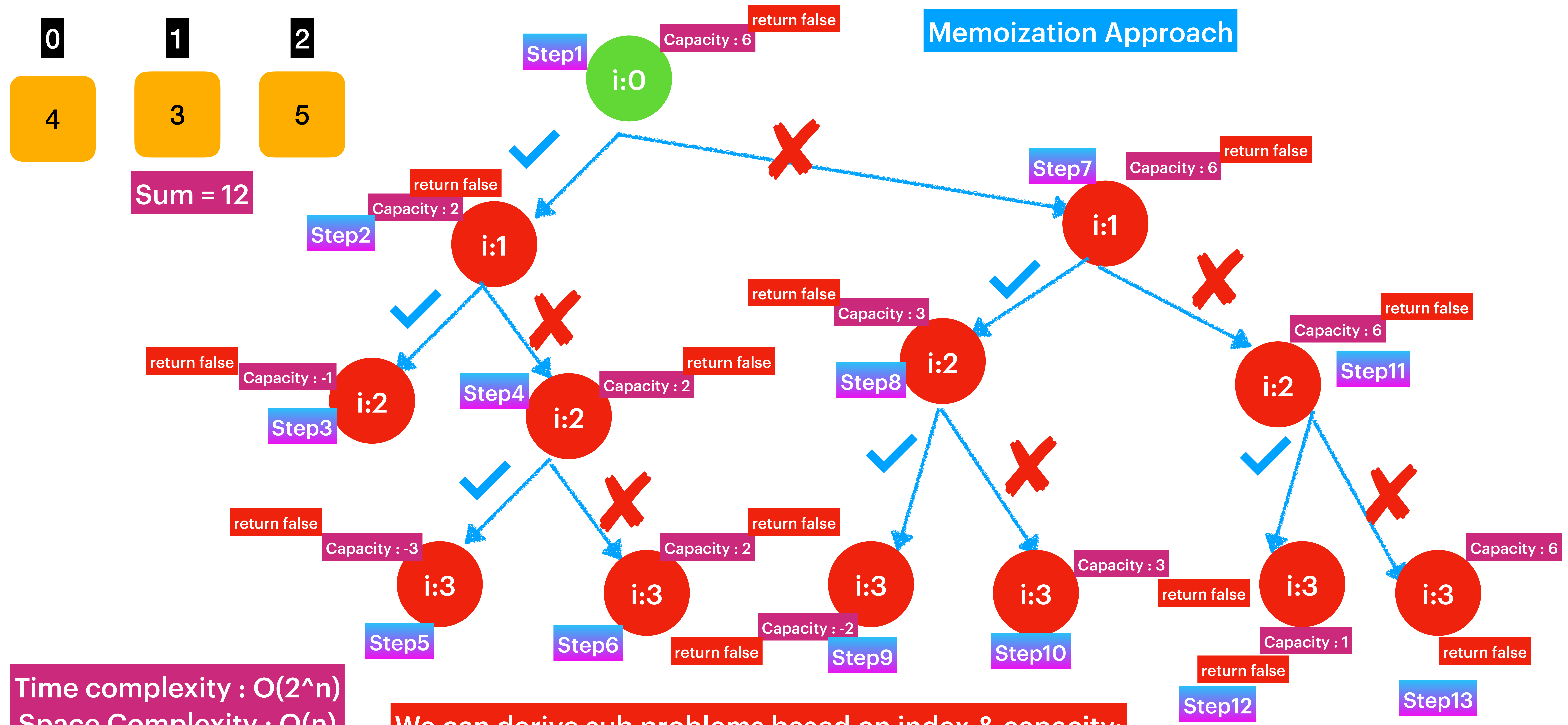
Memoization Approach



Time complexity : O(2^n)
Space Complexity : O(n)

