problem : knapsack of capacity c
n kinds of objects → each object has a weight
and profit : $O_i (w_i, p_i)$
there are unlimited copies of each O_i
maximise profit

Step 1 formulate into subproblem

∵ 'n' is not capped, we can't use knapsack(n)
we must use both n and c → knapsack(n, c)
if we don't choose n → knapsack(n−1, c)
if we do choose n → knapsack(n, c−w_n)
until n = 0 or c = 0

n^{th} object

① chosen : $C_{available}$ = c − w_n
profit = p_n + profit of rest
↓
subproblem(n, c−w_n)

② not chosen : $C_{avaible}$ = C
profit = profit of rest
↳ subproblem(n−1, c)

∵ current profit depends
on profit of subproblems,
for current profit to be maximum,
the solution of the subproblems should also be optimal.

principle of optimality holds.

Step 2 recursive function definition

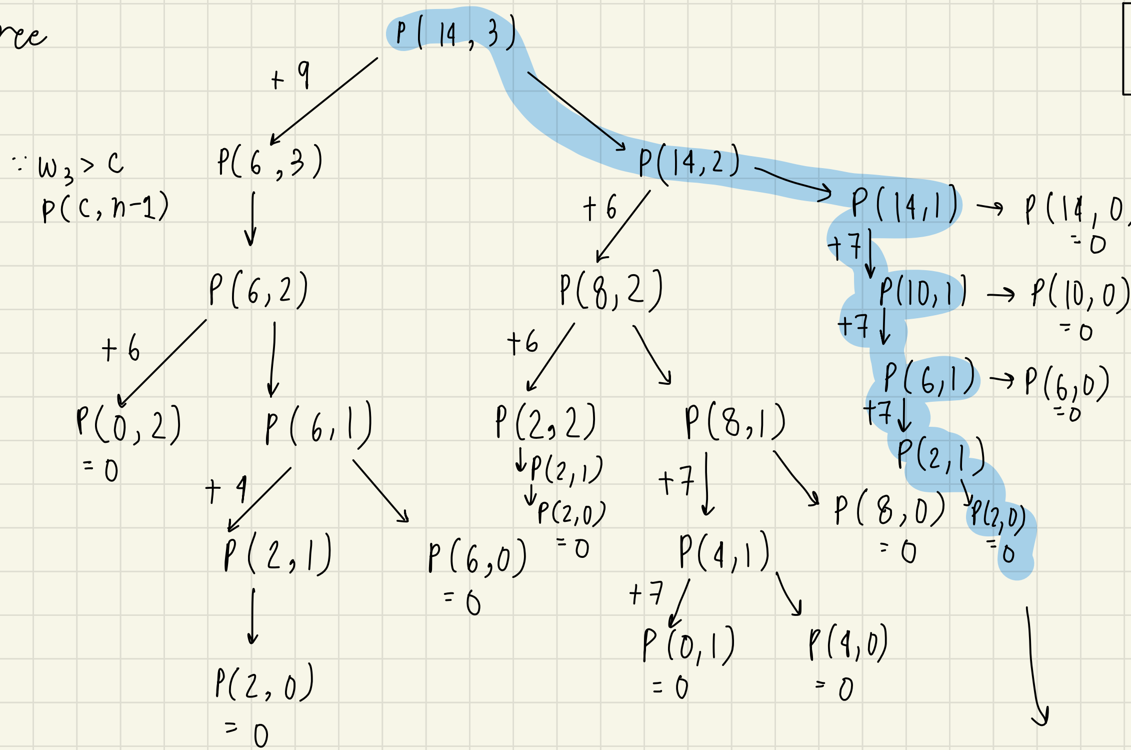
$$\text{profit}(c, n) = \max\left(\begin{array}{l} P_n + \text{profit}(c - w(n), n), \\ \text{profit}(c, n-1) \end{array} \right)$$

$$\text{profit}(0, n) = \text{profit}(c, 0) = 0$$

Step 3 subproblem graph $\longrightarrow$

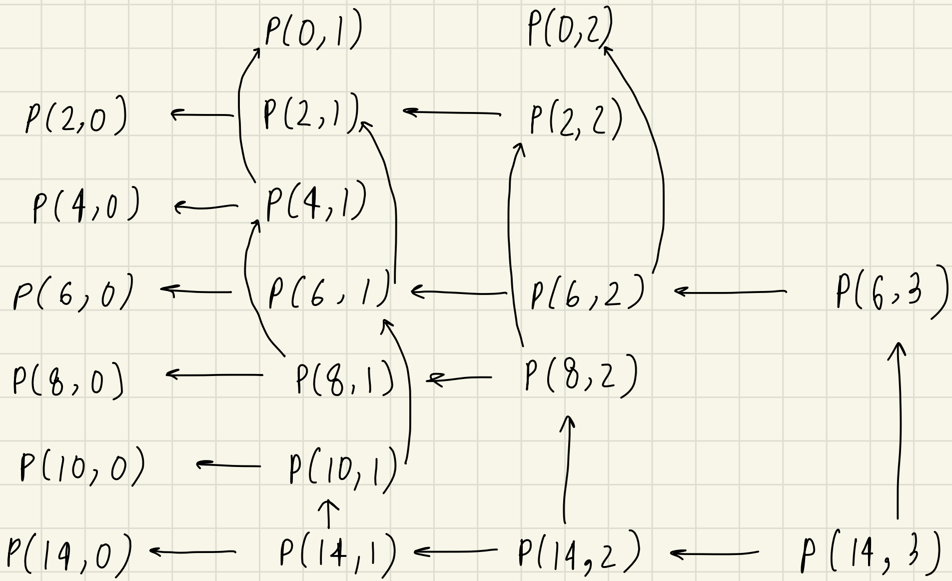
Step 3
subproblem tree

	1	2	3
w_i	4	6	8
p_i	7	6	9



max = 21

Subproblem graph



P(a, b)

a : capacity

b : object