AlgoExpert   Quad Layout   Java   12px   Sublime   Monokai   00:00:00

Prompt   Scratchpad   Our Solution(s)   Video Explanation   Run Code

Solution 1

```java
// Copyright © 2020 AlgoExpert, LLC. All rights reserved.

import java.util.*;

class Program {
  // O(n^2 + m) time | O(n + m) space
  public static String[] patternMatcher(String pattern, String str) {
    if (pattern.length() > str.length()) {
      return new String[] {};
    }
    char[] newPattern = getNewPattern(pattern);
    boolean didSwitch = newPattern[0] != pattern.charAt(0);
    Map<Character, Integer> counts = new HashMap<Character, Integer>();
    counts.put('x', 0);
    counts.put('y', 0);
    int firstYPos = getCountsAndFirstYPos(newPattern, counts);
    if (counts.get('y') != 0) {
      for (int lenOfX = 1; lenOfX < str.length(); lenOfX++) {
        double lenOfY =
            ((double) str.length() - (double) lenOfX * (double) counts.get('x'))
                / (double) counts.get('y');
        if (lenOfY <= 0 || lenOfY % 1 != 0) {
          continue;
        }
        int yIdx = firstYPos * lenOfX;
        String x = str.substring(0, lenOfX);
        String y = str.substring(yIdx, yIdx + (int) lenOfY);
        String potentialMatch = buildPotentialMatch(newPattern, x, y);
        if (str.equals(potentialMatch)) {
          return didSwitch ? new String[] {y, x} : new String[] {x, y};
        }
      }
    } else {
      double lenOfX = str.length() / counts.get('x');
      if (lenOfX % 1 == 0) {
        String x = str.substring(0, (int) lenOfX);
        String potentialMatch = buildPotentialMatch(newPattern, x, "");
        if (str.equals(potentialMatch)) {
          return didSwitch ? new String[] {"", x} : new String[] {x, ""};
        }
      }
    }
    return new String[] {};
  }

  public static char[] getNewPattern(String pattern) {
    char[] patternLetters = pattern.toCharArray();
    if (pattern.charAt(0) == 'x') {
      return patternLetters;
    }
    for (int i = 0; i < patternLetters.length; i++) {
      if (patternLetters[i] == 'x') {
        patternLetters[i] = 'y';
      } else {
        patternLetters[i] = 'x';
      }
    }
    return patternLetters;
  }

  public static int getCountsAndFirstYPos(char[] pattern, Map<Character, Integer> counts) {
    int firstYPos = -1;
    for (int i = 0; i < pattern.length; i++) {
      char c = pattern[i];
      counts.put(c, counts.get(c) + 1);
      if (c == 'y' && firstYPos == -1) {
        firstYPos = i;
      }
    }
    return firstYPos;
  }

  public static String buildPotentialMatch(char[] pattern, String x, String y) {
    StringBuilder potentialMatch = new StringBuilder();
    for (char c : pattern) {
      if (c == 'x') {
        potentialMatch.append(x);
      } else {
        potentialMatch.append(y);
      }
    }
    return potentialMatch.toString();
  }
}
```