Regex Practice: CPS 004G

Owen Astrachan

October 25, 2007

Name: ________________________________

Login: ______________

Honor code acknowledgment (signature) ________________________________
PROBLEM 1:  (Being Regular with Regex (22 points))

Part A (4 points)
We used a program in class that processed regular expressions and listed English words from a dictionary that matched the regular expression.

The regular expression (....)\1 generates exactly one match, the word beriberi. A small modification, using the regular expression (....)\1 generates two matches: bandstands and hodgepodge. Explain why beriberi does not match the second regular expression and why the second expression has the two matches it does.

Part B (4 points)
If the regular expression is changed to (....).*\1 then 13 matches are found as shown below.

atherosclerosis
bandstands
beriberi
hodgepodge
kinnickinnic
knickerbocker
knickerbockers
lightweight
misunderstander
misunderstanders
nationalization
rationalization
rationalizations

Explain broadly why there are more matches, including all three described in part A. Explain specifically why atherosclerosis matches. Finally, circle the five out of the thirteen that match (....).*\1 (note: one extra dot in the parentheses).
Part C (2 points)
The regular expression `sp[a1]s` generates twelve matches as follows. Explain why `despise` and `spasm` match this regular expression.

- despise
- despised
- despises
- despising
- dispassionate
- spasm
- spastic
- trespass
- trespassed
- trespasser
- trespassers
- trespasses

Part D (4 points)
This regular expression `^(.{2,4})\1$` generates a list of 10 words as follows. Based on your knowledge of regular expressions and your ability to analyze data, provide an explanation of how the regular expression works in generating this list. Note that we did not discuss the `{2,4}` part of the expression, you have to offer an explanation of this based on what you know and what the data show.

- beriberi
- booboo
- coco
- dada
- isis
- mama
- mimi
- murmur
- papa
- toto

Part E (4 points)
To find seven-letter palindromes, someone enters this regular expression `(.)(.)(.).\3\2\1$`. This generates seven matches, but only the last two are seven letter palindromes. Explain why `precipice` matches this regular expression and how to fix the regex so that only palindromes match it.

- analyticity
- interpret
- precipice
- recognizing
- reinterpret
- reviver
- rotator
Part F (4 points)
Recall that a start codon is \textit{ATG} and that a stop codon is any of the three \textit{TAG}, \textit{TGA}, or \textit{TAA}. The Java code below is an attempt to find start/stop codon pairs in a strand of DNA. A run is shown for the strand indicated in the program.

Here’s the code

```java
import java.util.regex.*;

public class Restrict {
    static String dna = "ATGxxxTAG...ATGyyyyzzzTGA...ATGwwwwTAA...ATGaaaaaaa";
    // 01234567890123456789012345678901234567890123

    public static void main(String[] args) {
        Pattern starter = Pattern.compile("(ATG).*?(TAG|TGA|TAA)" );
        Matcher match = starter.matcher(dna);
        while (match.find()) {
            System.out.println(match.start() + " "+match.end());
            System.out.println(match.group());
            System.out.println("---");
        }
    }
}
```

When this code is run it generates the three matches shown below on the left. However, when the regular expression is changed so that the question mark is removed, that is it becomes "(ATG).*(TAG|TGA|TAA)" , then the output generated shows only one match as displayed on the right below (this is a separate run of the code).

<table>
<thead>
<tr>
<th>Run/executed with ? in regex</th>
<th>Run/executed without ? in regex</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 9 ATGxxxTAG</td>
<td>0 38 ATGxxxTAG...ATGyyyyzzzTGA...ATGwwwwTAA</td>
</tr>
<tr>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>12 25 ATGyyyyzzzTGA</td>
<td></td>
</tr>
<tr>
<td>---</td>
<td></td>
</tr>
<tr>
<td>28 38 ATGwwwwTAA</td>
<td></td>
</tr>
<tr>
<td>---</td>
<td></td>
</tr>
</tbody>
</table>

Provide an explanation of the different behavior based on your knowledge of regular expressions and your ability to reason. We did not discuss the question mark as part of regular expressions. When it’s used, the regular expression matches are called \textit{reluctant}. When the question mark is not used, a match is called \textit{greedy}. These terms may help in your explanation.