All permutations of a set

Published 23 June 2013, updated 21 June 2015

Algorithm **Computer science** **Interview question** **Java** **JavaScript** **Open source**

This article looks at the interview question - *Implement a function that gets all possible permutations (or orderings) of the characters in a string. For example for the input string `"abc"`, the output will be `"abc"`, `"acb"`, `"bac"`, `"bca"`, `"cab"` and `"cba"`.*

Analysis

This problem is very similar to all combinations of a set, though the actual computing of the values will be quite different. Let's start by defining the inputs and outputs.

```
ArrayList<String> getPermutations(String characters);
```

Now let's look at how this problem is naturally solved. When I write down a set of permutations by hand, I tend to start with the first letter (`a`), and then find all permutations without that letter in it. So for `"abc"` I would write:

```
a bc
a cb
b ac
b ca
c ab
c ba
```

This approach can be translated exactly into a recursive function in which for all letters in a string, we pull the letter out of the string and prepend it to all permutations of the string without that letter in it. The base case when the string is a single character will return the character.

```
permutations(abc) = a + permutations(bc) +
                    b + permutations(ac) +
                    c + permutations(ab)

permutations(ab)  = a + permutations(b) +
                    b + permutations(a)

permutations(a)   = a
```

Pseudocode

```
function getPermutations (string text)
  define results as string[]
  if text is a single character
    add the character to results
    return results
  foreach char c in text
    define innerPermutations as string[]
    set innerPermutations to getPermutations (text without c)
    foreach string s in innerPermutations
      add c + s to results
  return results
```

Complexity

Much like all combinations of a set, the time and space complexity of the above algorithm should be the same as the number of items produced. The number of unique permutations of any set of size n is $n!$, therefore our algorithm is $O(n!)$.

Code

Java JavaScript

View source on GitHub

```
function getAllPermutationsOfASet(text) {
  var results = [];

  if (text.length === 1) {
    results.push(text);
    return results;
  }

  for (var i = 0; i < text.length; i++) {
    var first = text[i];
    var remains = text.substring(0, i) + text.substring(i + 1);
    var innerPermutations = getAllPermutationsOfASet(remains);
    for (var j = 0; j < innerPermutations.length; j++) {
      results.push(first + innerPermutations[j]);
    }
  }

  return results;
}
```

Textbooks

Here are two CS textbooks I personally recommend; the Algorithm Design Manual (Steven S. Skiena) is a fantastic introduction to data structures and algorithms without getting to deep into the maths side of things, and Introduction to Algorithms (CLRS) which provides a much deeper, math heavy look.

Share this page

More posts tagged Interview question

- Check if a binary tree is balanced
- Determine if a string is a palindrome
- Find the kth last element in a linked list
- Find the median of two sorted arrays
- Given random5(), implement random7()
- Implement a maximum value aware stack
- Implement a queue using 2 stacks
- Reverse a linked list
- Reverse a string
- The Fibonacci sequence

Comments

Follow me