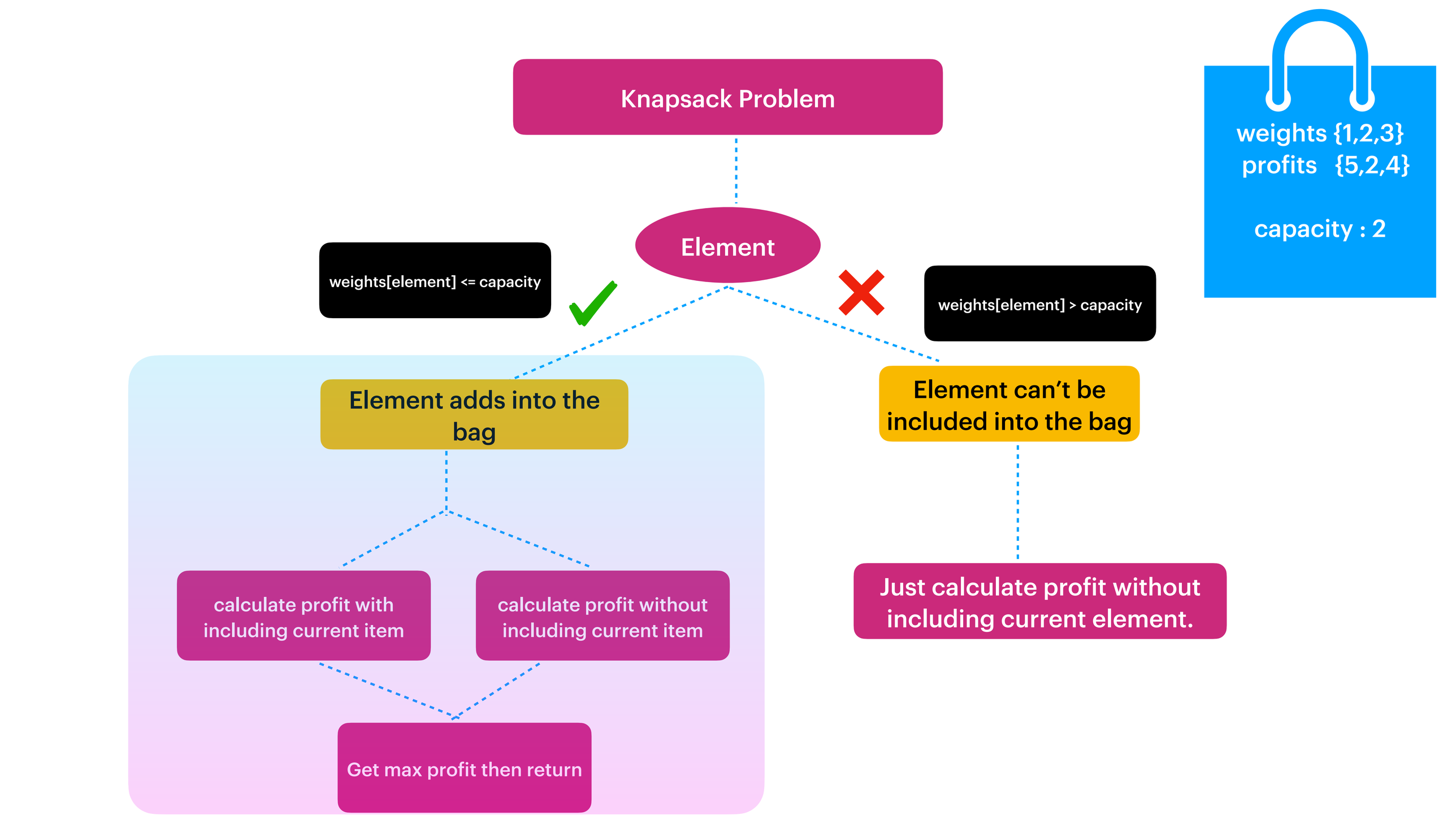

Knapsack Problem
Element
weights[element] <= capacity
weights[element] > capacity
Element adds into the bag
Element can’t be included into the bag
calculate profit with including current item
calculate profit without including current item
Just calculate profit without including current element.
Get max profit then return
weights {1,2,3}
profits {5,2,4}
capacity : 2

Problem Statement : Given a set of positive numbers, determine if a subset exists whose sum is equals to a given number ‘S’.
Ex : {1,3,5} , S = 8

Expected Output : True
Because we have subset {3,5}

Ex: {1,3,6} , S =8
Expected Output : False
Because we don’t find any subset.

Problem Statement : Given a set of positive numbers, determine if a subset exists whose sum is equals to a given number ‘S’.
Ex : {1,3,5} , S = 8
Expected Output :
Ex: True. Its possible with {3,5}

Ex: {1,3,5}, S = 8

Let me take up the all the possible combination subsets for the given input {1,3,5}

Total Combinations 8. As array size = 3 we got 2^3 combinations.

