* Introduction to Greedy Algorithms [Maximum or Minimum Values]

All probable Greedy Algo Questions for coding interviews: →

1. Minimum number of coins *
2. Minimum number of platforms *
3. Activity Selection Problem *
4. Job Scheduling Problem
5. Chocolate Distribution Problem * (Minimum Absolute Difference)
6. Fractional Knapsack
7. 0/1 Knapsack
8. Huffman Encoding
9. Gas Station Problem *
10. Lemonade Change Problem *
11. Minimum Cost of Connecting Ropes
12. Minimum Arrows to burst all balloons.
13. Nimum & Donuts
14. Minimum Number of Steps
15. Sliding Window Problems

Just Code
Geeks For Geeks
Coding Ninjas

(Minimum no of Coins)

(91 x 1 = 91 coins) max

sort ← coins = {1, 2, 5, 10, 20, 50, 100, 200, 500, 1000, 2000}

Value, V = 91 → (50, 20, 20, 1) 4 coins

res = {50, 20, 20, 1}

```
for (i = size-1; i >= 0; i--)
    while (V >= coins[i])
        V -= coins[i];
        res.add(coins[i]);
min coins = res.size();
```

91 > 50
91 − 50 = 41
41 − 20 = 21
21 − 20 = 1
1 − 1 = 0

** Activity Selection Problem: →

① sort (finish) ⑪ 1st Activity
⑪ s(n) >= f(p)

Given a list of activities with their start & finish times, select the (maximum) number of activities that can be completed by a person in a given time if he/she can only perform one activity at a certain time without overlapping. (Custom Comparator) **

(Max Activities)

Activity	Start	Finish
A1	5	7
A2	8	9
A3	1	4
A4	5	9
A5	0	6
A6	3	5

Sort (Finish):
A3 (1, 4)
A6 (3, 5)
A5 (0, 6)
A1 (5, 7)
A4 (5, 9)
A2 (8, 9)

Output: A3 (1,4) 0 is; A1 (5,7) is; A2 (8,9)

(1,4) (3,7) (8,9)
s f s f s f

Minimum Number of Platforms:

arr[] = {900, 940, 950, 1100, 1500, 1800}

dep[] = {910, 1200, 1120, 1130, 1900, 2000}

sort dep[] = {910, 1120, 1130, 1200, 1900, 2000} sort

platform = 0 → +1, −1, +1, +1, +1

max platform = 0 → +1, +1, +2, +3

860: Leetcode (Lemonade Change)

① 5, 5, 5, 10, 20 — true

bill	$5	$10
5	1	0
5	2	0
5	3	0
10	2	1
20	1	0

10 + 5 = 15

② 5, 5, 10, 10, 20 — false

bill	$5	$10
5	1	0
5	2	0
10	1	1
10	0	2
20		

* Gas Station Problem: → (index = 3)

5 √ TCS basic
6 → √ TCS Digital
8-10 TCS Ninja

NOT

4 − 1 = 3
3 + 5 − 2 = 6
6 + 1 − 3 = 4
4 + 2 − 4 = 2
2 + 3 − 5 = 0
0 + 4 − 1 = 3

final gas[i] | Cost[i] | diff[i] = gas[i] − cost[i] | curr tank | total tank | index

station	final gas[i]	Cost[i]	diff[i]	curr tank	total tank	index
*0	1	3	−2	−2	−2	1
1	2	4	−2	−2	−4	2
2	3	5	−2	−2	−6	3
3	4	1	3	3	−3	× 3
4	5	2	3	3	0	× 3

Cost to go to next

if curr < 0 → it'll start at (i+1)

if the total tank value is 0 or greater, we found (i+1)

(Dynamic Programming)

"Those who forget the past are forced to repeat it!"

* It is the solution of smaller overlapping subproblems to solve an even bigger problem.

4 steps:
Question
① Recursion
② Memoization → Top-Down Approach
③ Tabulation → Bottom-Up Approach
④ Space Optimization

f(0) f(1)
0, 1, 1, 2, 3, 5, 8, 13, 21, ...
0 1 2 3 4 5 6 7 8 → f(n) = f(n−1) + f(n−2)

① Recursion → No choice (All calls)
② Memoization → Previous values are not calculated.
③ f(0), f(1) — Store array — Tabulation

Recursion Tree