Today's Content

- Currency enchange
- fractional KnaPSacK
- Greedy Properties
- Activity Selection
- job Scheduling

→ to use the mondemon, if you can

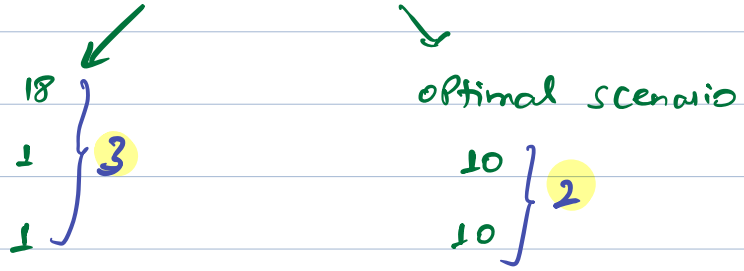
Indian currency: 1, 2, 5, 10, 20, 50, 100, 200, 500, 2000

Cash: 5548 → min number of coins/notes to get required cash?
↳ multiples of same coins are allowed.

notes/coins	count		leftout	
2000	2	→	1548	
500	3	→	48	will it always work? ↳ NOPE
20	2	→	8	
5	1	→	3	
2	1	→	1	
1	1	→	0	if Notes[i-1]*2 <= Notes[i]
	10			

Currency: 1 10 18

Money: 20 → min coins required?



Q) Fractional Knapsack

↳ you can consume K Kg of vegetables, you can eat any integral amount of item. Max Protein you can get?

Vegetables		Eating complete item Protein gained	Eat on the basis of total Protein	max 70 Kg consumption: min Kg	max Protein/Kg
Tomato	20 Kg	200 P			10 P/Kg - 20 - 200P
Apples	15 Kg	180 P	50 Kg → 250P	5 Kg → 100P	12 P/Kg - 15 - 180P
Onion	50 Kg	250 P	20 Kg → 200P	10 Kg → 150P	5 P/Kg
Chicken	10 Kg	150 P		12 Kg → 132P	15 P/Kg - 10 - 150P
Potato	25 Kg	200 P		15 Kg → 180P	
Mango	12 Kg	132 P		20 Kg → 200P	8 P/Kg - 8 - 64P
Seafood	5 Kg	100 P	450P	8 Kg → 64P	11 P/Kg - 12 - 132P
				826 P	20 P/Kg - 5 - 100P
					826 P

Greedy Properties:

① for min/max related Problems: → binary search, dp, greedy → greedy will, 2 pointers

② Based on what Parameter we want to apply greedy.

③ check the approach by coming up with counter examples.

↳ atleast one counter example.

Tomato → 5 kg → 50 ₹

Apple → 6 kg → 600 ₹

7 kg

using smallest wt. → Protein/kg

5 kg → 50 ₹ | 10 ₹/kg → 1 → 10 ₹

2 kg → 200 ₹ | 100 ₹/kg → 6 → 600 ₹

250 ₹ | 610 ₹

Ⓑ Solve Problems by "locally optimal choices" at (greedy)

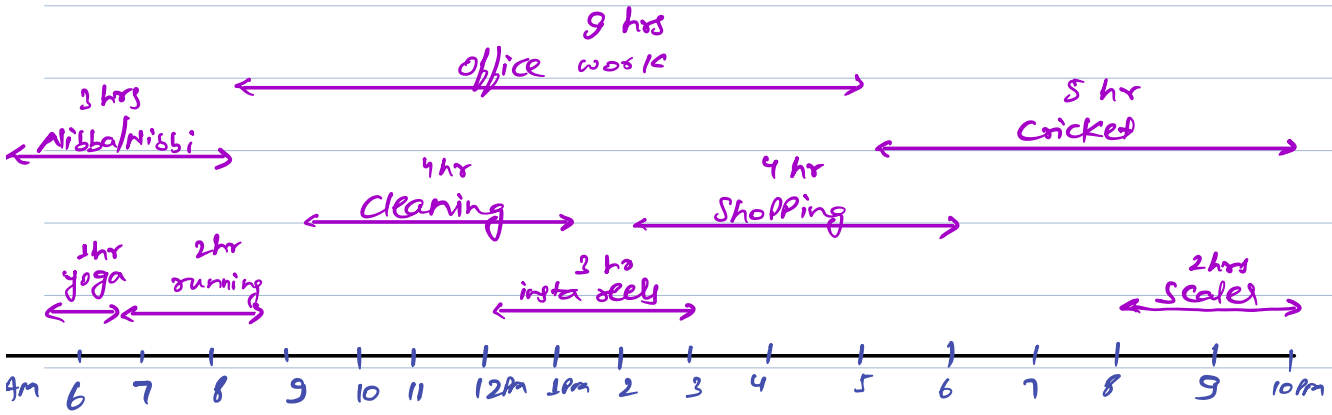
each step, and this is giving you the final best answer.