With empty Set always Zero

{} => sum = 0

Choosing 1 element at a time

{1} => sum = 1
{3} => sum = 3
{5} => sum = 5

Choosing 2 elements at a time

{1,3} => sum = 4
{1,5} => sum = 6
{3,5} => sum = 8

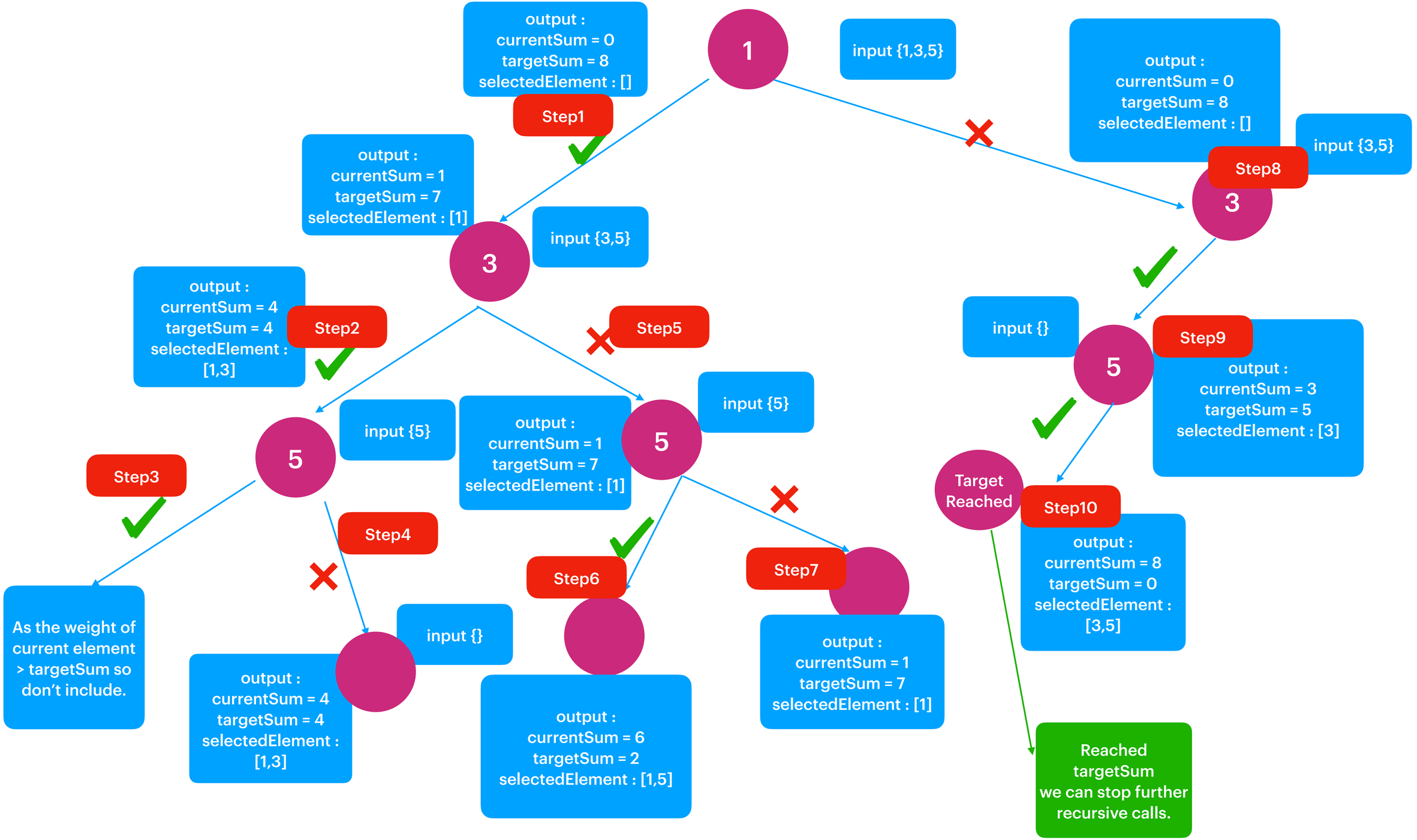
Choosing 2 elements at a time

{1,3,5} => sum = 9

As the targetSum is 8, we ca with subset {3,5}

How to solve ? To get into the solution, we should see all the possible combinations (i.e) subproblems.

Solution1 => Recursion


output :
currentSum = 0
targetSum = 8
selectedElement : []
1
input {1,3,5}
Step1
output :
currentSum = 1
targetSum = 7
selectedElement : [1]
3
input {3,5}
output :
currentSum = 0
targetSum = 8
selectedElement : []
Step8
input {3,5}
3
output :
currentSum = 4
targetSum = 4
selectedElement :
[1,3]
Step2
Step5
input {}
5
Step9
output :
currentSum = 3
targetSum = 5
selectedElement : [3]
input {5}
5
input {5}
output :
currentSum = 1
targetSum = 7
selectedElement : [1]
Target
Reached
Step10
output :
currentSum = 8
targetSum = 0
selectedElement :
[3,5]
Step3
Step4
Step6
Step7
As the weight of
current element
> targetSum so
don't include.
input {}
output :
currentSum = 4
targetSum = 4
selectedElement :
[1,3]
output :
currentSum = 6
targetSum = 2
selectedElement : [1,5]
output :
currentSum = 1
targetSum = 7
selectedElement : [1]
Reached
targetSum
we can stop further
recursive calls.