All combinations of a set

Published 21 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 combinations (or subsets) of the characters in a string with length of at least one. For example for the input string `"abc"`, the output will be `"a"`, `"b"`, `"c"`, `"ab"`, `"ac"`, `"bc"` and `"abc"`.*

Analysis

Firstly we will define our method signature. It's fairly simple as it's described in the problem, the input is a `String` and the output is a list of `Strings`

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

While this sort of problem seems trivial at first, trying to describe an algorithm is not. When designing an algorithm like this it is very useful to look at some example input/output and look for patterns. Let's first try sorting the output by the length of the string, we'll use the example "abcd".

```
a    ab    abc    abcd
b    ac    abd
c    ad    acd
d    bc    bcd
     bd
     cd
```

No real pattern that would indicate a suitable algorithm seems to emerge from this, now let's try ordering it by the character the sequence begins with.

```
a       b      c     d
ab      bc     cd
ac      bd
ad      bcd
abc
acd
abd
abcd
```

Now we're getting somewhere, if you look closely at the above you'll notice a pattern. Starting from the right (`d`), the column to the left (`c`) contains the all combinations on the right plus the empty set with `c` added the start. This pattern follows with `b`and `c`, we add `b` to all items to the right (`c`, `cd`, `d`) and add `b`itself.

```
d row = d
c row = c+{}, c+d
      = c, cd
b row = b+{}, b+d, b+c, b+cd
      = b, bd, bc, bcd
c row = a+{}, a+d, a+c, a+cd, a+b, a+bc, a+bd, a+bcd
      = a, ad, ac, acd, ab, abc, abd, abcd
```

This indicates a potential algorithm that will solve the problem.

Pseudocode

Given the analysis above, we can come up with a recursive algorithm that will solve the problem.

```
function getCombinations (string text)
  define results as string[]
  foreach char c in text
    foreach string r in results
      add c + r to results
    add c to results
  return results
```

Complexity

While determining the complexity of a function like the above may seem intimidating, you can take a different approach to it and look at it in terms of the output. Looking at the function, it is apparent that no additional work in done in each for loop other than generating the strings. Therefore the time and space complexity should be in the order of the number of combinations produced, namely $O(2^n)$ as our algorithm produces exactly $2^n - 1$ combinations.

Code

Java JavaScript

View source on GitHub

```java
public static ArrayList<String> getCombinations(String text) {
    ArrayList<String> results = new ArrayList<String>();
    for (int i = 0; i < text.length(); i++) {
        // Record size as the list will change
        int resultsLength = results.size();
        for (int j = 0; j < resultsLength; j++) {
            results.add(text.charAt(i) + results.get(j));
        }
        results.add(Character.toString(text.charAt(i)));
    }
    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