String Permutations by changing case (medium)

Problem Statement

Given a string, find all of its permutations preserving the character sequence but changing case.

Example 1:

```
Input: "ad52"
Output: "ad52", "Ad52", "aD52", "AD52"
```

Example 2:

```
Input: "ab7c"
Output: "ab7c", "Ab7c", "aB7c", "AB7c", "ab7C", "Ab7C", "aB7C", "AB7C"
```

Try it yourself

Try solving this question here:

☕ Java	🐍 Python3	JS JS	C++ C++

```python
1  def find_letter_case_string_permutations(str):
2    permutations = []
3    # TODO: Write your code here
4    return permutations
5
6
7  def main():
8    print("String permutations are: " +
9          str(find_letter_case_string_permutations("ad52")))
10   print("String permutations are: " +
11         str(find_letter_case_string_permutations("ab7c")))
12
14   main()
```
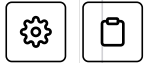
≡ 🖥 educative

Solution

This problem follows the Subsets
(https://www.educative.io/collection/page/5668639101419520/5671464854355968/567024937861
1200) pattern and can be mapped to Permutations
(https://www.educative.io/collection/page/5668639101419520/5671464854355968/572075819401
2160/).

Let's take Example-2 mentioned above to generate all the permutations. Following a **BFS**
approach, we will consider one character at a time. Since we need to preserve the character
sequence, we can start with the actual string and process each character (i.e., make it upper-
case or lower-case) one by one:

1. Starting with the actual string: `"ab7c"`

2. Processing the first character ('a'), we will get two permutations: `"ab7c"`, `"Ab7c"`

3. Processing the second character ('b'), we will get four permutations: `"ab7c"`, `"Ab7c"`,
   `"aB7c"`, `"AB7c"`

4. Since the third character is a digit, we can skip it.

5. Processing the fourth character ('c'), we will get a total of eight permutations: `"ab7c"`,
   `"Ab7c"`, `"aB7c"`, `"AB7c"`, `"ab7C"`, `"Ab7C"`, `"aB7C"`, `"AB7C"`

Let's analyze the permutations in the 3rd and the 5th step. How can we generate the
permutations in the 5th step from the permutations in the 3rd step?

If we look closely, we will realize that in the 5th step, when we processed the new character
('c'), we took all the permutations of the previous step (3rd) and changed the case of the letter
('c') in them to create four new permutations.

Here is the visual representation of this algorithm:

Given string: **"ab7c"**
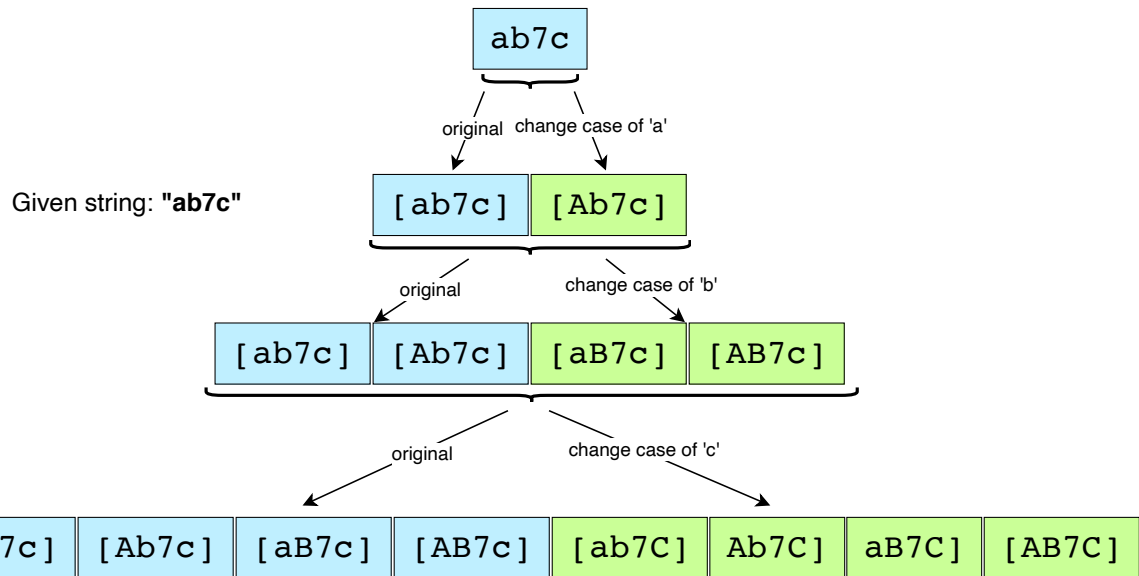
Code

Here is what our algorithm will look like:

```python
def find_letter_case_string_permutations(str):
  permutations = []
  permutations.append(str)
  # process every character of the string one by one
  for i in range(len(str)):
    if str[i].isalpha():  # only process characters, skip digits
      # we will take all existing permutations and change the letter case appropriately
      n = len(permutations)
      for j in range(n):
        chs = list(permutations[j])
        # if the current character is in upper case, change it to lower case or vice versa
        chs[i] = chs[i].swapcase()
        permutations.append(''.join(chs))

  return permutations


def main():
  print("String permutations are: " +
        str(find_letter_case_string_permutations("ad52")))
  print("String permutations are: " +
        str(find_letter_case_string_permutations("ab7c")))


main()
```

educative

Time complexity

Since we can have 2^N permutations at the most and while processing each permutation we convert it into a character array, the overall time complexity of the algorithm will be $O(N * 2^N)$.

Space complexity

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

☑ **Mark as Completed**

⊘ Report an Issue

⁇ Ask a Question (https://discuss.educative.io/tag/string-permutations-by-changing-case-medium__pattern-subsets__grokking-the-coding-interview-patterns-for-coding-questions)

Could not connect to the reCAPTCHA service. Please check your internet connection and reload to get a reCAPTCHA challenge.