Todays Content:

1. Franctional knapsack
2. Greedy Properties
3. Activity Selectim
4. Job Scheduling

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

Cash: 5548 : Min coins/notes requried to get requried cash?

Remaining Amount/Cash

5548	:	2000 * 2
1548	:	500 * 3
48	:	20 * 2
8	:	5 * 1
3	:	2 * 1
1	:	1 * 1

Total Coins: 10 min

Idea:

1. We are going from higher to lower denominatim
2. When we are at a denominatim
   : We remove as much as possible using that denom..

Greedy: Parameter: Demonination Order: dec

Greedy: Solving by selecting best optim available at eachstep.
so that we can get overall min or max = Greedy.

Currency: 1 10 18

Cash: 20 Min coins/notes requried to get requried cash?

Remaining Amount/Cash

20	18 * 1
2	1 * 2

Total Coins = 3 according to Greedy, not working.

Actualans = 10*2, 2 is finalans.

Super Market: We can eat 70kg, calculated max protien we can get.

Note : We can eat in kg denominatsm

Q: Max protien we can gain?

Vegetable		Protein Gained
Tomato	20kg	200p = 10p/kg
Apple	15kg	180p = 12p/kg
Onion	50kg	250p = 5p/kg
chicken	10kg	150p = 15p/kg
potato	25kg	200p = 8p/kg
Mango	12kg	132p = 11p/kg
Seafood	5kg	100p = 20p/kg

Gredy: Max Protien

Vegetable	Protein
Onion 50kg	250p
Tomato 20kg	200p
Protien =	450p

Gredy: Max Protien/kg ratio:

Vegetable	Protein
Seafood 5kg	100p
chicken 10kg	150p
apple 15kg	180p
mango 12kg	132p
Tomato 20kg	200p
Potato 8kg	64p
Protien =	826p

When to apply greedy?

a. For optimisation min/max related Problems : Greedy can be a technique

b. Based on what parameters, we need to apply greedy

c. Parameter we choose should tells, what's best at each step

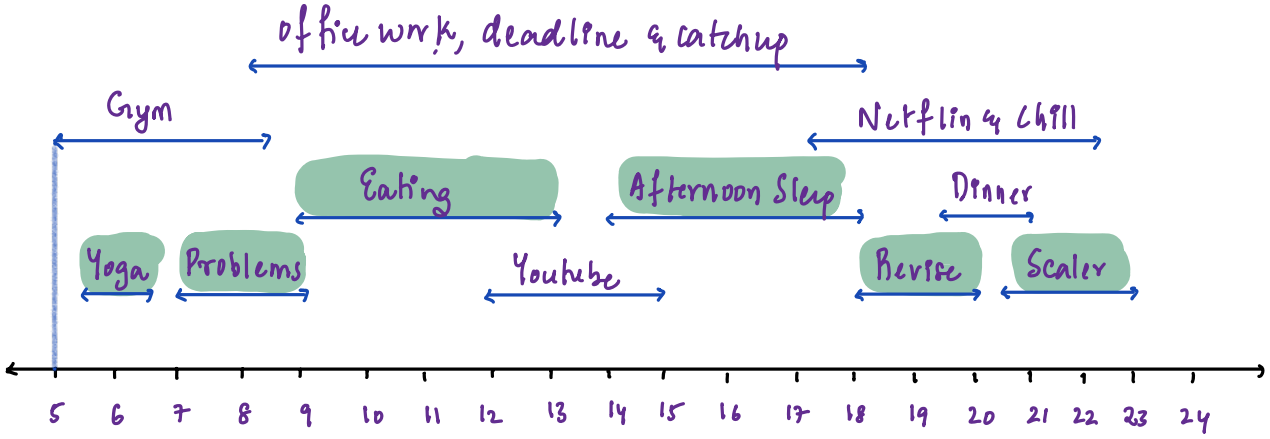
Correctness: By coming with counter examples

Real world applications?

a. Prims / kruskals algorithms

b. Dijktra's algo ⟶ Shortest Paths

Activity Selection:



