**Justin\_Array\_Backtracking\_0039.**  **Combination Sum**

**Concept:**

利用 recursive 一一加入，總和等於 target，則加入 result

只要總和超過 target 則 return

# targer = 7 [2, 2, 2, 2] ==> return 找 [2, 2, 3]

**Code:**

class Solution:

def combinationSum(self, candidates: List[int], target: int) -> List[List[int]]:

## RC ##

## APROACH : BRACKTRACKING ##

## REFERECE : https://www.youtube.com/watch?v=irFtGMLbf-s ##

def dfs(curr, nums):

if(sum(curr) > target):

return

if(sum(curr) == target):

#answer found

result.append(curr[:])

return

for i in range(len(nums)):

dfs(curr + [nums[i]], nums[i:]) # pass nums from this i to end of the list ( i also coz we can have [2,2,2,2] for total = 8 )

# if you pass nums directly you will get combinations like [2,2,3] and

# when you are traversing starting with 3 you will get [3,2,2] which is not required. so nums[i:]

# Instead of this, you can also use index variable

result = []

dfs([], candidates)

return result