Greedy 1
Monday, 23 June 2021 7:17 PM

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

Cash : 5548 } In min number coins/notes

Thus may +10

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

Why? [Every coin ≥ 2* previous coin]

Test: –

Eg: coins 1 10 18 , Money = 20

By greedy : 18 + 1 + 1 : 3

Actual Best : 10 + 10 : 2

2nd)

SuperMarket : 70 kg

Vegetables :		Grained Protein	If Grained Protein	Protein/kg		
Tomato	20 kg	200 p	→ 20 kg (200p)	10p/kg	→ 20 kg	= 200
Apples	15 kg	180 p		12p/kg	→ 15 kg	180
onion	50 kg	250 p	→ 50 kg (250p)	5p/kg		
chicken	10 kg	150 p	450 p	15p/kg	→ 10 kg	150
potato	25 kg	200 p	Greedy based on total grained protein. *	8p/kg	→ 8 kg	64
Mango	12 kg	132 p		11p/kg	→ 12 kg	132
Sea food	5 kg	100 p		20p/kg	→ 5 kg	100
						826

If you want you can pick a kg from each item

[Greedy, based on protein/kg]

3) Greedy can mostly be applied on optimization problems
↳ [min max

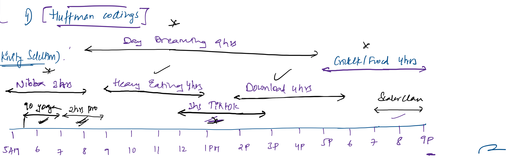
2) Decide on what parameter you want to apply greedy

3) Best way to disprove greedy by example

→ Greedy Algo :

1) Prim's Algorithm
2) Kruskal Algorithm
3) Dijkstra
4) [Huffman coding]

1st) (Activity Selection).



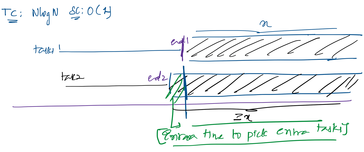
1) Start a task finish it
2) At any given point we can only do single task
3) Find max no of task we can do in entire day : 5

5:30 – 7
7 pm – 9pm
9 – 1pm
2 – 6pm
7 – 9pm

Greedy :
1) Start time : Nirban + Day Cricket : 3 *
2) Interval : Yoga ✓, gym ✓, Scale ✓, Trekk ✓ } 4 *
3) Endtime :
Increasing : yoga ✓, 2hr prog ✓, heavy eating ✓, Download, Scale } 5 ✓

Find 61 :
1) Sort tasks in increasing order of their end time
Based on above, who we will pick

TC : NlogN SC : O(1)



[Extra time to pick extra task]

2nd) Job Scheduling

1) There are N tasks to complete, & each task is allocated a deadline day [Before an certain day we can complete that activity]

For solving each task, you are also paid, Note: on a day you can do 1 certain task.

→ Q: Find max payment which we can get?

N = 10, D = 5
↳ no of days we have

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

1850
1800
1500
1650
1900
1650
1950
1400
1650

Greedy :
1) For every deadline task max

1	2	3	4	5
350	200	250	300	600

Money earned 1900 *

2) Pick Greedily based on money in decreasing order

↳ Sort in decreasing
< Money, Deadline >

1	2	3	4	5
task 4	task 7	task 5	task 9	task 8
350	250	300	400	600

1900

Note:) If 2 tasks same, we pick task with more deadline

Catch: (we pick a task in greedy order whose deadline 5)

[Check 5 check 4 check 3
9 * * *]

Given a deadline d, [write a function which can give max available day x = d]

Pseudocode :

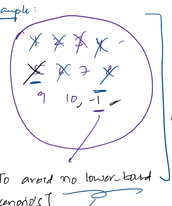
1) Sort given data based on money & deadline → [NlogN]
2) Set < int > s, [Insert all 1 to D with -1] [No of days we have] [DlogD]

int sum=0;
for(int i = 0; i < N; i++){
int start = task[i].first
int d = task[i].second
int x = lower_bound(s, d)
if(x == -1) { continue }
else { sum += arr[i]; s.erase(d) }
}
return sum;

NlogD

TC : [NlogN + DlogD + NlogD] SC : O(D)

For example :



[To avoid no lower_bound scenarios]

[(logD) time]

deadline = 5 → 5 is there set Allocate 5 here

deadline 8 → 8 is there set, Allocate 8

deadline 4 → 4 is there set, Allocate 4

deadline = 5 → [is 5 present, No *
lower_bound of 5 : 3 ✓

deadline = 6 → [6 is there set, Allocate 6]

deadline = 2 → [2 is there, Allocate 2]

deadline = 1 → [1 is there, Allocate 1]

deadline 4 → [4 is not there, -1] * skip