Q: Max tasks which can be done

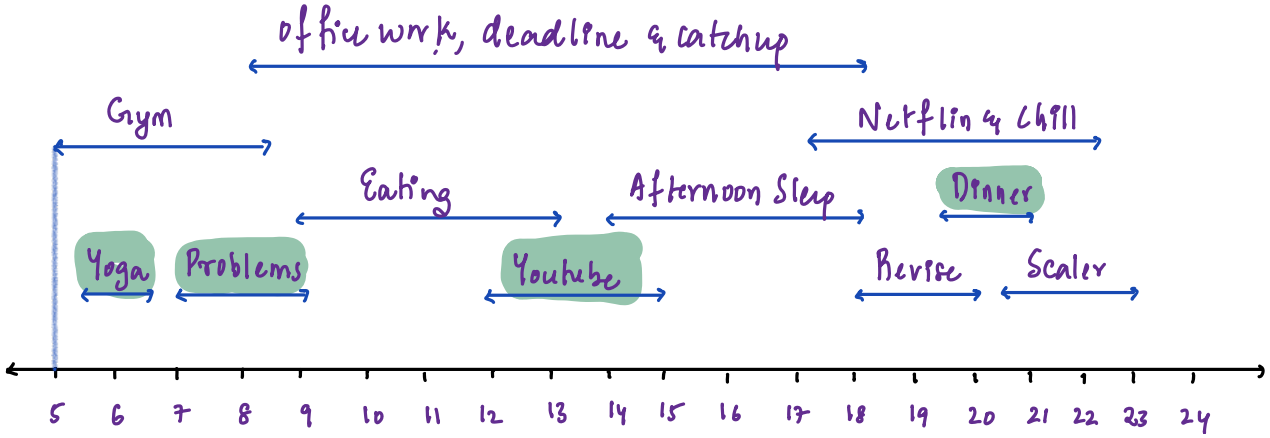
Note1: Once we start we need to finish the task

