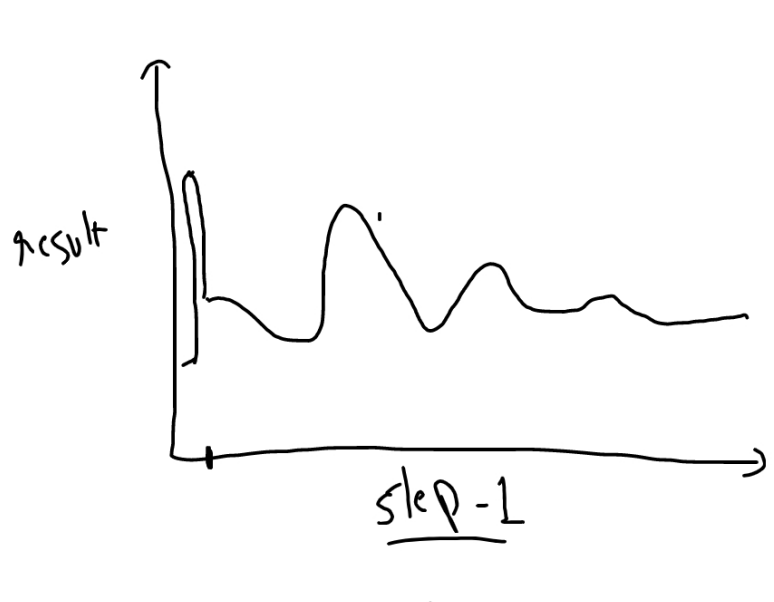
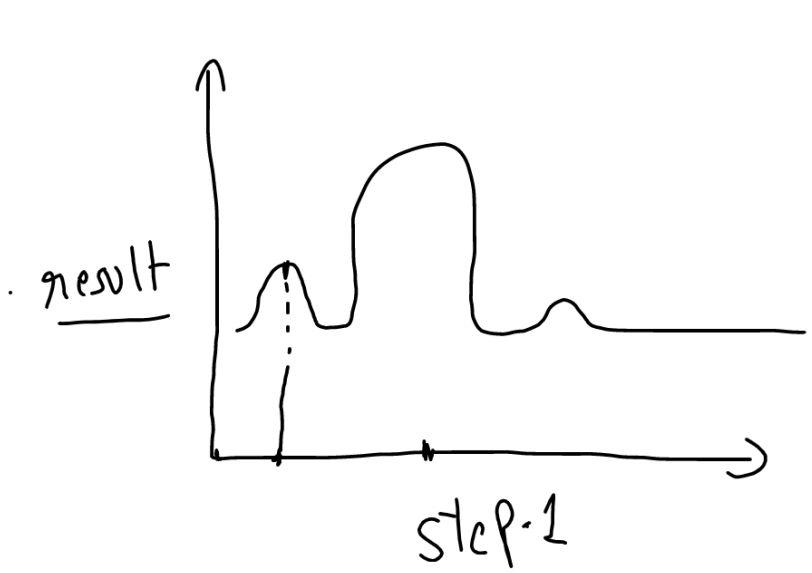
[1, 2, 2, 4, 5]   T = ~~8~~ ~~7~~ ~~5~~ 3

= 3 tasks can be done.



We can take a greedy step to achieve the best result



We should not take a greedy step & explore the wider picture

⇒ values: [60, 100, 120]
weights: [10, 20, 30]

W = 50

$\frac{1}{1}$ $\frac{0.5}{1}$ $\frac{0.33}{1}$ - - - -

50 − 30 = 20 units wt.

value: 60   100
wt: 10   20

$\overset{40}{\cancel{120}} \times \frac{2}{\cancel{3}} = 80$

$\overset{10}{\cancel{30}} \times \frac{2}{\cancel{3}} = 20$

Total value = 60 + 100 + 80 = 240
Total wt = 10 + 20 + 20 = 50 (≤ W)

Values: [60, 100, 120]
wts: [10, 20, 30]
(value/wt): [6, 5, 4]

W: ~~50~~ 40

$\left(\frac{20}{30}\right)$

20×4 = 80

$\frac{6}{1}\ \frac{6}{2}\ \frac{6}{3}\ \cdots\ \frac{6}{10}\ \frac{5}{11}\cdots\frac{5}{30}\ \frac{}{50}$

20

Value: [60, 100, 120]
wt: [10, 20, 30]
value/wt: [6, 5, 4]

