We have Items, every Item has weight and profit.

Unbounded Knapsack Problems

Fixed Capacity

Rod Cutting :

Given a rod of length ‘n’, we are asked to cut the rod and sell the pieces in a way that will maximize the profit. We are also given the price of every piece of length ‘i’ where ‘1 <= i <= n’ .

Example:
Lengths: [1, 2, 3, 4, 5]
Prices:[2, 6, 7, 10, 13]
Rod Length: 5

Output : 14
{1,2,2} max profit can be possible.

Lets see all possible positive options :::

Lengths: [1, 2, 3, 4, 5]
Prices: [2, 6, 7, 10, 13]
Rod Length: 5

=> {1,1,1,1,1} => 5*2 = 10RS

=> {1,1,1,2} => 3*2 + 6 = 12 RS

=> {1,1,3} => 2 * 2 + 7 = 11

=> {1,4} => 2 + 10 = 12

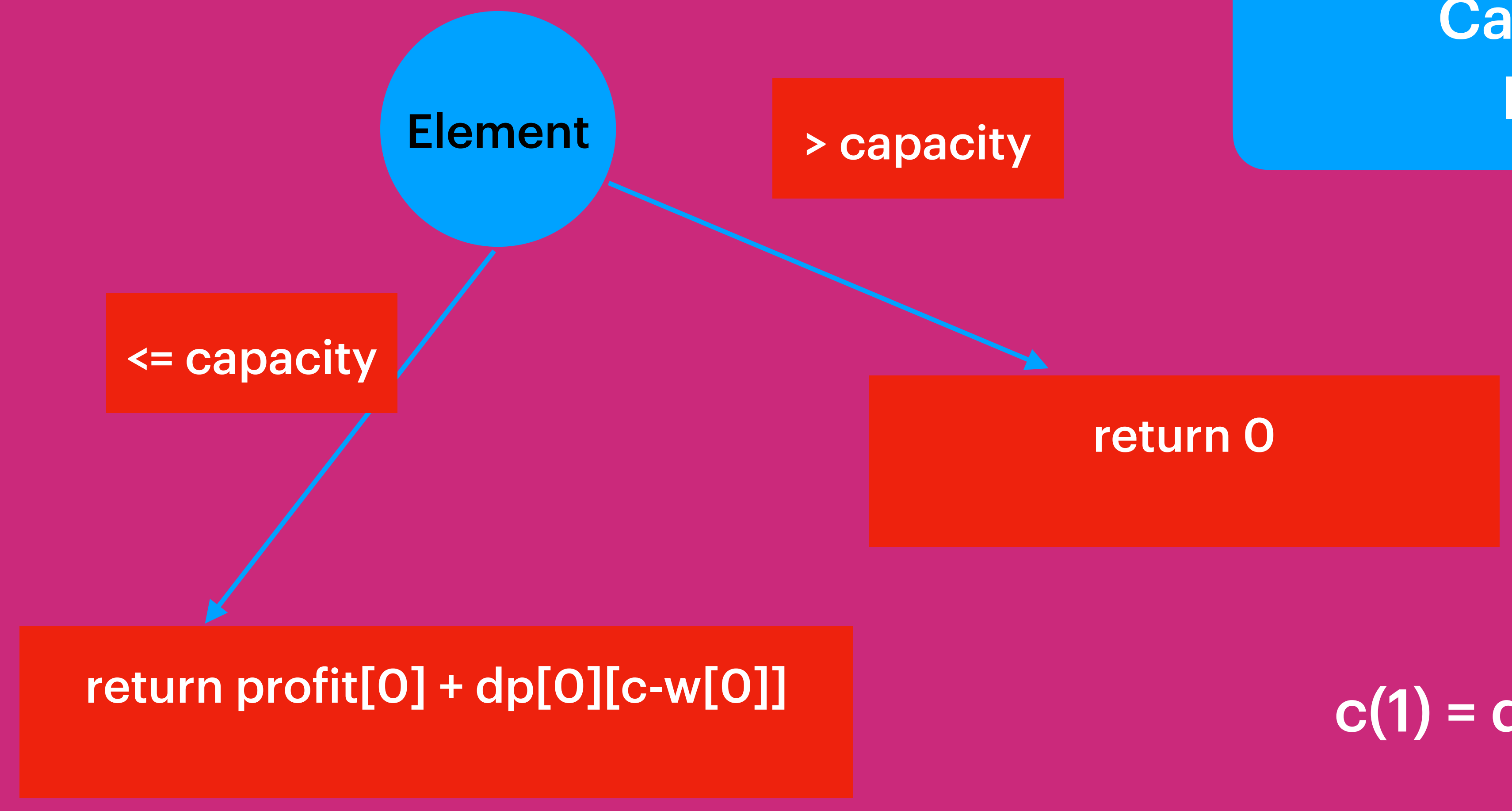
=> {1,2,2} => 1 * 2 + 2 * 6 = 14 RS

=> {2,3} => 1 * 6 + 1 * 7 = 13 RS

=> {5} => 1 * 13 = 13 RS

So Max Profit we got is {1,2,2} => 14

Lengths: [1, 2, 3, 4, 5]
Prices: [2, 6, 7, 10, 13]
Rod Length: 5



Calculating subproblem with element 1 :
Index = 0, w{1} p{2} , Rod Length: 5

$c(5) = dp[0][5] = \{1,1,1,1,1\} \Rightarrow 5 * 2 = 10$

Lets see how it works !!!!

$c(0) = dp[0][0] = w(0) > c(0) = 0$

$c(1) = dp[0][1] = p[0] + dp[0][1-1] = p[0] + dp[0][0]$
$= 2 + c(0) = 2 + 0 = 2$

$c(2) = dp[0][2] = p[0] + dp[0][2-1] = p[0] + dp[0][1]$
$= p[0] + c(1) = 2 + 2 = 4$

$c(3) = dp[0][3] = p[0] + dp[0][3-1] = p[0] + dp[0][2] = p[0] + c[2]$
$= 2 + 4 = 6$