Note2: At any point we cannot do overlapping tasks

ans: 6 tasks

Idea: Using Greedy: Taking best at each step

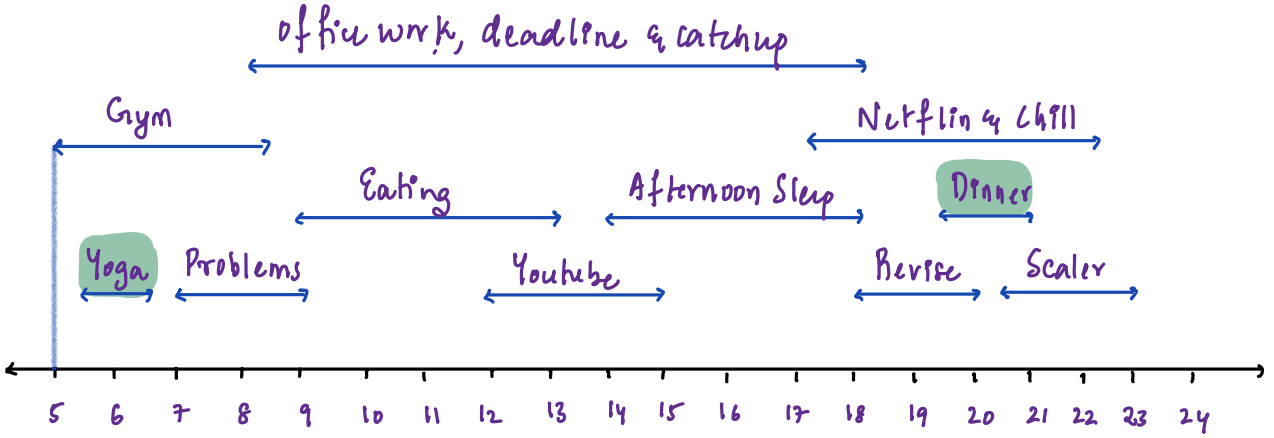
Parameter: Min time = 4 tasks *



Idea: Using Greedy: Taking best at each step

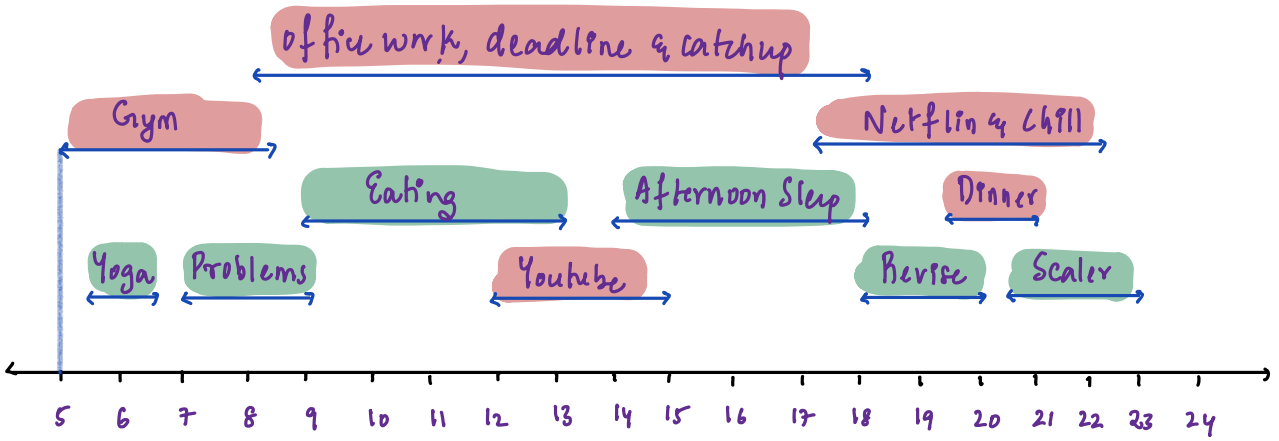
Parameter: {Task Size, start time}

1st prize 2nd prize

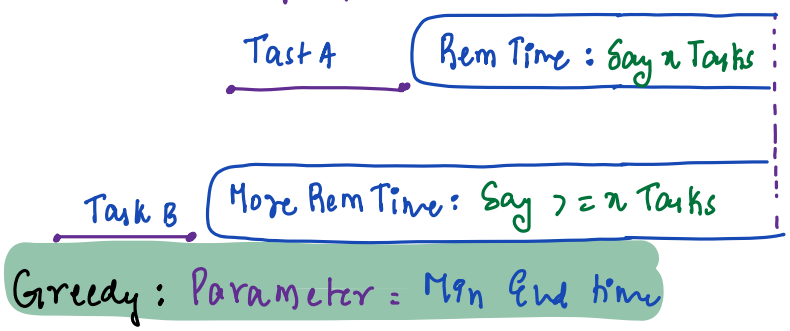


Idea: Using Greedy: Taking best at each step

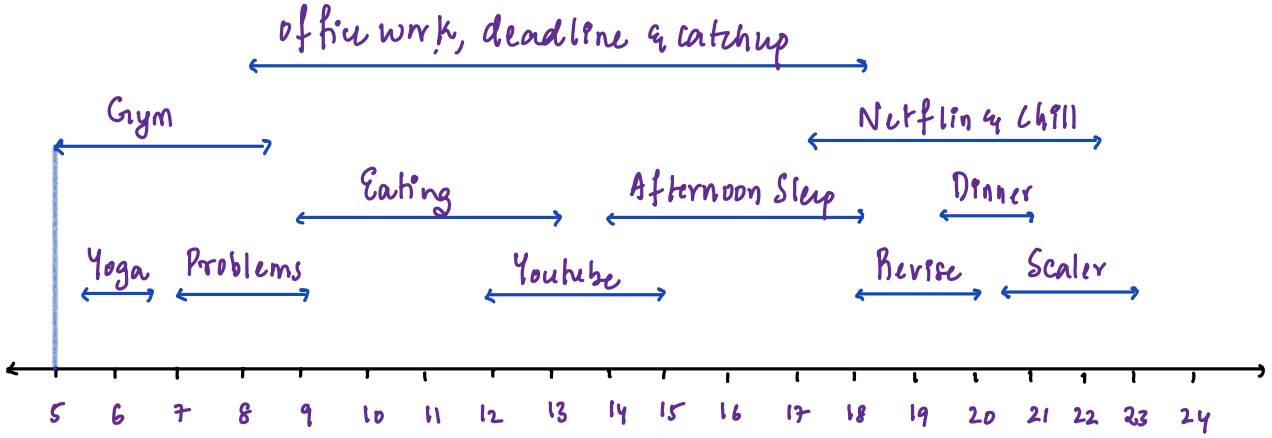
Parameter: Min end time ⇒ Task ending first



Correctness of logic: If we do tasks, which ends first, it will give us more time to spare, hence we can do more tasks.



