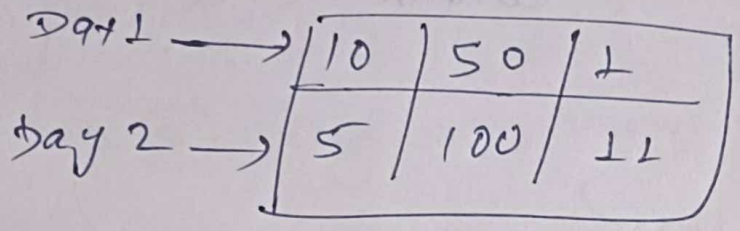
Que.: Ninja's Training

⟹ Why Greedy approach fails?

ex. N=2



By Greedy approach at Day 1 we choose 50

at Day 2 we choose ~~11~~

we can't choose 100 at Day 2 bcz of the constraint

ans. — 50 + 11 ⟹ 61

but actual ans = 10 + 100 ⟹ 110

⟹ Now Greedy approach fails so whenever Greedy fails the first thing we have to do is

Try all possible ways.

↳ This leads to recursion.

In Recursion, if we want to write recurrance we should follow three steps.

Step → 1: Express every problem in term of index

Step → 2: Do stuffs on that index

Step → 3: Take max of all that stuffs bcz acc to que. we have to find maximum merit points.

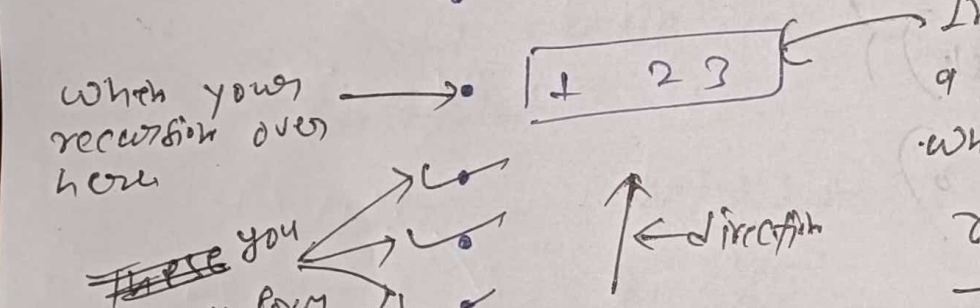
In the question we have given day (as variable) but we can treat it as index.

```
f(day, last_task) {
    if(day == 0)
    {
        maxi = 0
        for(i = 0 → 2) {
            if(i != last_task)
            {
                maxi = max(maxi, task[0][i]);
            }
        }
        return maxi
    }

    maxi = 0
    for(i = 0 to 2)
    {
```

```
if( i != last_task){
    points = task[day][i] + f(day-1, i)
    maxi = max(maxi, points);
}
return maxi;
```

⇒ ~~How~~ How to do stuffs on index / on that day



⇒ So acc. to condition along with index (day ~~is~~ chosen as index) ~~we~~ you can add a parameter in recurrance bcz when you perform all the

Stuffs you need to know what tasks you done in last day so we avoid constrains

So we choose the parameter last_task.

last_task = 0 → task 0 is performed on last day

last_task = 1 → —— 1 ——

last_task = 2 → —— 2 ——

last_task = 3 → ~~—— 3 ——~~ No! task was done

When was the condition of last_task = 3 arrise when you start from (n-1)th day 0 base indexing so we don't know what task we perform on nth day bcz we don't have data of nth day.

f(2,1) → This fn state that give me the max merit points that you can get if you perform task 0 to 2nd idx when you perform task 1 on 3rd day

idx → 0 1 2 3

task → |——————| 1

⇒ Check if their (in recursive tree) any overlaping subproblems presents or not.