W = ~~50~~ ~~40~~ ⇒ ~~20~~ 0

max_value: ~~0~~ ~~60~~ ~~160~~ $\boxed{240}$

$\left(\frac{W}{\text{wt of item}}\right) = \frac{20}{30}$

$\overset{40}{\cancel{120}} \times \frac{2}{\cancel{3}}$

30 → 120
20 → $\left(\frac{120}{30} \times 20\right)$

W = ~~50~~ ~~40~~ ~~10~~ 0

Value: $\begin{pmatrix}60\\10\end{pmatrix}\begin{pmatrix}100\\20\end{pmatrix}\begin{pmatrix}180\\30\end{pmatrix}$
wt:
value/wt: [6, 5, 6]

max_value = ~~0~~ ~~60~~ ~~240~~ 290

$\frac{10}{20}$ $\overset{50}{\cancel{100}} \times \frac{1}{\cancel{2}}$

⇒ (value/wt)   wt.

2.5

$\frac{5}{2} = 2.5$

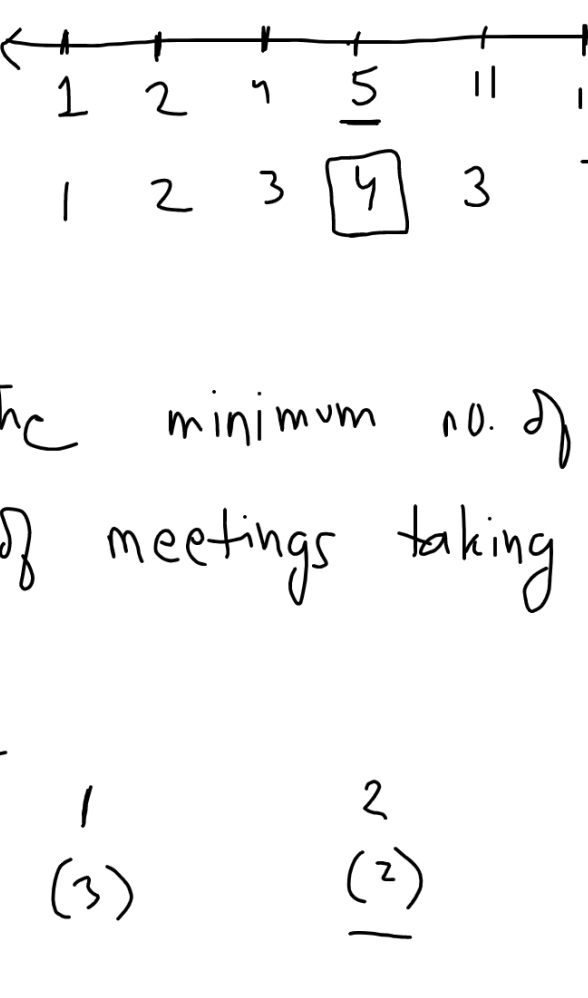
a: {5, 2} $\frac{5}{2} = 2$

b: {4, 2} $\frac{4}{2} = 2$

[{4,2}, {5,2}] ✗

$\frac{a.\text{first}}{a.\text{second}} > \frac{b.\text{first}}{b.\text{second}}$

[a.first * b.second > b.first * a.second]



Input 1:

```
A = [       [0, 30]
            [5, 10]
            [15, 20]
    ]
```

Input 2:

```
A = [       [1, 18]
            [18, 23]
            [15, 29]
            [4, 15]
            [2, 11]
            [5, 13]
    ]
```

— 1
— 2
— 2
⇒ 2.

1 2 3 4 5 6

1	2	3	4
(1,2)	(3,4)	(5)	(6)

(1,2)
(5,7)

(3,4)

1	2	3	4
(1, 2)	(3,4)	(5)	(6)

{~~time~~, 's/e'}

s	s	s	s	e	e	e	s	e	s	e	e	
1	2	4	5	11	13	15	15	18	18	23	29	
1	2	3	$\boxed{4}$	3	2	1	2	1	2	1	0	← ~~max~~ concurrent_meetings

[The minimum no. of conference rooms required is the maximum no. of meetings taking place at any point in time.]

