Q. Currency Exchange

Indian currency :-

1 2 5 10 20 50 100 200 500 2000

Cash : 5548 Rs. : Min no. of coins/notes for this amount.

	Notes / Coins	amount left
2000	2	$5548 - 4000 = 1548$
500	3	$1548 - 1500 = 48$
20	2	$48 - 40 = 8$
5	1	$8 - 5 = 3$
2	1	$3 - 2 = 1$
1	1	$1 - 1 = 0$
	10	

⇒ Every denomination is atleast 2 times greater than the previous denomination.

⇒ x Rs

↳ $x > 500, 200$ but $x < 2000$

↳ i) $x - 500 \Rightarrow 1$ notes.

ii) $x - 2 \times 200 \Rightarrow 2$ notes

Currency : 1, 10, 18

$n = 20$ rs.

$20 - 18 \times 1 - 1 \times 2 = 0$

$\Rightarrow$ # of notes / coins = 3.

$20 - 10 \times 2 = 0$

} Greedy ✗

Q.

Food items	Protein content	Protein/kg
Tomato : 20 kg	200	10
Apple : 15 kg	180	12
Onion : 50 kg	250	5
Chicken : 10 kg	150	15
Potato : 25 kg	200	8
Mango : 12 kg	132	11
Seafood : 5 kg	100	20

→ We can pick max of 70 kg

→ We can pick 1 kg from each item.

→ Pick the items s.t we get max protein.

1) Pick the items based on the total max protein :-

50 kg Onion P : 250 + 20 kg Tomato P : 200 → 70 kg P : 450 ✗

2) Protein / Kg. 7 Kg of food item

	Seafood	Chicken	Apple	Mango	Tomato	Potato
Wt.	5	10	15	12	20	8 Kg
Protein	100	150	180	132	200	64

Total Protein = 826.

Greedy Properties :-

1) for optimisation related problem.
   ↳ min / max (cost / Profit / Coins ----)

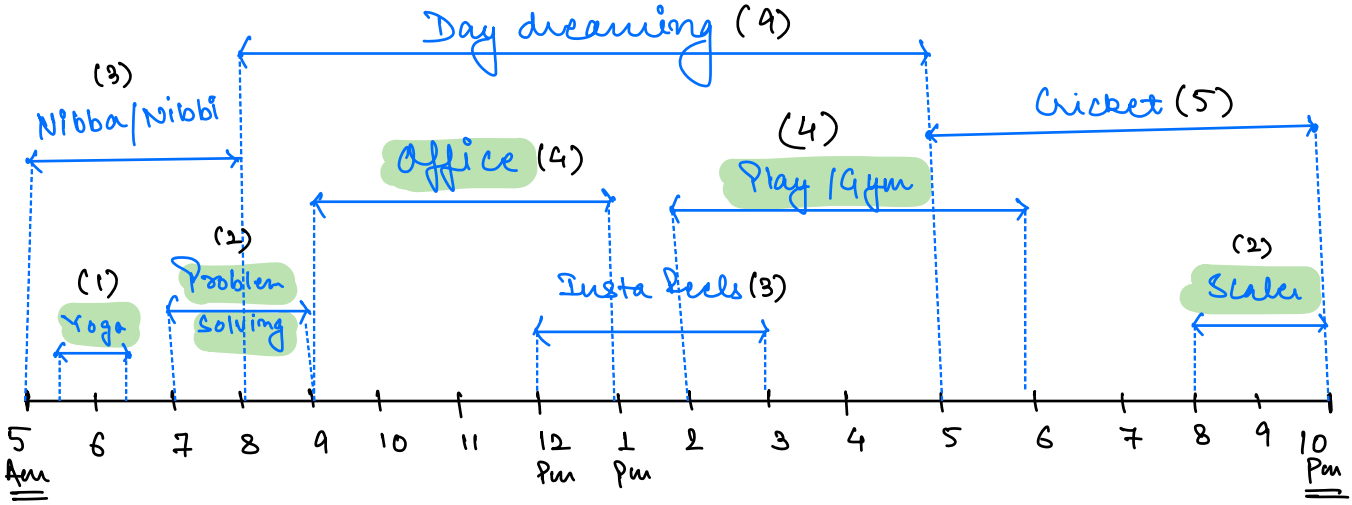
2) Based on what parameters we want to apply Greedy.

