

Subsets (easy)

We'll cover the following ^

- Problem Statement
- Try it yourself
- Solution
- Code
 - Time complexity
 - Space complexity

Problem Statement

Given a set with distinct elements, find all of its distinct subsets.

Example 1:

Input: [1, 3]
Output: [], [1], [3], [1,3]

Example 2:

Input: [1, 5, 3]
Output: [], [1], [5], [3], [1,5], [1,3], [5,3], [1,5,3]

Try it yourself

Try solving this question here:

 Java

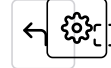
 Python3

 JS

 C++

```
1 def find_subsets(nums):
2     subsets = []
3     # TODO: Write your code here
4     return subsets
5
6
7 def main():
8
9     print("Here is the list of subsets: " + str(find_subsets([1, 3])))
10    print("Here is the list of subsets: " + str(find_subsets([1, 5, 3])))
11
12
13 main()
```





Solution

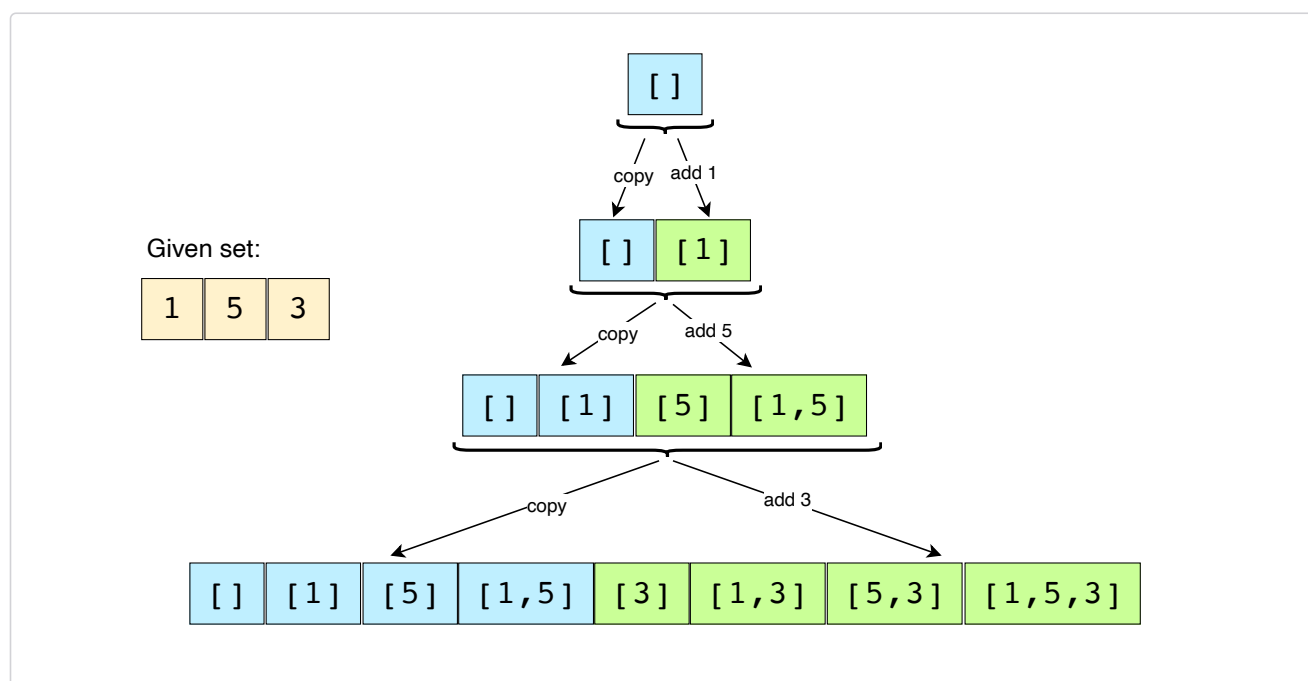
To generate all subsets of the given set, we can use the **Breadth First Search (BFS)** approach. We can start with an empty set, iterate through all numbers one-by-one, and add them to existing sets to create new subsets.

Let's take the example-2 mentioned above to go through each step of our algorithm:

Given set: [1, 5, 3]

1. Start with an empty set: `[]`
2. Add the first number (1) to all the existing subsets to create new subsets: `[]`, `[1]`;
3. Add the second number (5) to all the existing subsets: `[]`, `[1]`, `[5]`, `[1,5]`;
4. Add the third number (3) to all the existing subsets: `[]`, `[1]`, `[5]`, `[1,5]`, `[3]`, `[1,3]`, `[5,3]`, `[1,5,3]`.

Here is the visual representation of the above steps:



Since the input set has distinct elements, the above steps will ensure that we will not have any duplicate subsets.

Code

Here is what our algorithm will look like:

Java

Python3

C++

JS

```
1 def find_subsets(nums):
2     subsets = []
3     # start by adding the empty subset
```



```

4 subsets.append([])
5 for currentNumber in nums:
6     # we will take all existing subsets and insert the current number in them to create new subsets
7     n = len(subsets)
8     for i in range(n):
9         # create a new subset from the existing subset and insert the current element to it
10        set = list(subsets[i])
11        set.append(currentNumber)
12        subsets.append(set)
13
14    return subsets
15
16
17 def main():
18
19     print("Here is the list of subsets: " + str(find_subsets([1, 3])))
20     print("Here is the list of subsets: " + str(find_subsets([1, 5, 3])))
21
22
23 main()
24

```



Time complexity

Since, in each step, the number of subsets doubles as we add each element to all the existing subsets, the time complexity of the above algorithm is $O(2^N)$, where 'N' is the total number of elements in the input set. This also means that, in the end, we will have a total of $O(2^N)$ subsets.

Space complexity

All the additional space used by our algorithm is for the output list. Since we will have a total of $O(2^N)$ subsets, the space complexity of our algorithm is also $O(2^N)$.

← Back

Introduction

Next →

Subsets With Duplicates (easy)



Mark as Completed



Report an Issue



Ask a Question

(https://discuss.educative.io/tag/subsets-easy__pattern-subsets__grokking-the-coding-interview-patterns-for-coding-questions)