We have Items, every Item has weight and profit.

Knapsack Pattern

Fixed Capacity

Example

Problem Statement : Every Knapsack has the capacity, we should choose the items , which can give higher profit, by considering capacity in mind.
example : itemsWeight = {1,2,3} itemsProfits = {5,2,4} capacity : 5

Dynamic Programming Solutions

Allow only unique items.

0/1 Knapsack [Bounded Knapsack]

Either we consider or don't consider

Allow duplicate items.

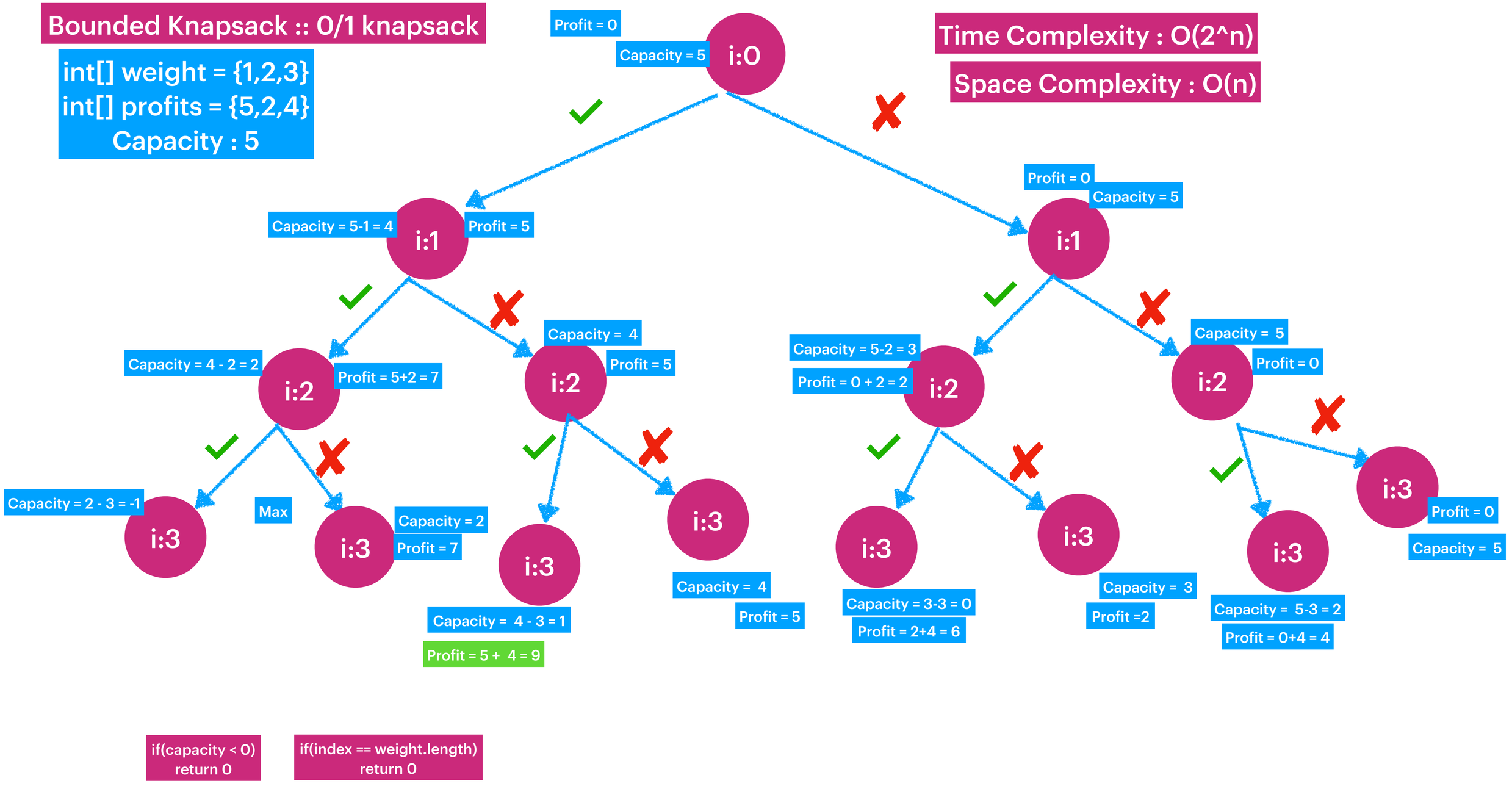
Unbounded Knapsack

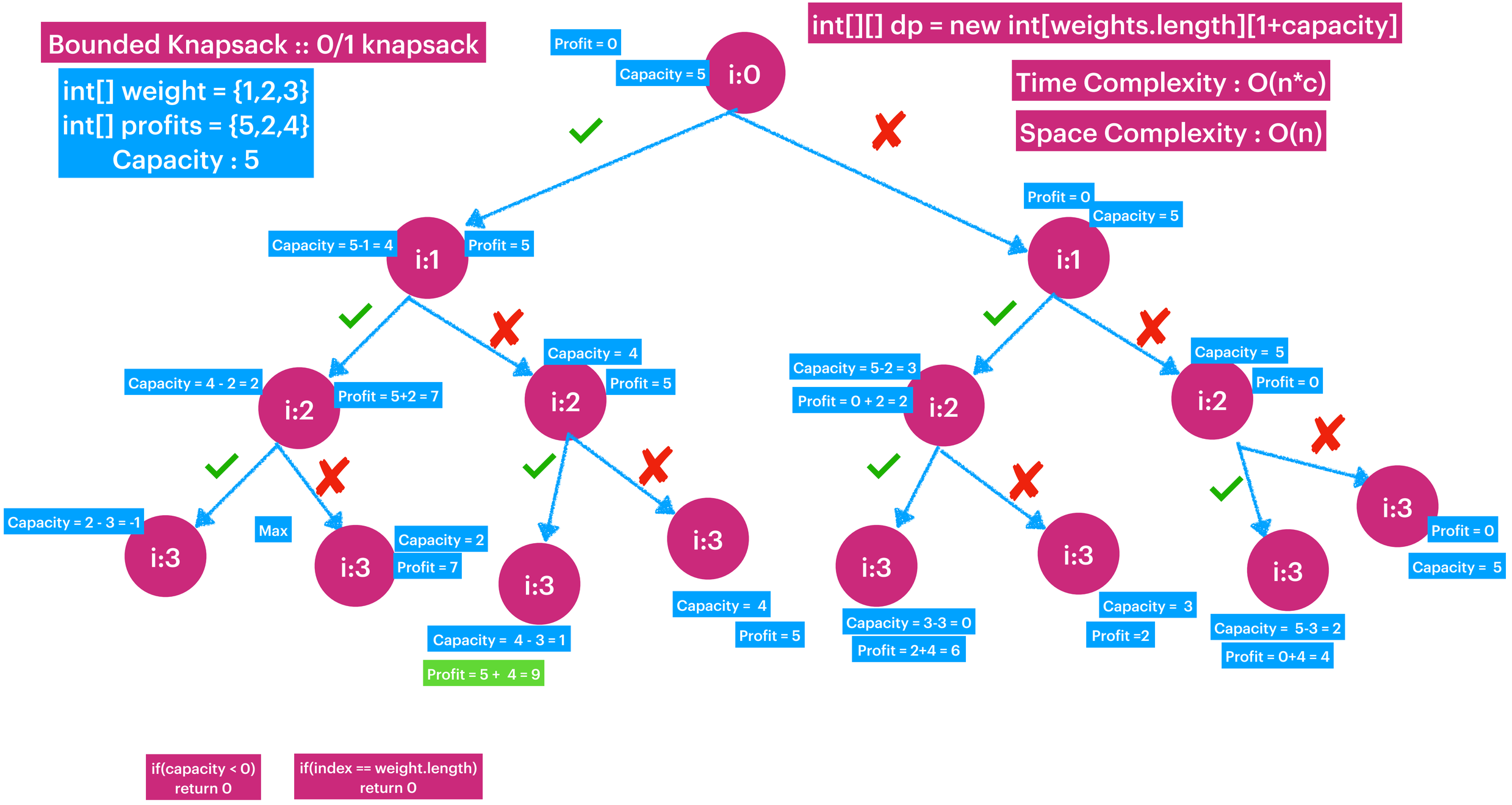
Item weight {1,2,3}
Item Profits {5,2,4}
Capacity : 5
Max Profit we can gain by choosing items {1,3} & respective profits {5,4} = 9

Capacity left in bag is 1. As we chosen
Item Weights {1,3} = 4

Item weight {1,2,3}
Item Profits {5,2,4}
Capacity : 5
Max Profit we can gain by choosing item {1} 5 times & profit is 5 * 5 = 25.

Capacity left in bag is 0. As we chosen item 1 , five times.


Bounded Knapsack :: 0/1 knapsack
int[] weight = {1,2,3}
int[] profits = {5,2,4}
Capacity : 5
Time Complexity : O(2^n)
Space Complexity : O(n)
Profit = 0
Capacity = 5
i:0
Capacity = 5-1 = 4
i:1
Profit = 5
Profit = 0
Capacity = 5
i:1
Capacity = 4 - 2 = 2