3) Check for any counter examples where Greedy won't work.

Greedy Algorithms :-

1) Prims / Kruskals Algo.
2) Dijkstra's Algo.
3) Huffman coding.

Q. Activities Selection



1) Once we start a task, we need to complete it.
2) At any given time, we can only perform a single task.
3) No. of max tasks we can do in a day.

⇒ Tasks

1) Yoga
2) Problem Solving
3) Office
4) Playing
5) Scaler.

⇒ 5

Greedy :-

1) Pick the tasks with min duration

→ Yoga : 1 hr
→ Problem Solving : 2hr
→ Scaler : 2hrs
→ Insta reels : 3hrs.

} 4 tasks ✗

2) Pick the tasks with min start time.
(Starting early)

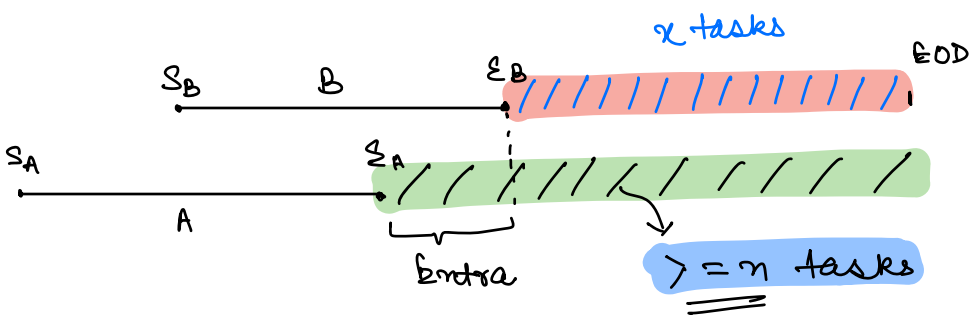
→ Nibba / Nibbi
→ Day dreaming
→ cricket

} 3 tasks ✗

3) Pick tasks with min end time
(Ending early)

1) Yoga
2) Problem Solving
3) Office
4) Playing
5) Scaler.

} 5



* By picking the tasks that ends first we are leaving more slots/time for upcoming tasks.

Q)

Job Scheduling

1) Given N tasks to complete.
2) Deadline assigned for each task, day on or before we can do the task.
3) Payment is assigned to each task.
4) On any given day we can perform only (1) task & each task takes 1 day.
5) Find max payment we can get.

Ex

Jobs	Deadline	Payment
a	(3)	100
b	1	19
c	2	27
d	1	25
e	3	30

finish on or before Day 3.

Days

d	c	a
1	2	3

$25 + 27 + 100 = 152$ ✗

c	e	a
1	2	3

$27 + 30 + 100 = 157$ ✓ ans.

a	c	e
1	2	3

157

Ex: Based on the deadline.

b	d	c	a	e
1	1	2	3	3
19	25	27	100	30

~~19~~ ~~25~~ 27
100 30

⇒ 157

Ex

Jobs	Deadline	Payment
a	3	5
b	1	1
c	3	6
d	2	3
e	3	9

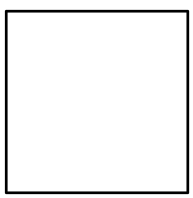
Sort Based on the deadline.

b	d	a	c	e ↓
1	2	3	3	3
1	3	5	6	9

~~4~~, ~~2~~, 5
6, 9

→ 5+6+9

20 rs.



→ insert()
→ size()
→ getMin()
→ deleteMin()

Min Heap.

Eg

b	d	c	f	a	e ↓
1	2	2	2	3	3
1	3	2	20	5	9

~~1~~, ~~3~~, ~~2~~,
20, 5, 9

ans = 34

HW Try to implement. ⇒ 10-15 mins.

- Sort based on the deadline.
- List<Pair<int, int>> data

———— * ————