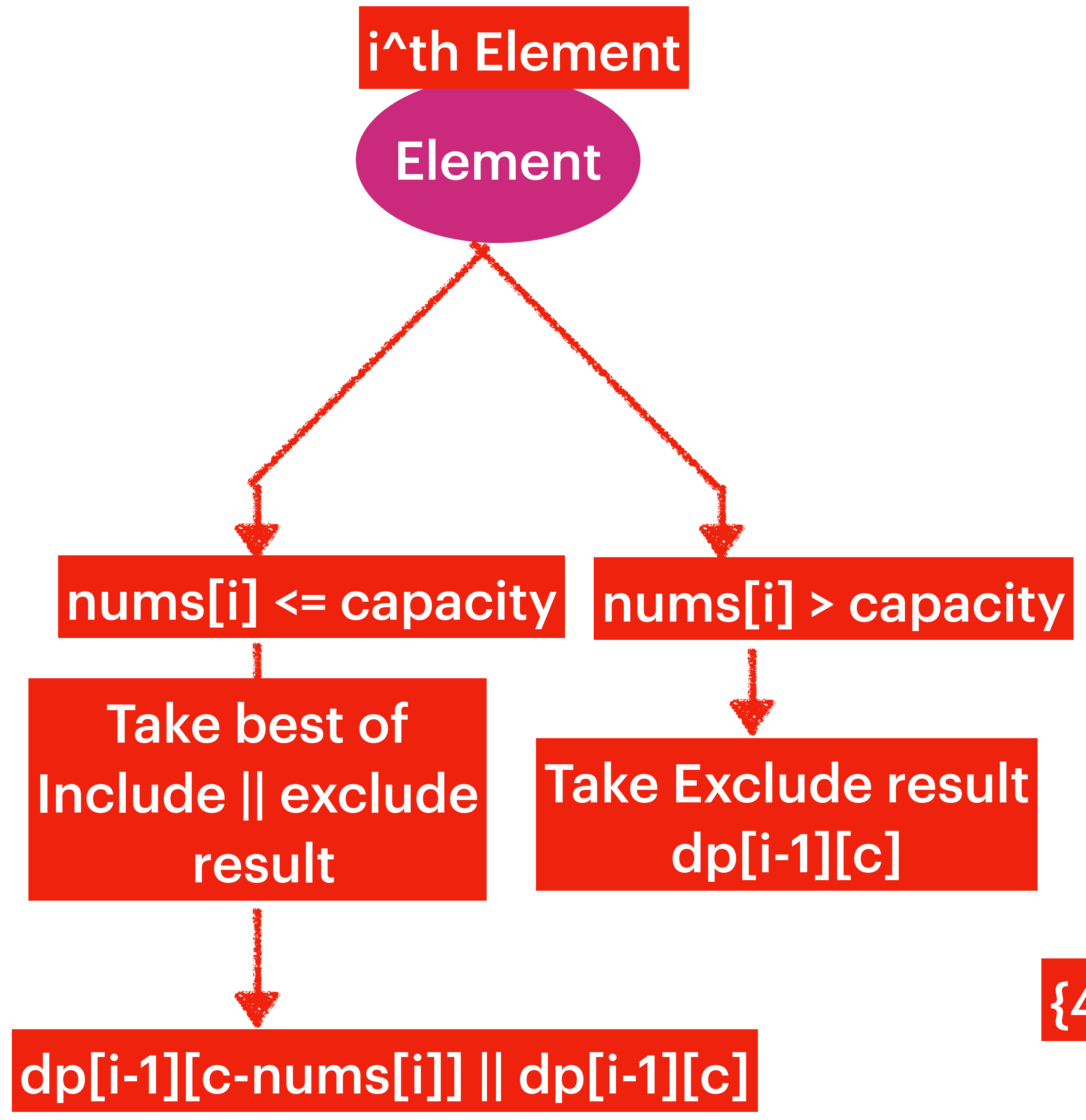
We can derive sub problems based on index & capacity:
dp[index][capacity].
Possible indexes are 0 to nums.length-1
And possible targetCapacity is 0 To capacity
int[][] dp = new int[nums.length][capacity+1]

Time Complexity : O(MN)
Space Complexity :O(N)

