Documentation for Subsets Solution

Problem Description

Given an integer array nums of unique elements, the task is to return all possible subsets (the power set) of the array. The solution set must not contain duplicate subsets, and the subsets can be returned in any order.

Examples

Example 1:

- *Input:* nums = [1, 2, 3]
- *Output:* [[], [1], [2], [1, 2], [3], [1, 3], [2, 3], [1, 2, 3]]

Example 2:

- *Input:* nums = [0]
- *Output:* [[], [0]]

Constraints

- 1 <= nums.length <= 10
- -10 <= nums[i] <= 10
- All elements in nums are unique.

Solution Description

The solution uses a backtracking approach to generate all possible subsets of the given array nums. The backtracking algorithm allows us to explore all potential subsets by either including or excluding each element in nums.

Explanation

1. **Function Signature:**
   - The function subsets takes a list of integers nums as input and returns a list of lists, where each inner list represents a subset of nums.

2. **Backtracking Function:**
   - The inner function backtrack takes two arguments: start (the starting index for the current recursive call) and path (the current subset being constructed).
   - result.append(path[:]): The current subset (path) is added to the result list. path[:] creates a shallow copy to avoid reference issues.
   - The for-loop iterates over the remaining elements starting from start to the end of nums.

3. **Include and Exclude Mechanism:**
   - path.append(nums[i]): Adds nums[i] to the current subset.
   - backtrack(i + 1, path): Recursively calls backtrack with the next starting index, thus exploring subsets that include nums[i].
   - path.pop(): Removes nums[i] from the current subset to backtrack and explore subsets that do not include nums[i].

4. **Initialization and Execution:**

- result = []: Initializes the result list to store all subsets.
- backtrack(0, []): Initiates the backtracking process starting from index 0 with an empty subset.
- return result: Returns the complete list of subsets.

Usage

- To use the Solution class to find all subsets of a given array nums, create an instance of the class and call the subsets method with nums as the argument.

```
solution = Solution()

print(solution.subsets([1, 2, 3]))

# Output: [[], [1], [2], [1, 2], [3], [1, 3], [2, 3], [1, 2, 3]]
```

This code will generate all possible subsets of the array [1, 2, 3] as demonstrated in the examples.