CompositeNums

Wednesday, 11 March 2020 10:36 am

Question
Given a set of prime numbers return all possible composite numbers from the given set (given numbers included).
For example :
Input: [2, 3, 5]
Output : [1, 2, 3, 5, 6, 10, 15, 30]

Problem Understanding:
Here we are presented with an input array of prime numbers and we need to find all possible composite numbers from that given set.

What is composite numbers?
Numbers composed as a product of other numbers. A composite number has factors in addition to one and itself.

Like 6 = 2x3 apart from 1x6 = 6 , so it composite

A prime number is a whole number that only has two factors which are itself and one.
like 2 = 1x2 = 2

Intuition:
Rather than going into the details of prime and composite, the first thing to understand is how the output array is formed from the input array.

So, it must include the input number itself like 2,3,5

How 6 is formed : 2x3 = 6
10 : 5x2
15 : 5x3
30 : 2x3x5

Using this, it feels like a combination kind of problem like if we start from 2, the combination that can be formed
2
2x3

2x3x5
2x5

And if we start from 3
3
3x5

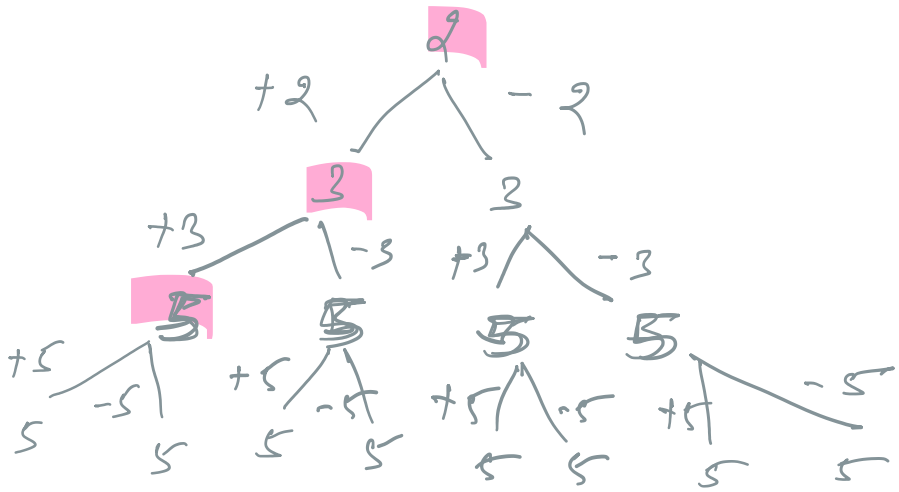
And if we start from 5
5

and have to add 1 to the output.

Getting the above combination from for loop is a bit tricky like for instance in case we start from 2
and have to get
2x5 will be tricky and even trickier when we have large number of inputs.

Q) So how to generate those combination?
The other option we can choose for those combination is to use the recursion tree where we will have two options at each stage, i.e. either to choose a given number at a index or not to choose a given number at index.

For instance, + indicates we have chosen the number (i.e. on left) and - indicates we didn't chose a number

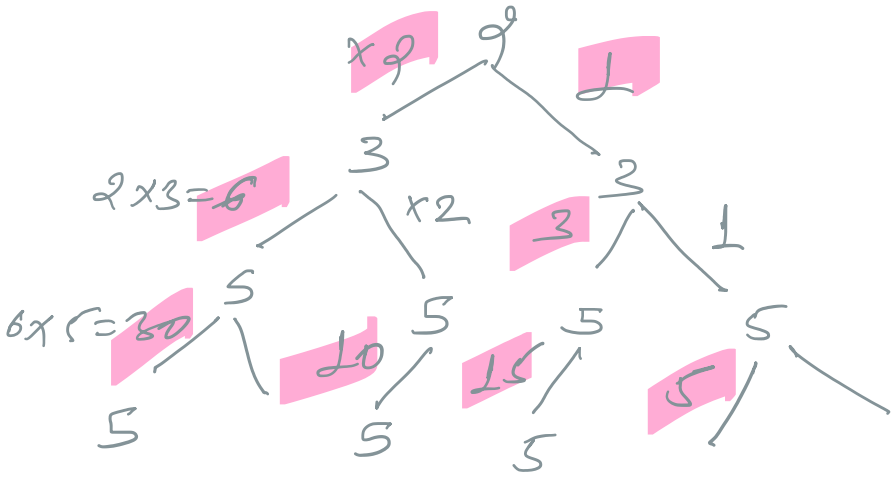


For instance if we look at +2 +3 +5 means
we have combination 2,3,5

and if we look at +2 -3 +5 means we have combination
  2,5

So we are able to generate all the possible combination from the recursion tree. If it seems confusing, just remember that we can generate all possible combination from a given input if we choose a recursion tree where we have a option to either choose or not choose a number at a given state.

So, going with the same recursion tree, our product will be



So, yes we have generated all the possible composite numbers

So our recursion formula will be

```
composite(index,product)
  //include this index in our prodcut
  composite(index+1, arr[index]*product)
  //don't include this index in our prodcut
  composite(index+1,product)
```

The time complexity will be
$O(2^n)$ as our recursion resembles a complete binary tree

Space Complexity will be O(n) as per the recursion diagram.

I am not 100% confident of complexity analysis.

So, overall code will be

```
public Set<Integer> getCompositeNums(int[] arr) {
        Set<Integer> compositeSet = new HashSet<>();
        findCompositeSet(0, 1, arr, compositeSet);
        return compositeSet;
    }
     private void findCompositeSet(int index, int
product, int[] arr, Set<Integer> compositeSet) {
        compositeSet.add(product);
        if (index >= arr.length)
            return;
        findCompositeSet(index + 1, product * arr[index],
arr, compositeSet);
        findCompositeSet(index + 1, product, arr,
compositeSet);
    }
```