1	2	3
(3)	(2)	(1)

⇒ 3 ⇐

(18, 23)
(18, 20)
(18, 22)

18

[18 — 23]

18 → 3
19 → 3
20 → 2
21 → 2
22 → 1
23 → 0

{ time, char denoting 's' or 'e'}

{ time, +1 / −1 }
+1: denotes that it is a start time
−1: end time

conc_meeting = ~~5~~ 6 5

[(15, 1)   (15, −1)] ⇒ [(15, −1), (15, +1)]
$\boxed{e}$ $\boxed{s}$

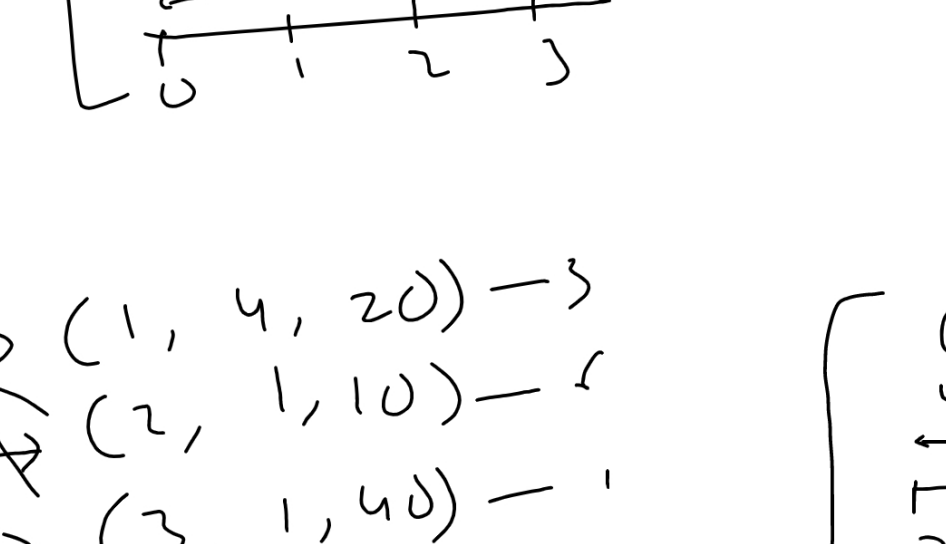
(2, 3)
(3, 4)

