APPRENTICE TECHNICAL LOG DOC

PERMUTATIONS

Leonardo Trevizo Herrera

OVERVIEW – In this document we will explore the process and challenges I had to overcome while working on the first breakable toy for the spark program  
  
CONTEXT – We want to get the permutations from a given array of numbers  
<https://github.com/technologic-technologic/LeetCode.git>  
  
SOLUTION (YOUR APPROACH) – I initially tried to implement a brute-force algorithm, but I noticed a pattern: I was constantly going back and forth, mixing numbers in various ways. That’s when I thought of using a recursive approach instead.

From my earlier brute-force attempt, I already knew I needed an index to keep track of the current position in the array so I could swap the current element with others. I also realized I needed a copy of the array to modify during the recursive calls. Later, I understood that I needed an additional array to store all the resulting permutations.

Right now, my solution is halfway there, but how do I properly implement the recursive function?

What I want to do is go through each of the elements in the array, and I imagine or visualize this like in the next diagram.  
A group of rectangular objects with text

AI-generated content may be incorrect.

So now, I can represent this abstraction into some human level kind of code The first call to the recursive function (represented by the upper yellow box in my diagram), and if you follow my line of thinking you see that this first call has three different possible numbers in the first element of the temporary, this first part of changing the first element with the different options that comes from the original array can be made with a simple for loop, while the downwards boxes can be made with recursiveness, calling the function itself, but with the elements switched from the previous step (the for loop step), after calling the function itself I decided that I needed to change the values back to original so I can really explore all the possible permutations. Now, when does the program stop? If not for a base case, the program would return an array with way too many arrays inside. So, what is the base case? You just must see the diagram, there you can see that it finishes when there is no more space to the right in the arrays, how can you represent this in code? Simple, I just compared the index to the length from the original array, if these values are equal, I append the auxiliar array. And it all finishes when I finished my first for loop.  
A screenshot of a computer

AI-generated content may be incorrect.

ALTERNATIVE SOLUTIONS – Other than the brute force sort of algorithm that probably every novice come up with (just like myself), my other idea was useless, but way more interesting. This idea was to calculate the number of permutations possible given the input array, then create an array “aux” to store the permutations, from here I just randomly mix the original array, check if this result is already in the aux array, if not store the results in the aux array. Do this until the length of the aux array is the same as the number of possible permutations. Pros: It’s an interesting way to solve this problem. Cons: Useless. Apart from being a creative way to solve the problem, short and even kind of elegant, it is useless, and very slow with large input arrays.