Combination Sum II Documentation

Description:

The "Combination Sum II" problem involves finding all unique combinations in a given set of candidate numbers where the numbers sum up to a specified target value. However, each number in the candidate list can be used only once in any combination. This problem is a variation of the classic combination sum problem but with the added constraint of uniqueness within combinations.

Function Signature:

def combinationSum2(candidates: List[int], target: int) -> List[List[int]]:

Parameters:

- **candidates:** A list of integers representing the candidate numbers from which combinations are formed.
- **target:** An integer representing the target sum that combinations should achieve.

Return Value:

- Returns a list of lists, where each inner list represents a unique combination of candidate numbers that sum up to the target value.

Approach:

1. Sort the candidate list to handle duplicates properly.
2. Implement a backtracking algorithm to explore all possible combinations.
3. During backtracking, maintain a `start` index to avoid duplicate combinations.
4. For each candidate, explore its inclusion in the combination and recursively explore other candidates to achieve the target sum.
5. Avoid duplicate combinations by skipping identical candidates during iteration.

Constraints:

- The length of the `candidates` list is between 1 and 100.
- Each candidate number is between 1 and 50.
- The target value is between 1 and 30.

Example Usage:

```
solution = Solution()

print(solution.combinationSum2([10,1,2,7,6,1,5], 8))  # Output: [[1, 1, 6], [1, 2, 5], [1, 7], [2, 6]]

print(solution.combinationSum2([2,5,2,1,2], 5))      # Output: [[1, 2, 2], [5]]
```