Question 1.2 from CTCI: Check Permutations

January 31, 2018

Question

Given two strings, write a method to decide if one is a permutation of the other.

Explanation and Algorithm

Following are two solutions to the problem:

In the first solution, we determine if two strings are permutations of each other by sorting and then comparing the strings. First, we must note that two strings cannot be permutations of each other if they are not the same length. This is due to the property of permutations: strings that are permutations of each other have the same characters in different orders. Knowing this property of permutations, we can solve this problem by sorting the strings so that the characters are in the same order. Then, we can compare the strings. If they are equal, then we know that the strings are permutations of each other.

An alternate approach to this problem involves counting the occurrences of characters in each string and checking that the counts are equal. As in the first solution, we must check that the strings are of equal length. Then we can iterate through the first string and track the number of occurrences of each character in an int array. To check that the second string is a permutation of the first string, the number of occurrences of each character in the second string must be equal to the number of occurrences in the first string. To check this, we can iterate through the second string and decrement the number of occurrences for each character in the int array. If there are more occurrences of a character in the second array than in the first, we will know that the strings are not permutations of each other.

Hints

1. Consider the definition of a permutation. What specific property should two strings have to be permutations of each other?

2. Strings that are permutations of each other have the same number of occurrences for each character. You will need to check that this property holds for the given strings. How might you go about this?

3. How might sorting the strings help solve the problem?

4. Alternatively, consider how you might store occurrences of characters in the two strings. What data structure would you use for this?

Code

```
/* Solution 1 */
public String sort(String str){
   char[] content = str.toCharArray();
   java.util.Arrays.sort(content);
   return new String(content);
}

public boolean CheckPermutation(String s1, String s2){
   if(s1.length != s2.length){
      return false;
   }

   return sort(s1).equals(sort(s2));
}
```

```
/* Solution 2 */

public boolean CheckPermutation(String s, String t){

   if(s.length() != t.length()){
      return false;
   }

   int[] letters = new int[128];

   char[] s_array = s.toCharArray();
   for(char c: s_array){
      letters[c]++;
   }

   for(int i = 0; i < t.length(); i++){
      int c = (int) t.charAt(i);
      letters[c]--;
      if(letters[c] < 0){
```

```
            return false;
        }
    }

    return true;

}
```

Big O Analysis

1. Solution 1: The running time is *O(nlog(n))* where n is the length of the input strings.
2. Solution 2: The running time is *O(n)* where n is the length of the input strings.

Sources

Question, answer and other material taken from Cracking the Coding Interview 6th edition by Gayle Laakmann McDowell.