CTCI 1.5: One Away

October 25, 2017

Question

There are three types of edits that can be performed on strings: insert a character, remove a character, or replace a character. Given two strings, write a function to check if they are one edit (or zero edits) away.

For example:
pale, ple $->$ true
pales, pale $->$ true
pale, bale $->$ true
pale, bae $->$ false

Explanation and Algorithm

Since we are given the operations on a string, the brute force solution that comes to mind is to test insertion, removal, and replacement with each character and comparing. This would be most inefficient, so think of how we can optimize the brute force solution. Note that insertion is the inverse of removal and vice versa. Therefore, we can merge the check for removal and insertion into one step and do the check for replacement in a separate step. Based on the nature of these steps, the length of the string should be telling of what to check for. We will do three methods: one for what to check based on the string lengths, another for performing insert/removal on the string, and another for replacement. If it works, return true. Otherwise, return false.

Hints

1. Those of you new to strings, how will you loop through a string? What properties can you use to get numbers (index of a particular character or .charAt will be useful)
2. Once you figure out the brute force solution, think of how you can use StringBuilder and make another method.

3. Use this method to your advantage to separate the compressed string from the input!

Code

```
/*Answer 1*/

boolean oneEditAway(String first, String Second){
   if(first.length() == second.length()){
      return oneEditReplace(first,second);
    }else
      if(first.length()+1 == second.length()){
         return oneEditInsert(first,second);
        }else
         if(first.length()-1 == second.length()){
               return oneEditInsert(first,second);
            }
        return false;
}

boolean oneEditReplace(String first, String second){
   boolean foundDifference = false;
    for(int i = 0; i < first.length(); i++){
      if(first.charAt(i) != second.charAt(i)){
         if(foundDifference){
               return false;
            }
            foundDifference = true;
        }
    }
   return true;
}

boolean oneEditIndert(String first, String second){
   int index1 = 0;
    int index2 = 0;
   while(index2 < second.length() && index1 < first.length()){
      if(first.charAt(index1) != second.charAt(index2)){
         if(index1 != index2){
            return false;
            }
        }
        index2++;
    }else{
      index1++;
        index2++;
    }
    return true;
}
```

Run time analysis

This is pretty efficient! Let n equal the length of the shorter string. Therefore, it takes $O(n)$

Sources

.