s	s	e	e
2	3	3	4
1	$\boxed{2}$	1	0

✗

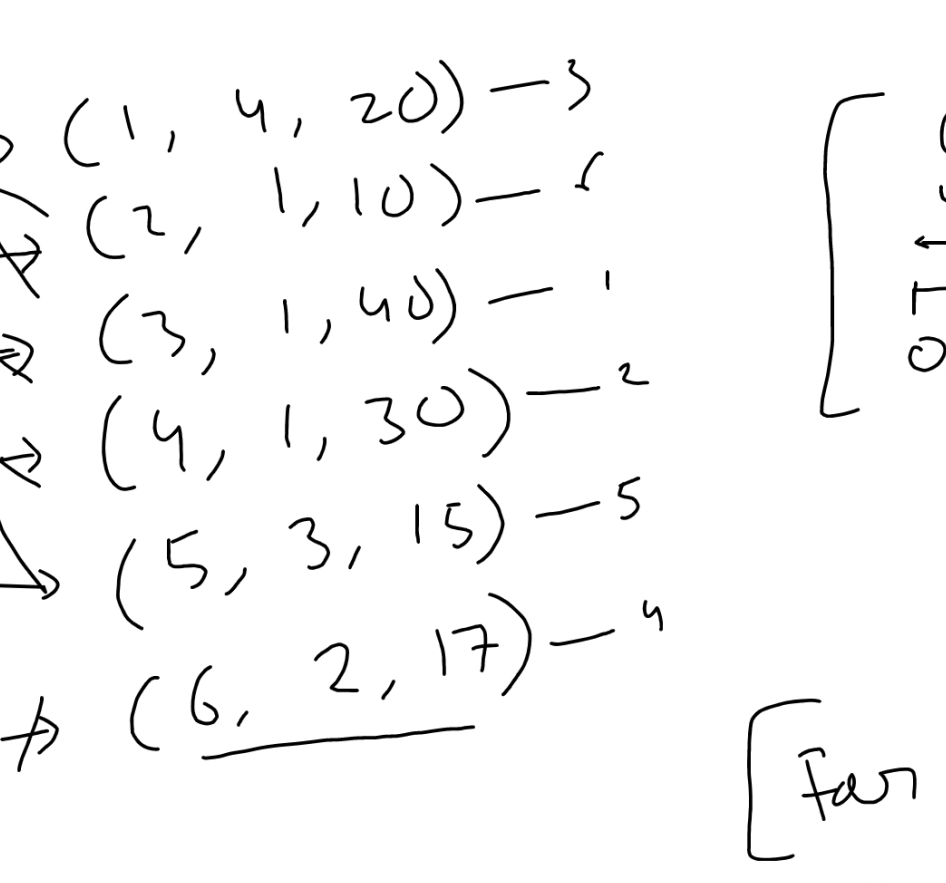
s	e	s	e
2	3	3	4
$\boxed{1}$	0	1	0

✓



```
Input:
N = 4
Jobs = {(1,4,20),(2,1,10),(3,1,40),(4,1,30)}
Output:
2 60
Explanation:
Job₁ and Job₃ can be done with
maximum profit of 60 (20+40).
```

t = 1   t = 0



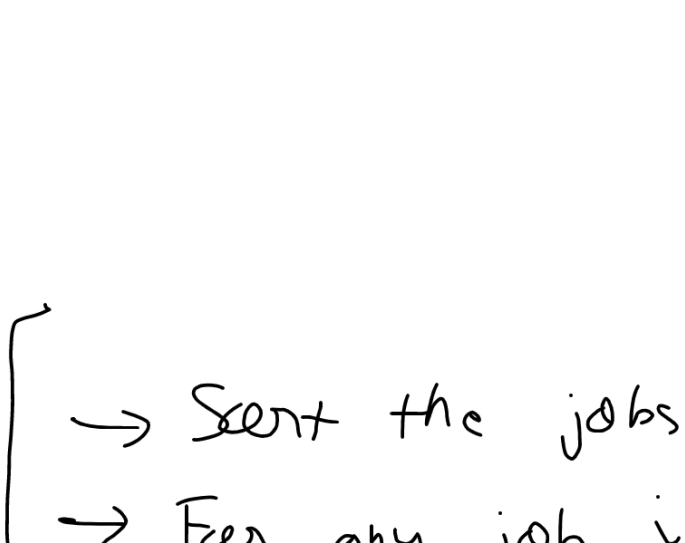
```
Input:
N = 5
Jobs = {(1,2,100),(2,1,19),(3,2,27),
        (4,1,25),(5,1,15)}
Output:
2 127
Explanation:
2 jobs can be done with
maximum profit of 127 (100+27).
```

t = 0   t = 1

Job[i] = {id, deadline, profit}

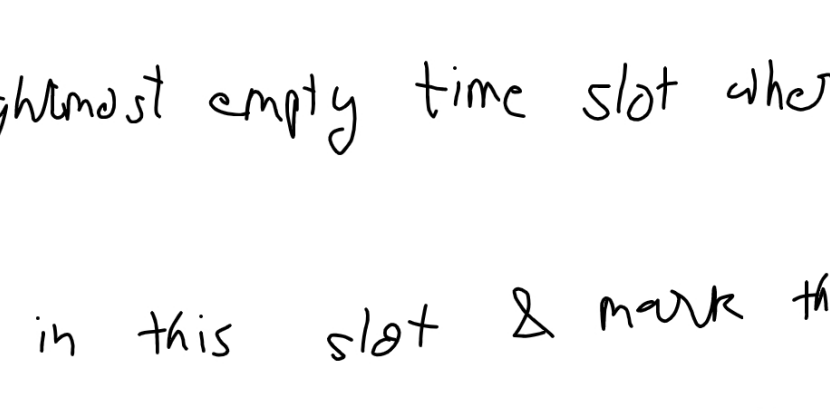
t: 0 → 1 ⇒ Job₃
t: 1 → 2 ⇒ Job₁

(t   t+1)



((1, 4, 20) (2, 4, 40))
((1, 4, 100) (2, 1, 20))
✗

(1, 4, 20) — 3
(2, 1, 10) — 6
(3, 1, 40) — 1
(4, 1, 30) — 2
(5, 3, 15) — 5
(6, 2, 17) — 4



[For any given job, we will search for the rightmost empty slot where it can be accomodated.]

- → Sort the jobs in dec. order of profits.
- → For any job i:
  - → Find the rightmost empty time slot where it can be accomodated.
  - → Put the job in this slot & mark the slot as filled
  - → If no such slot exist, skip the job & move ahead.