Q) Activity Selection

↳ Max Count of task you can do.

Note: (i) Start a task, we need to complete.

(ii) At any Point of time, single task.



(i) min duration task first.

yoga, Scales, running, insta reels

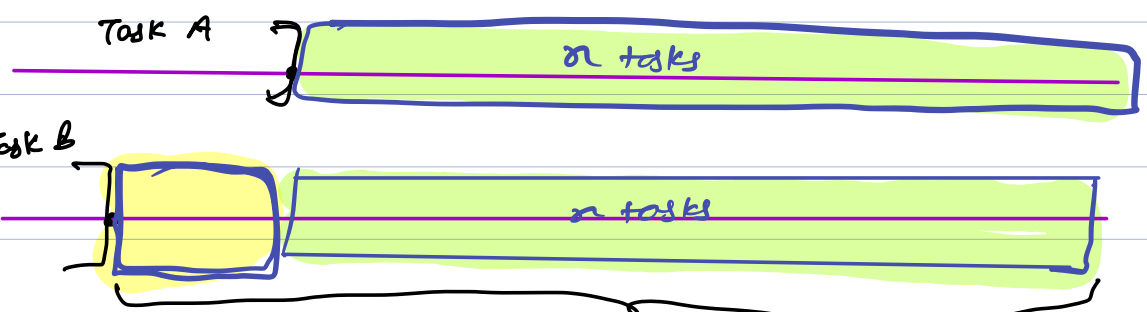
(ii) Starting early

Nibba/Nibbi, office work, Cricket

(iii) ending early

↳ yoga, running, Cleaning, Shopping, Scales

Correctness:



>= n tasks

By picking ending early tasks, we are leaving more slots to do more tasks.

Non-overlapping intervals ??

Algorithm:

(I) Sort all the tasks on the basis of end time.

(II) get all non-overlapping task.

Job Scheduling

- Given N tasks to complete
  - → Deadline assigned for each task, day on or before we can do task.
  - → Payment assigned to each task.
  - → on any given day we can perform only 1 task and each task take 1 day to finish.
  - → find max Payment we can get.

Parameter: Payment (i) + deadline (ii) → Payment + deadline (iii)

Ex1:

Job	deadline day	Payment
a	3	100
b	1	19
c	2	27
d	1	25
e	3	30

c (1) e (2) a (3) → 157

Job	deadline day	Payment
a	3	100
b	1	19
c	~~1~~	27
d	2	25
e	~~1~~	30

e (2) d (1) a

Ex2:

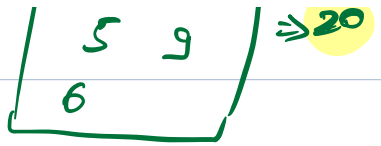
Job	deadline day	Payment
a	3	5
b	1	1
c	3	6
d	2	3
e	3	9

Job: b d a c e

deal: 1 2 3 3 3

Pay: 1 3 5 6 9

~~1~~ ~~2~~



Ex2:

Tasks:	1	2	3	4	5	6	7	8	9	10
dead:	2	1	1	1	4	5	4	5	5	2
money:	200	250	200	350	300	100	250	600	400	150

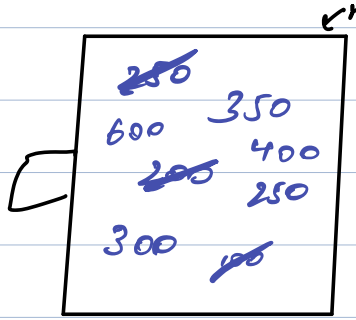
Sort on the basis of deadline

											i
dead:	1	1	1	2	2	4	4	5	5	5	
money:	250	200	250	200	150	300	250	100	600	400	

min heap

350

1900 ←



// Pseudo code :

↳ (i) Sort the data on the basis of deadline

↳ (ii) iterate and pick the jobs if Payment is better

Pain of discipline or Pain of regret

Eg: arr = {-1 -2 -3} b = 5

(-1)

{-1 -2 -3}
-4 -6
-6 -12

{-1 -2 -3}
-2 -4 -6
-3
-4

∴ -4 = ans

-4 < -1