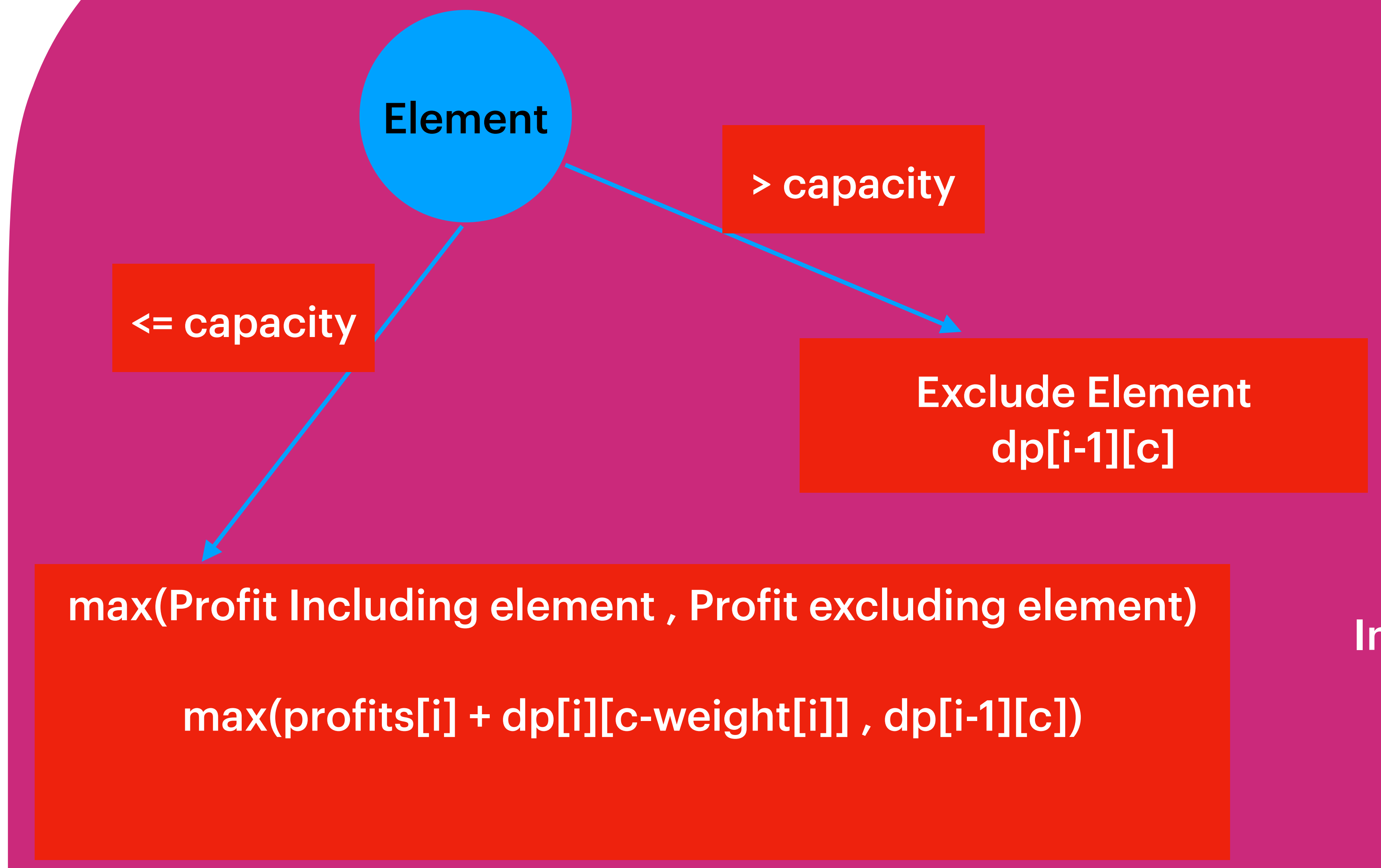
$c(4) = p[0] + c(3) = 2 + 6 = 8$

$c(5) = p[0] + c(4) = 2 + 8 = 10$

Calculating subproblem with element 1 :

When we consider only one element, we should see either we can include or exclude the element.
If weight of element > capacity
then profit = 0
Otherwise
profit = currentElementProfit + profitOfCapacityLeft
= profit[0] + dp[0][capacity - w[0]]

Calculating subproblem with elements 2
Index = 1,   w{1,2} p{2,6} , Rod Length i.e capacity : 5



c(0) = dp[1][0] = w(2) > c(0) = 0

c(1) = dp[1][1] = w(2) > c(1) = dp[0][1] = 2

c(2) = dp[1][2] = w(2) <= c(2) = 6
Exclude Element = dp[0][2] = 4
Include Element = p[1] + dp[1][2-2] = p[1] + dp[1][0] = 6 + 0 = 6
Max(IncludingElement, Excluding Element)
Max(6,4) = 6

c(3) = dp[1][3] = w(2) <= c(3) =
Exclude Element = dp[0][3] = 6
Include Element = p[1] + dp[1][3-2] = p[1] + dp[1][1] = 6 + 2 = 8
Max(IncludingElement, Excluding Element)