i:2
Profit = 5+2 = 7
Capacity = 4
i:2
Profit = 5
Capacity = 5-2 = 3
Profit = 0 + 2 = 2
i:2
Capacity = 5
i:2
Profit = 0
Capacity = 2 - 3 = -1
i:3
Max
i:3
Capacity = 2
Profit = 7
i:3
Capacity = 4 - 3 = 1
Profit = 5 + 4 = 9
i:3
Capacity = 4
Profit = 5
i:3
Capacity = 3-3 = 0
Profit = 2+4 = 6
i:3
Capacity = 3
Profit =2
i:3
Capacity = 5-3 = 2
Profit = 0+4 = 4
i:3
Profit = 0
Capacity = 5
if(capacity < 0)
return 0
if(index == weight.length)
return 0


Bounded Knapsack :: 0/1 knapsack
int[] weight = {1,2,3}
int[] profits = {5,2,4}
Capacity : 5
int[][] dp = new int[weights.length][1+capacity]
Time Complexity : O(n*c)
Space Complexity : O(n)
Profit = 0
Capacity = 5
i:0
Capacity = 5-1 = 4
i:1
Profit = 5
Profit = 0
Capacity = 5
i:1
Capacity = 4 - 2 = 2
i:2
Profit = 5+2 = 7
Capacity = 4
i:2
Profit = 5
Capacity = 5-2 = 3
Profit = 0 + 2 = 2
i:2
Capacity = 5
i:2
Profit = 0
Capacity = 2 - 3 = -1
i:3
Max
i:3
Capacity = 2
Profit = 7
i:3
Capacity = 4 - 3 = 1
Profit = 5 + 4 = 9
i:3
Capacity = 4
Profit = 5
i:3
Capacity = 3-3 = 0
Profit = 2+4 = 6
i:3
Capacity = 3
Profit =2
i:3
Capacity = 5-3 = 2
Profit = 0+4 = 4
i:3
Profit = 0
Capacity = 5
if(capacity < 0)
return 0
if(index == weight.length)
return 0