Ex.

t0	t1	t2	← tasks
2	1	3	d0 → days
3	4	6	d1
10	1	6	d2
8	3	7	d3

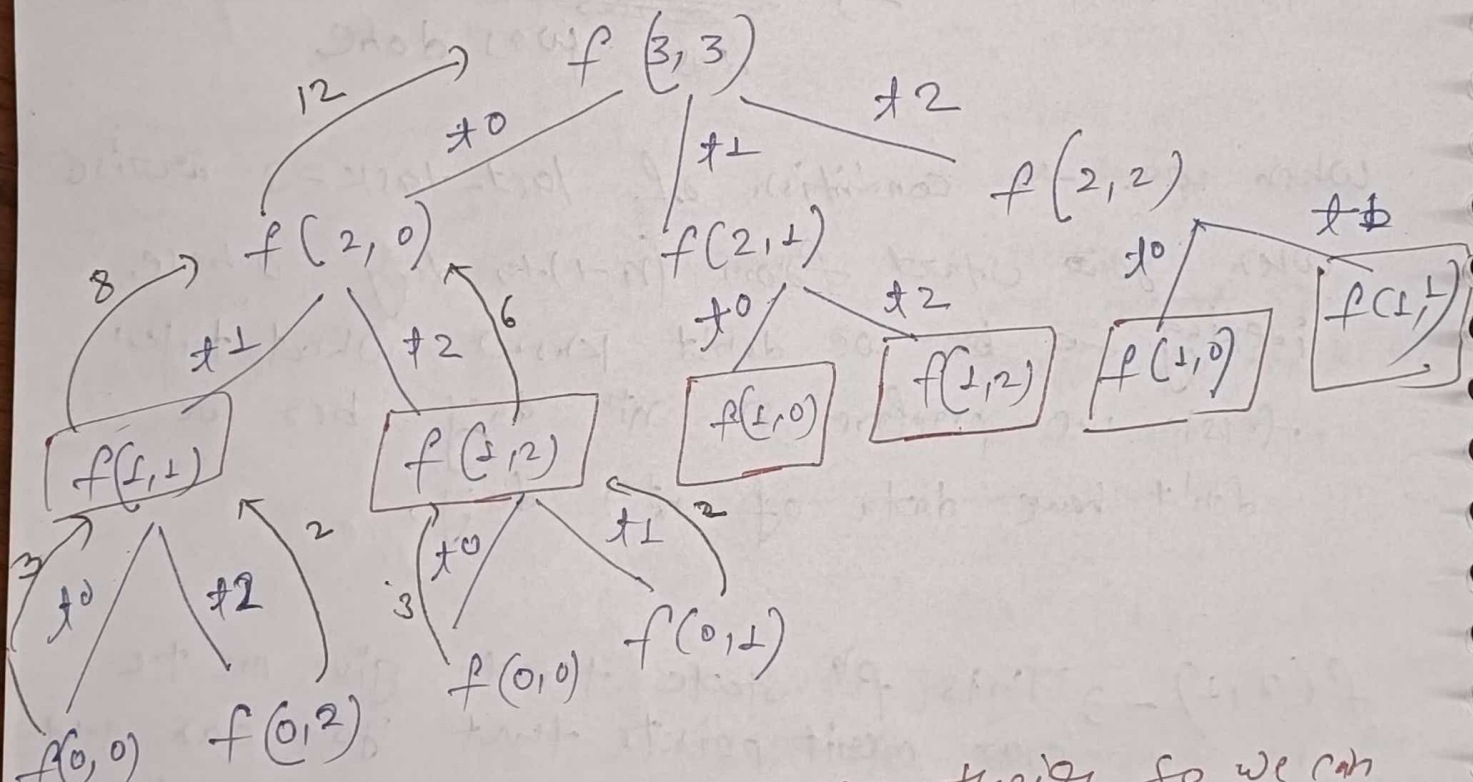
$f(\text{day}=3, 3)$

day=3 ↑ task=3 means no task perform

$f(2,1)$

↑ d2 (day-2) ↑ previously at day3 d3 we perform task 1 t1

Recursive Tree



overlaping subproblems are their so we can use DP

⇒ Now how to choose DP data structure?

ans:→ Here two parameters are changing day & last-task

the value of day can be 0, 1, 2 ---- N-1 so their are N different value that day can have

the value of last-task can be 0, 1, 2, 3 so their are 4 different values that last-task can have.

day → N

last-task (0,1,2,3) → 4

So for every N states their can be 4 diff possible so we use 2D array

2D array DP → N × 4