#1: Permutations of a String – (CS 4A: Fall 2024)

Due Date: 10/27/24

The problem statement:

Given a string, find all the permutations of the characters in the string, and return them in a list.

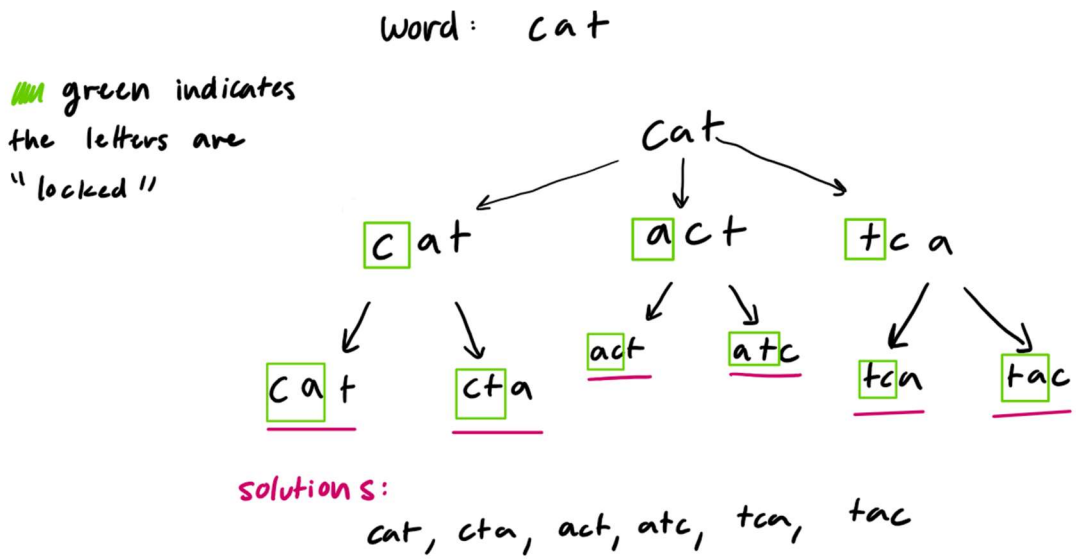
1. High-Level Description

To solve this problem, the string is turned into a character array, which will be manipulated to achieve the various permutations. A for loop will run the swaps. When the function is first called, it will kickstart the recursive solution by creating the character array, path, and passing 0 for the starting point of the array. Subsequent calls run a for loop starting at the passed in index, swap the values at the passed index and the for-loop index, then make another call. This continues until the end of the string (array) is reached, in which case a copy of the array is saved to the list of strings. Then the calls return back, undoing the swaps as it goes backwards. This is the backtracking aspect, where swaps are made then undone so that the next indices in the array can be swapped, building up permutations.

2. Model

In between calls, the modified array, path, and starting index + 1 are passed to the next call. This allows the swaps to continue without affecting previous characters. It is like the previous characters are locked in place in the array, so they will not move, while the characters to the right of the index (index + 1) can be swapped freely. See the models below:

Idea Model: Visual description of the model. Each branch is formed recursively.



3. Java Source Code / 4. Testing Program

```java
import java.util.ArrayList;

public class SolutionOne {
    public static void main(String[] args) {
        String test1 = "cat";
        ArrayList<String> result1 = PermutationOfAString(test1);
        PrintList(result1);
        System.out.println();

        String test2 = "word";
        ArrayList<String> result2 = PermutationOfAString(test2);
        PrintList(result2);
        System.out.println();

        String test3 = "012";
        ArrayList<String> result3 = PermutationOfAString(test3);
        PrintList(result3);
        System.out.println();


    } // end main

    public static ArrayList<String> PermutationOfAString(String str) {
        ArrayList<String> list = new ArrayList<>();
        char[] arr = str.toCharArray();

        System.out.println("Permutations of " + str + ": ");
        Perm(arr, list, 0);

        return list;
    } // END PermutationOfAString

    public static void Perm(char[] arr, ArrayList<String> list, int index) {

        if(index == arr.length) {
            list.add(String.valueOf(arr));
        }

        for (int i = index; i < arr.length; i++) {
            Swap(index, i, arr);

            Perm(arr, list, index + 1);

            Swap(index, i, arr);
        }

    } // END Perm (helper)

    public static void Swap(int index1, int index2, char[] arr) {
        char temp = arr[index1];
        arr[index1] = arr[index2];
        arr[index2] = temp;
    } // END Swap
```

```
    public static void PrintList(ArrayList<String> list) {
        for (int i = 0; i < list.size(); i++) {
            System.out.println(list.get(i));
        }
    } // END PrintList

} // END SolutionOne
```

5. Example Inputs and Outputs

Input: cat

Output: cat, cta, act, atc, tac, tca

Input: word

Output: word, word, word, wrdo, wdro, wdor, owrd, owdr, orwd, ordw, odrw, odwr, rowd, rodw, rwod, rwdo, rdwo, rdow, dorw, dowr, drow, drwo, dwro, dwor

Input: 012

Output: 012, 021, 102, 120, 210, 201

- **Examples Directly from Running Program:**

```
Permutations of cat:
cat
cta
act
atc
tac
tca

Permutations of word:
word
wodr
wrod
wrdo
wdro
wdor
owrd
owdr
orwd
ordw
```

```
odrw
odwr
rowd
rodw
rwod
rwdo
rdwo
rdow
dorw
dowr
drow
drwo
dwro
dwor

Permutations of 012:
012
021
102
120
210
201


Process finished with exit code 0
```