Greedy: Parameter = Min End time



Task	s	e
Gym	5	8
Yoga	5.5	6.5
Problems	7	9
office	8	18
Eating	9	13
Youtube	12	15
AfterSleep	14	18
Netflix	12	22.5
Revise	18	20
Dinner	19.5	21
Scaler	20.5	23

Sort based on min end time →

Task	s	e		rs	re	
Yoga	5.5	6.5	✓	5.5	6.5	
Gym	5	8	✗	5.5	6.5	
Problems	7	9	s>=re	7	9	update
Eating	9	13	s>=re	9	13	update
Youtube	12	15	✗	9	13	
office	8	18	✗	9	13	
AfterSleep	14	18	s>=re	14	18	update
Revise	18	20	s>=re	18	20	update
Dinner	19.5	21	✗	18	20	
Netflix	12	22.5	✗	18	20	
Scaler	20.5	23	s>=re	20.5	2	update

TC: $O(N \log N + N)$

Code: TODO

Job Scheduling:

Given N tasks & payement & deadline, attached to each task

Find max payement we can get?

Note: We can finish a task on or before deadline day

Note: Each task takes 1 day

Note: Every day we can only do 1 task.

Eg1:

Job	Deadline Day	Payement
a	3	100
b	1	19
c	2	27
d	1	25
e	3	30

Day: 1 2 3

a e c = * c's deadline is 2

b c a = 146

d c a = 152

e c a = 157 } // max amount = 157

c e a = 157 }

Idea: Using Greedy: Taking best at each step

Parameter: Payements in dec order.

As per greedy: Day 1 2 3

a c = 180 *

Idea2: Using Greedy: Taking best at each step

Parameter: Sort them based on deadlines inc

obs1: If we have x days we can only do x tasks

obs2: On a day i → allcated task to amount
↳ allcate task so amount

Job	Deadline Day	Payement
a	3	100
b	1	19
c	2	27
d	1	25
e	3	30

Sort on deadlines inc

Task	b	d	c	a	e
Deadline	1	1	2	3	3
amount	19	25	27	100	80
	✓	19	✓	✓	25 ✓

Tasks Done

~~19~~ 25

27 100

→ operations

1. getMin()
2. insert()
3. deleteMin()
4. Min Priority Queue

Ex2:

Task	Deadline	Reward
a	3	5
b	1	1
c	3	6
d	2	4
e	3	9

Sort on deadlines inc

Task	b	d	a	c	e
Deadline	1	2	3	3	3
amount	1	4	5	6	9
	✓	✓	✓	✗	✗

Tasks Done

✗ ✗ 5

6 9

= Final ans = 20

Tasks:

Task	Deadline	Money
a	2	200
b	1	250
c	1	200
d	1	350
e	4	300
f	5	100
g	4	250
h	5	600
i	5	400
j	2	150

Sort on deadlines inc

Task	b	c	d	a	j	e	g	f	h	i
Deadline	1	1	1	2	2	4	4	5	5	5
amount	250	200	350	200	150	300	250	100	600	400
EmptySlot:	✓	x	x	✓	x	✓	✓	✓	x	x
Replace:		x	~~250~~		x				~~100~~	~~200~~

Tasks Done

~~250~~ 350 ~~200~~ 300 250

~~100~~ 600 400

Final ans = 1900

Code:

1. Sort based on deadline in inc TC: O(NlogN + NlogN)
2. PriorityQueue<Integer> mpq; SC: O(N)
3. Iterate on all tasks:

```
// Say for ith task: deadline = d & Amount = A
if(d > mpq.size()) { mpq.add(A) // EmptySlot }
else if(mpq.peek() < A) { // If peek < A:
    mpq.remove();
    mpq.add(A);
}
```

4. Delete & add all elements from mpq;