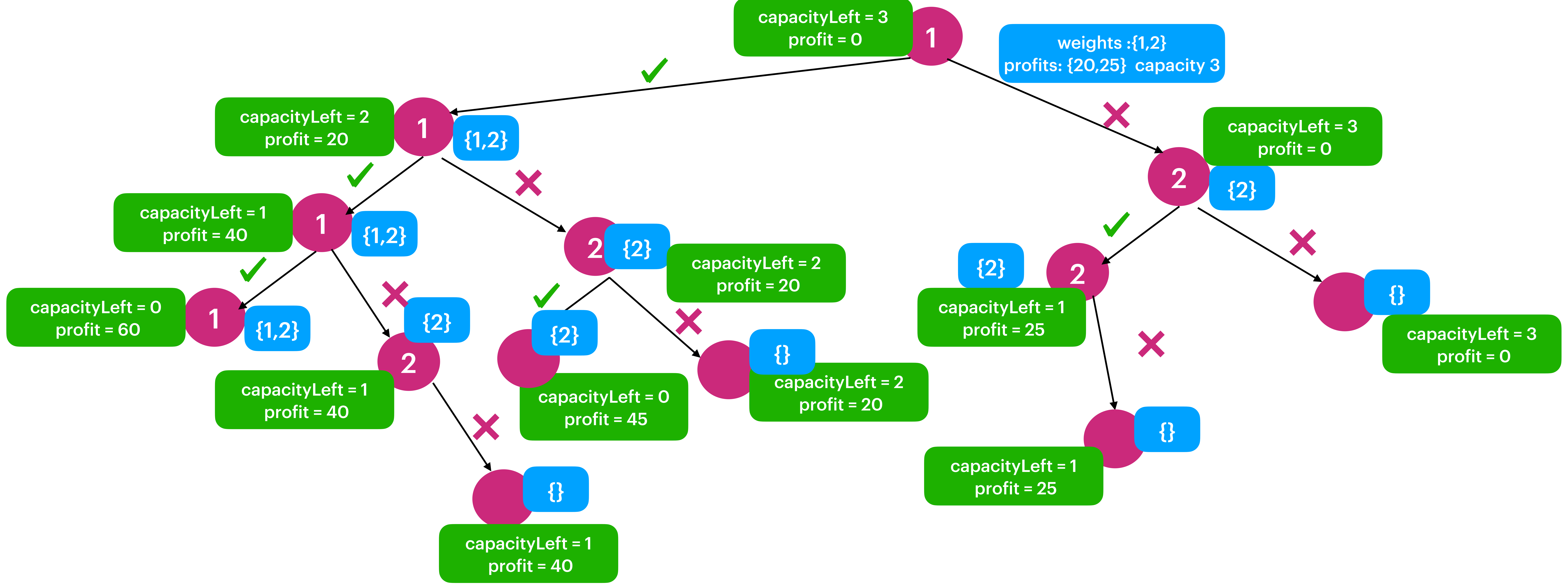
Problem Statement : Given item weights and their respective profits, itemWeight = {1,2,3} ItemsProfit = {5,2,4} , Get the max profit. We can have duplicate Items.

Constraints :