Max(8,6) = 8

c(4) = dp[1][4] = w(2) <= c(4) =
Exclude Element = dp[0][4] = 8
Include Element = p[1] + dp[1][4-2] = p[1] + dp[1][2] = 6 + 6 = 12
Max(IncludingElement, Excluding Element)
Max(12,8) = 12

c(5) = dp[1][5] = w(2) <= c(5) =
Exclude Element = dp[0][5] = 10
Include Element = p[1] + dp[1][5-2] = p[1] + dp[1][3] = 6 + 8 = 14
Max(IncludingElement, Excluding Element)
Max(14,10) = 14

Lengths: [1, 2, 3, 4, 5]
Prices: [2, 6, 7, 10, 13]
Rod Length: 5
For zero capacity all the items gives 0 profit.
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.
Target Sum
0 1 2 3 4 5
Prices
2 6 7 10 13
Length
1 2 3 4 5
Index
0 1 2 3 4
0 2 4 6 8 10
0 2 6 8 12 14
0 2 6 8 12 14
0 2 6 8 12 14
0 2 6 8 12 14
Element weight
<= capacity
> capacity
Max(Including element , excluding element)
max(profits[i] + dp[i][c-weight[i]] , dp[i-1][c])
Exclude Elem
dp[i-1][c]

Back Tracking print selected elements to gain MaxProfit.