0	1	2
4	3	5

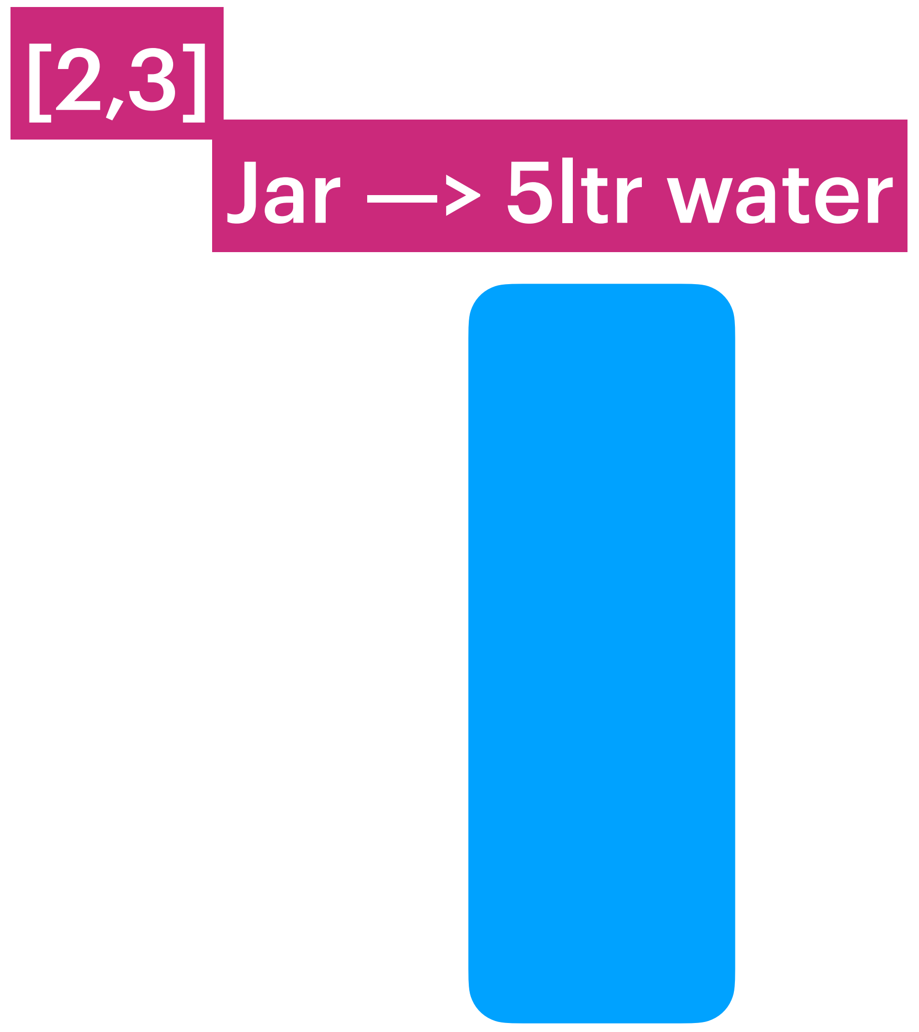
Total Sum = 12 , SubSetSum or Capacity = 6

Tabulation Approach : int[][] dp = new int[nums.length+1] [subSetSum+1]



i^th Element
Element
nums[i] <= capacity
nums[i] > capacity
Take best of Include || exclude result
Take Exclude result dp[i-1][c]
dp[i-1][c-nums[i]] || dp[i-1][c]

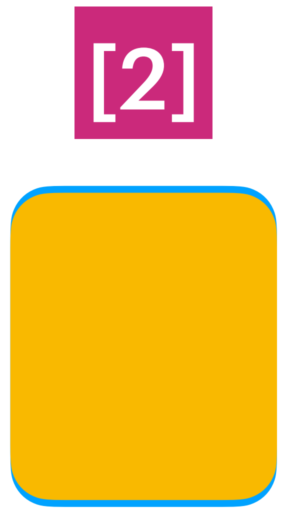
		SubSetSum						
		0	1	2	3	4	5	6
{ }	0	True	False	False	False	False	False	False
{4}	1	True	False	False	False	True	False	False
{4,3}	2	True	False	False	True	False	False	False
{4,3,5}	3	True	False	False	True	False	True	False

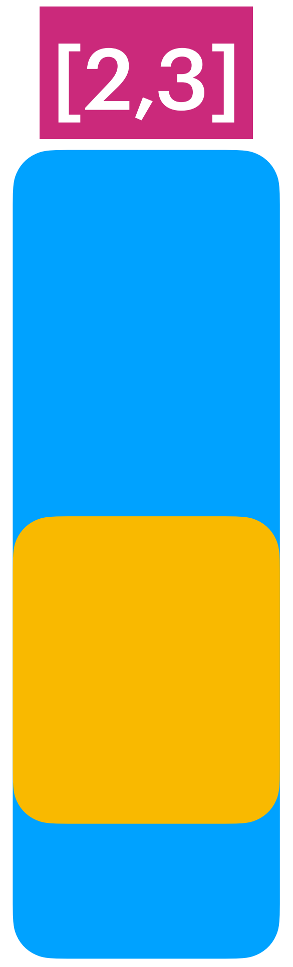


dp[0][2] = true

{2[0],3[1]}

dp[1][5] = dp[0][2] ; —-> dp[i-1][c-nums[i]]

