String Substitution

String Substitution was one of the hard CodeEval challenges which had a success rate less than 40%.

Problem Description

Given a string S, and a list of strings of positive length, F1,R1,F2,R2,...,FN,RN, proceed to find in order the occurrences (left-to-right) of Fi in S and replace them with Ri. All strings are over alphabet { 0, 1 }. Searching should consider only contiguous pieces of S that have not been subject to replacements on prior iterations. An iteration of the algorithm should not write over any previous replacement by the algorithm.

Input Sample

Program should accept as its first argument a path to a filename. Each line in this file is one test case. Each test case will contain a string, then a semicolon and then a list of comma separated strings. eg.

10011011001;0110,1001,1001,0,10,11

Output Sample

For each line of input, print out the string after substitutions have been made. eg.

11100110

The above output is the result of following transformations:

```
10011011001 => 10100111001 [replacing 0110 with 1001]
10100111001 => 10100110    [replacing 1001 with 0]
   10100110 => 11100110    [replacing 10 with 11]
```

Solution

I used a two-level substitution algorithm to solve the problem. Two-level substitution prevents overwriting previous replacements during the replacement loop.

Pseudocode

```
STRINGSUB(inputString, find-replace)
  Map a marker character to each pair of find-replace strings using a
  Hash Map, MARKERS

  For each MARKER in MARKERS
    Get find string mapped to MARKER and replace it with MARKER in
the
    inputString

  For each MARKER in MARKERS
    Get replace string mapped to MARKER and replace MARKER with
    it in the inputString

  Return inputString
```

Algorithm Execution Example

Map find-replace pairs to marker characters

$$10011011001;\underset{A}{\underline{0110,1001}},\underset{B}{\underline{1001,0}},\underset{C}{\underline{10,11}}$$

First Level Substitution: Replace find strings with markers

```
10011011001 => 10A11001
   10A11001 => 10A1B
      10A1B => CA1B
```

Second Level Substitution: Replace markers with replacement string

```
   CA1B => C10011B
C10011B => C100110
C100110 => 11100110
```

Complexity Analysis

Since the algorithm loops only through the number of find-replace pairs, the complexity will be in the order of **n**, where **n** is the number of pairs of find-replace strings given. So we can conclude the complexity of the algorithm as **O(n)**

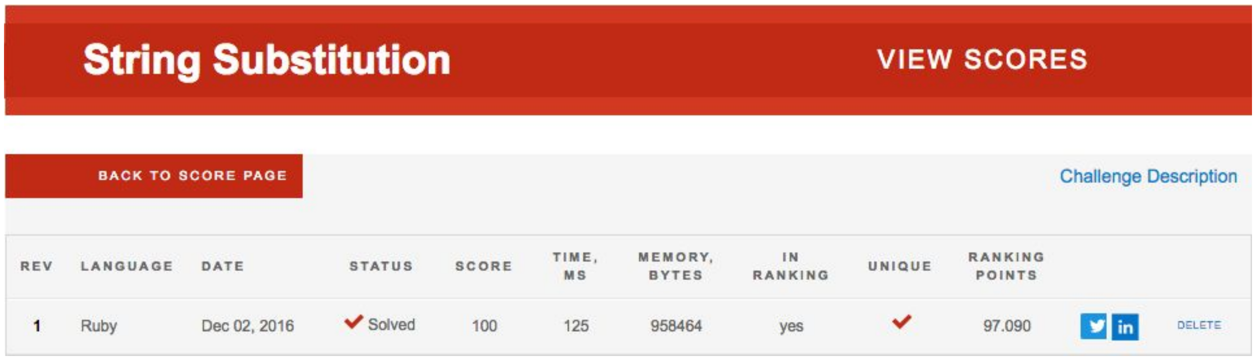
Please note that in the above analysis we treated string replacement function as constant time operation.

Implementation

Here is an implementation of the algorithm in Ruby.

```ruby
# Parses the input and transform the string with give find-replace pairs
# @param input <String> Input string with find-replace pairs
# @return <String> Transformed string
def strsub(input)
  # Parse input to separate input string and find-replace pairs
  str, find_replace = input.split(';')
  find_replace = find_replace.split(',').each_slice(2).to_a

  # Generate markers and map them to find-replace pairs
  markers = ('A'..'Z').to_a.first(find_replace.size)
  find_replace = markers.zip(find_replace).to_h

  # Replace find strings with markers
  find_replace.each do |marker,find_replace|
    find, replace = find_replace
    str = str.gsub(find,marker)
  end

  # Replace markers with replace strings
  find_replace.each do |marker,find_replace|
    find, replace = find_replace
    str = str.gsub(marker,replace)
  end

  return str
end
```

CodeEval Score



REV	LANGUAGE	DATE	STATUS	SCORE	TIME, MS	MEMORY, BYTES	IN RANKING	UNIQUE	RANKING POINTS
1	Ruby	Dec 02, 2016	✓ Solved	100	125	958464	yes	✓	97.090

Github

Code and this document is hosted at https://github.com/akhilstanis/CSCI174-Final-Project