The knapsack cacapacity : 5

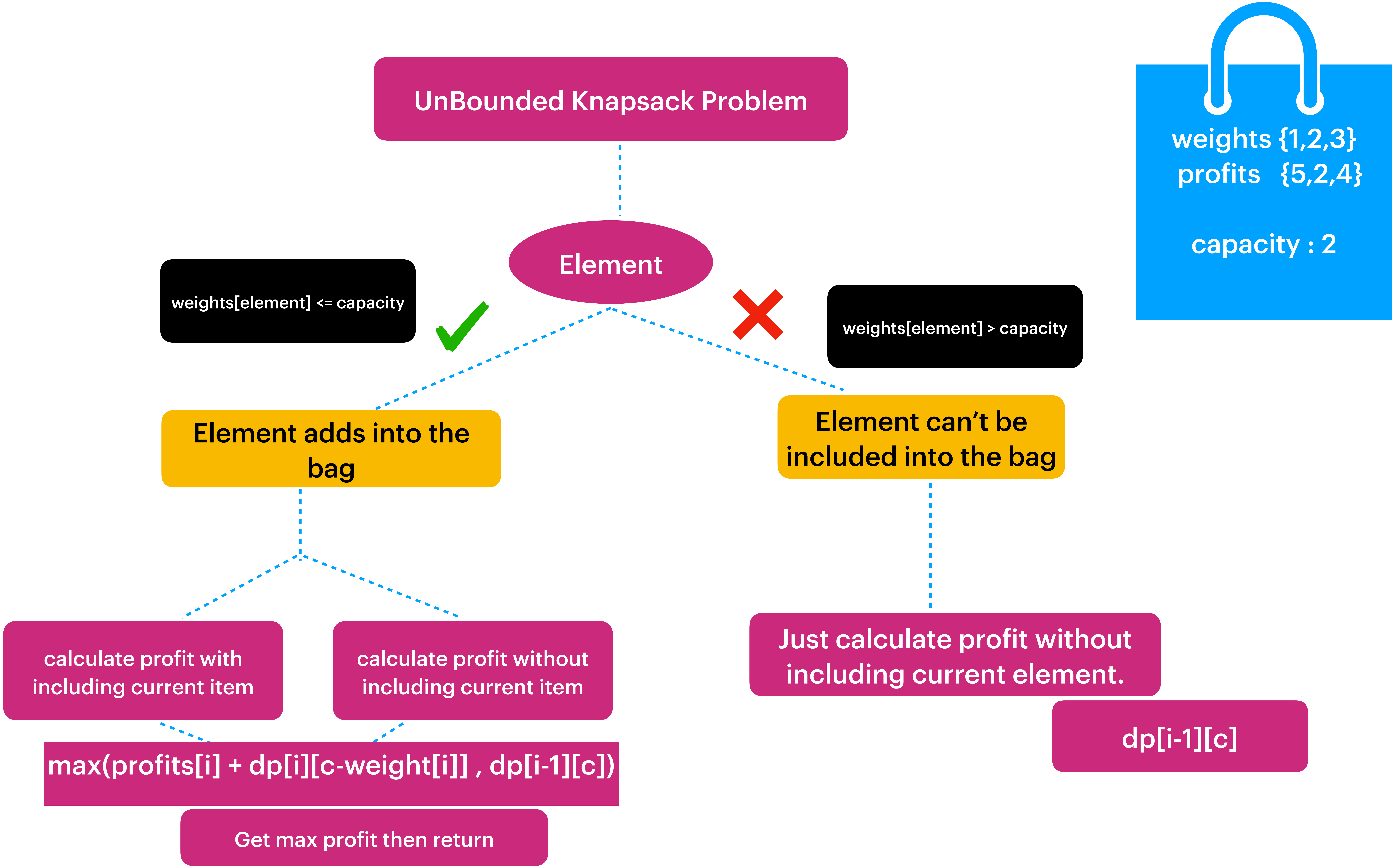
Expected Output :

Max Profit we can gain by choosing itemWeights {1,1,1,1,1} & respective profits {5,5,5,5,5} = 25



Time Complexity : 2^n+c
Space Complexity : O(n+c)

Only difference, we can move to next element only when we exclude item.


UnBounded Knapsack Problem
weights {1,2,3}
profits {5,2,4}
capacity : 2
Element
weights[element] <= capacity
weights[element] > capacity
Element adds into the bag
Element can't be included into the bag
calculate profit with including current item
calculate profit without including current item
Just calculate profit without including current element.
max(profits[i] + dp[i][c-weight[i]] , dp[i-1][c])
dp[i-1][c]
Get max profit then return

itemWeight = {1,2,3}
ItemsProfit = {5,2,4},
capacity : 5
For zero capacity all the items gives 0 profit.
Target Sum
0 1 2 3 4 5
Profits
5
2
4
Weight
1
2
3
Index
0
1
2
0 5 10 15 20 25
0 5 10 15 20 25
0 5 10 15 20 25
When we are solving for the 1st element.
i.e at index 0.
We don't have option of comparison so
If weight of the 1st element is <= capacity
then value will be,
profits[0] +
dp[0][c-weight[0]].
f weight of the 1st element is > capacity then value is 0.
Element weight
<= capacity
> capacity
Max(Including element , excluding element)
max(profits[i] + dp[i][c-weight[i]] , dp[i-1][c])
Exclude Element
dp[i-1][c]