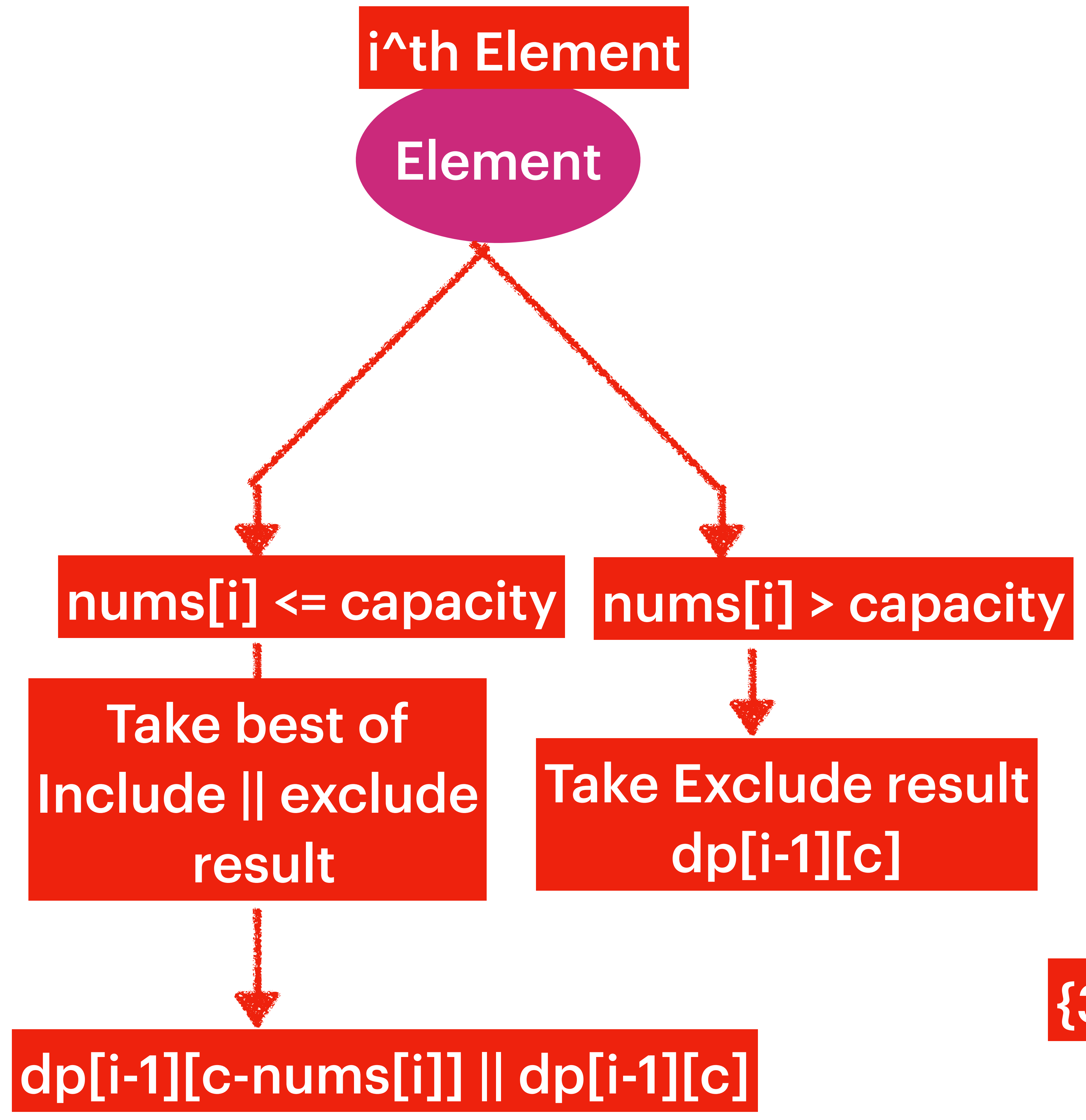




0	1	2
3	1	4

Total Sum = 8 , SubSetSum or Capacity = 4

Tabulation Approach : int[][] dp = new int[nums.length+1] [subSetSum+1]



i^th Element
Element
nums[i] <= capacity
nums[i] > capacity
Take best of Include || exclude result
Take Exclude result dp[i-1][c]
dp[i-1][c-nums[i]] || dp[i-1][c]

SubSetSum

		0	1	2	3	4
{ }	0	True	False	False	False	False
{3}	1	True	False	False	True	False
{3,1}	2	True	True	False	False	True
{3,1,4}	3	True	True	False	False	True