39. Combination Sum

Given an array of **distinct** integers `candidates` and a target integer `target`, return *a list of all **unique combinations** of* `candidates` *where the chosen numbers sum to* `target`*.* You may return the combinations in **any order**.

The **same** number may be chosen from `candidates` an **unlimited number of times**. Two combinations are unique if the frequency of at least one of the chosen numbers is different.

It is **guaranteed** that the number of unique combinations that sum up to `target` is less than `150` combinations for the given input.

Example 1:

```
Input: candidates = [2,3,6,7], target = 7
Output: [[2,2,3],[7]]
Explanation:
2 and 3 are candidates, and 2 + 2 + 3 = 7. Note that 2 can be used
multiple times.
7 is a candidate, and 7 = 7.
These are the only two combinations.
```

Example 2:

```
Input: candidates = [2,3,5], target = 8
Output: [[2,2,2,2],[2,3,3],[3,5]]
```

Example 3:

```
Input: candidates = [2], target = 1
Output: []
```

Example 4:

```
Input: candidates = [1], target = 1
Output: [[1]]
```

Example 5:

```
Input: candidates = [1], target = 2
Output: [[1,1]]
```

```python
class Solution:
    def combinationSum(self, candidates: List[int], target: int) -> 
List[List[int]]:
        res = []
        self.combinationalSum(candidates,target,res,0,[])
        return res


    def combinationalSum(self,candidates,target,res,idx,asf):
        if target==0:
            temp = asf[:]
            res.append(temp)
            return
        if target<0 or idx==len(candidates):
            return
        else:
            asf.append(candidates[idx])
            self.combinationalSum(candidates,target-
candidates[idx],res,idx,asf)
            asf.pop()
            self.combinationalSum(candidates,target,res,idx+1,asf)
```