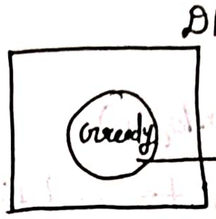
Greedy Problems

Exa : sum = 15 ↳ Min no. of coins needed to make this sum
↓
Coins = 1 or 2

∴ We can take 7 coins of '2' and 1 coin of '1'.

$2 \times 7 + 1 = 15$

Exa : 2/3 → Sum = 16
↳ $3 \times 4 + 2 \times 2 = 16$ (Min no. of coins)



All dp problems can be solved through Greedy.

Greedy Use → sorting (based on starting & ending pt)

Greedy string or Array Problems

Scheduling Problems

↳ Graphs (Prim's Algo, Kruskal Algo, Dijkstra Algo)

① Job scheduling :-

Start Time	End Time
1	5
1	3
1	2
4	7
2	4
3	6

A single worker attend jobs. What is max no. of non overlapping jobs he can attend ?

Start time > End Time
↓
This is the condⁿ needed to be applied

Greedy Identification

1) Higher constraint : $10^5/10^6$
↓
Use dp with greedy

2) Best step
↳ Can I make a decision without revisiting later

Approach :-
1) Sort by End time.

1	2	✓
1	3	✗
2	4	✓
1	5	✗
3	6	✗
4	7	✓

Count = ~~1~~ ~~2~~ 3

End = ~~2~~ ~~4~~ 7

∴ $\boxed{\text{Ans} = 3}$

2. Non - overlapping Intervals :-

Start	End
1	5
1	9
1	2
2	4
2	5
2	3
3	9
3	6
4	7

1	2	
2	3	
2	4	✓
1	5	✓
2	5	✓
3	6	
~~3~~	~~9~~	
4	7	✓
1	9	
3	9	

count = ~~0~~ ~~1~~ ~~2~~ ~~3~~ ~~4~~ 5

end = ~~3~~ 6

So we need to remove 5 intervals.

so non - overlapping intervals are : {1,2}, {2,3}, {3,6}

Start > End → If not, remove.

Overall T.C : O (n log n)

1) sort according to end time → O (n log n)

2) Check which interval are overlapping, with cond^n,

if (start < end) → inc. count

else update end with the current end.

```
* sort [intervals.begin (), intervals.end (), [] (const vector <int> &a, const
        vector <int> & b) {

      If a[1] < b[1]
        then vector a would be on left side & b would be on rightside when
        sorting

      else
        vice - versa.
  }
```

3. Fractional Knapsack

Ex:1

Wt :	10	20	30
val :	60	100	120

capacity = 50.

→ We will take value whose $\frac{val}{wt}$ will be greater.

Ratio : 6 5 4

Result

$\frac{val}{wt}$ = +60 + 100 + 80

= $\boxed{240}$ ↳ ans

50 - 10 = 40 - 20

↓

20 - 20

↓

0

We can take only 20 out of 30

so 1 → 4

20 → 20 × 4 = 80

Ex 2: wt : 5 2 3 4 $\boxed{\text{cap} = 12}$

val : 10 7 3 20

$\frac{\text{val}}{\text{wt}}$: $\overset{0}{2}$ $\overset{1}{3.5}$ $\overset{2}{1}$ $\overset{3}{5}$

Now sort

⇒ $\overset{③}{5}$ $\overset{①}{3.5}$ $\overset{⓪}{2}$ $\overset{②}{1}$

12 → +20

↓ −4

8−2 → +7

↓

6−5 → +10

↓

1−1 → +1

↓

0

38 → Ans

T.C : Sorting → O(nlogn) + Traversing → O(n) > O(nlogn)

S.C : O(n)

4. Nested Ranges Check (CSES Problem)

1. Keep each range with its original index, so we can ans in correct order later.
2. Sort the ranges by :
   → starting pt (x) inc.
   → If same x, then by ending pt (y) dec, bcoz bigger ranges come before smaller ones.
3. Go from left to right. Keep the max. y seen so far. If current y ≤ max_y → this range is inside an earlier one.
4. "Contains" check
   Go from right to left. Keep the min. y seen. If current y ≥ min y → this range contains a later one.

Ex :-

(1,6) → can contain others

(2,4) → is inside (1,6)

(4,8) → no one contains it & it contains no one

(3,6) → is also inside (1,6) ..