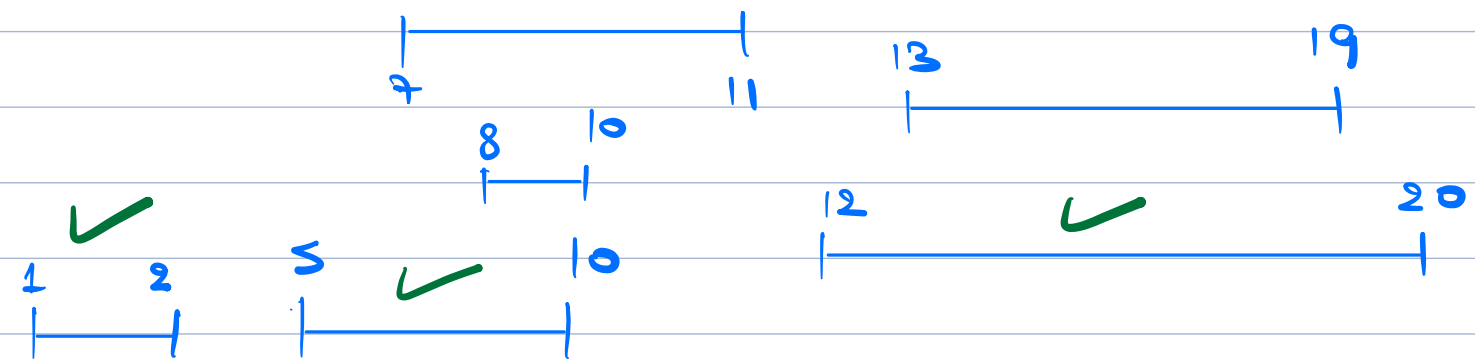


Ans = 4

Quiz

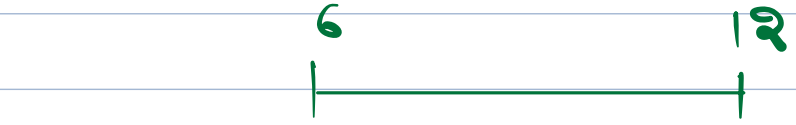
S = [1, 5, 8, 7, 12, 13]

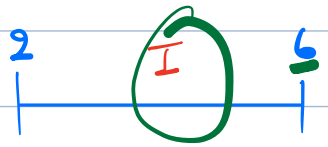
E = [2, 10, 10, 11, 20, 19]

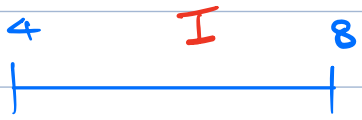


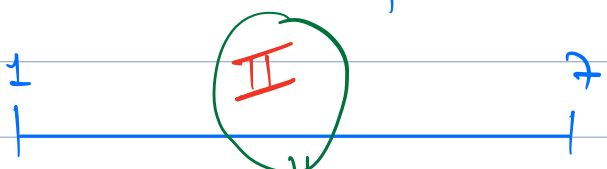
Ans = 2

Soln









Ends early ⇒ Greedily always select the possible jobs with lowest end time

Soln

1) Sort the jobs based on End time

2) Iterate & select as many jobs as possible.

$$S = [1, 5, 8, 7, 12, 13] \quad (0\ 1\ 2\ 3\ 4\ 5)$$

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

	0	1	2	3	4	5
S	1	5	8	7	13	12
E	2	10	10	11	19	20

Ans = 3

Code

1) Sort Based on End time — N log N

```
ans = 1
lastEndTime = E[0];

for (i=1; i<N; i++) {                    N
    if (S[i] >= lastEndTime) {
        ans++;
```

```
        lastEndTime = E[i];
    }
}
return ans;
```

T.C. = $O(N \log N)$

Contest ⇒ Friday

Trees, Heaps & Greedy

Wednesday ⇒ PS session + Hegs followup
