Previous (/articles/binary-tree-zigzag-level-order-traversal/) Next (/articles/shift-2d-grid/)

78. Subsets (/problems/subsets/)

Dec. 29, 2019 | 96.4K views

Average Rating: 4.52 (103 votes)

Given a set of **distinct** integers, *nums*, return all possible subsets (the power set).

Note: The solution set must not contain duplicate subsets.

Example:

```
Input: nums = [1,2,3]
Output:
[
  [3],
  [1],
  [2],
  [1,2,3],
  [1,3],
  [2,3],
  [1,2],
  []
]
```

Solution

Solution Pattern

Let us first review the problems of Permutations / Combinations / Subsets, since they are quite similar to each other and there are some common strategies to solve them.

First, their solution space is often quite large:

- Permutations (https://en.wikipedia.org/wiki/Permutation#k-permutations_of_n): $N!$.
- Combinations (https://en.wikipedia.org/wiki/Combination#Number_of_k-combinations): $C_N^k = \frac{N!}{(N-k)!k!}$
- Subsets: 2^N, since each element could be absent or present.

Given their exponential solution space, it is tricky to ensure that the generated solutions are ***complete*** and ***non-redundant***. It is essential to have a clear and easy-to-reason strategy.

There are generally three strategies to do it:

- Recursion
- Backtracking
- Lexicographic generation based on the mapping between binary bitmasks and the corresponding
  permutations / combinations / subsets.

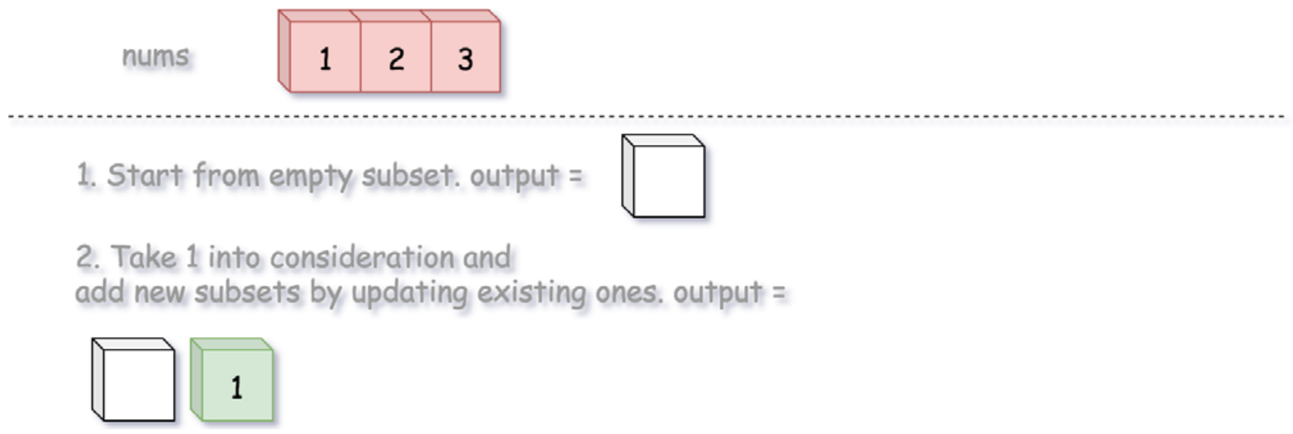
As one would see later, the third method could be a good candidate for the interview because it simplifies the problem to the generation of binary numbers, therefore it is easy to implement and verify that no solution is missing.

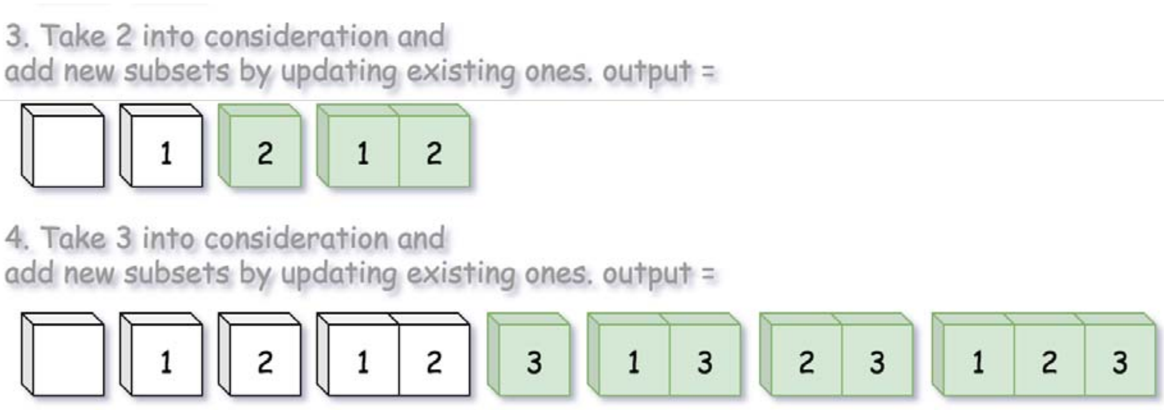
Besides, this method has the best time complexity, and as a bonus, it generates lexicographically sorted output for the sorted inputs.

Approach 1: Cascading

Intuition

Let's start from empty subset in output list. At each step one takes new integer into consideration and generates new subsets from the existing ones.


nums
1
2
3
1. Start from empty subset. output =
2. Take 1 into consideration and
add new subsets by updating existing ones. output =
1


3. Take 2 into consideration and
add new subsets by updating existing ones. output =
1
2
1 2
4. Take 3 into consideration and
add new subsets by updating existing ones. output =
1
2
1 2
3
1 3
2 3
1 2 3

Implementation

Java Python

```python
class Solution:
    def subsets(self, nums: List[int]) -> List[List[int]]:
        n = len(nums)
        output = [[]]

        for num in nums:
            output += [curr + [num] for curr in output]

        return output
```

Complexity Analysis

- Time complexity: $\mathcal{O}(N \times 2^N)$ to generate all subsets and then copy them into output list.
- Space complexity: $\mathcal{O}(N \times 2^N)$. This is exactly the number of solutions for subsets multiplied by the number N of elements to keep for each subset.
  - For a given number, it could be present or absent (*i.e.* binary choice) in a subset solution. As as result, for N numbers, we would have in total 2^N choices (solutions).

Approach 2: Backtracking

Algorithm