Approximate Regular Expressions: A Comparison of Exact and Approximate Matching Algorithms

Umme Salma Gadriwala, Tasnim Noshin, Rumsha Siddiqui

{gadriway, noshint, siddiqur}@mcmaster.ca

Department of Computing and Software, McMaster University
1280 Main St. W, Hamilton, Ontario, Canada L8S 4L8

April 2019

Motivation

Applications of finding exact and approximate string matches range far and wide. In browsing, the find function is often employed to locate exact text snippets on a screen. In computer security, approximate matching is necessary to find computer viruses and spam signatures that resemble certain patterns. In bioinformatics, approximate matching is vital for comparing and contrasting DNA and protein sequences [1]. The length of sequences involved in these applications reach billions. Thus, efficient string matching algorithms are needed to preserve time and cost of resources.

Problem

Different applications have different constraints on how strict of a match must be found. Some applications require both exact and approximate matching features. This project explores whether or not an implementation of exact matching can be replaced with one of approximate matching, where the error value for an exact match request would be set to zero. We aim to determine how an approximate matching algorithm fares in efficiency when compared to an exact matching algorithm.

Solution

A Python implementation of Thompson's exact matching algorithm is compared with that of Myers and Miller's approximate matching algorithm. Matches between string and regular expressions of various legnths are used for testing. The testing methodology compares the time each algorithm takes for the same samples.

Background Study

Exact Matching: Thompson's Algorithm: This exact matcing algorithm generates a non-deterministic finite automata (NFA) from a given regular expression. The construction is O(m). The number states and transitions is O(m) as well, where m is the number of symbols [1].

Approximate Matching: Myers and Miller's Algorithm: An approximate matching algorithms aims to find a match between a regular expression and a string with up to k errors. These k errors can consist of an insertion or deletion (a gap in the alignment of the regular expression and string), or a substitution (a difference in characters at a particular alignment position). The NFAs constructed in this approximate matching method are unique for each string and regular expression combination. The time complexity of Myers and Miller's is $O(mn/k + m2^k + (n + m) \log m)$, where m and n are the lengths of the regular expession and string respectively. The space complexity is $O(2^k m)$, where $k <= w$, the word size [1].

Figure 1: The NFA on the left is constructed by Thompson's algorithm based on regular expression (a or b) a*. The NFA on the right is generated by Myers and Miller's algorithm based on the same regular expression, but specifically with string 'ba'

Methodology

```
# Sample: compute runtime of exact matching algorithm
for i in  range(numLength):
    # Extend the string
    string = string + "ab"*4
    start = time.time()
    for i in  range(numIterations):
        exact_nfa.match(string)
    end = time.time()
    # Compute average time to output
    exactTimes = exactTimes + [(end-start)/numIterations]
```

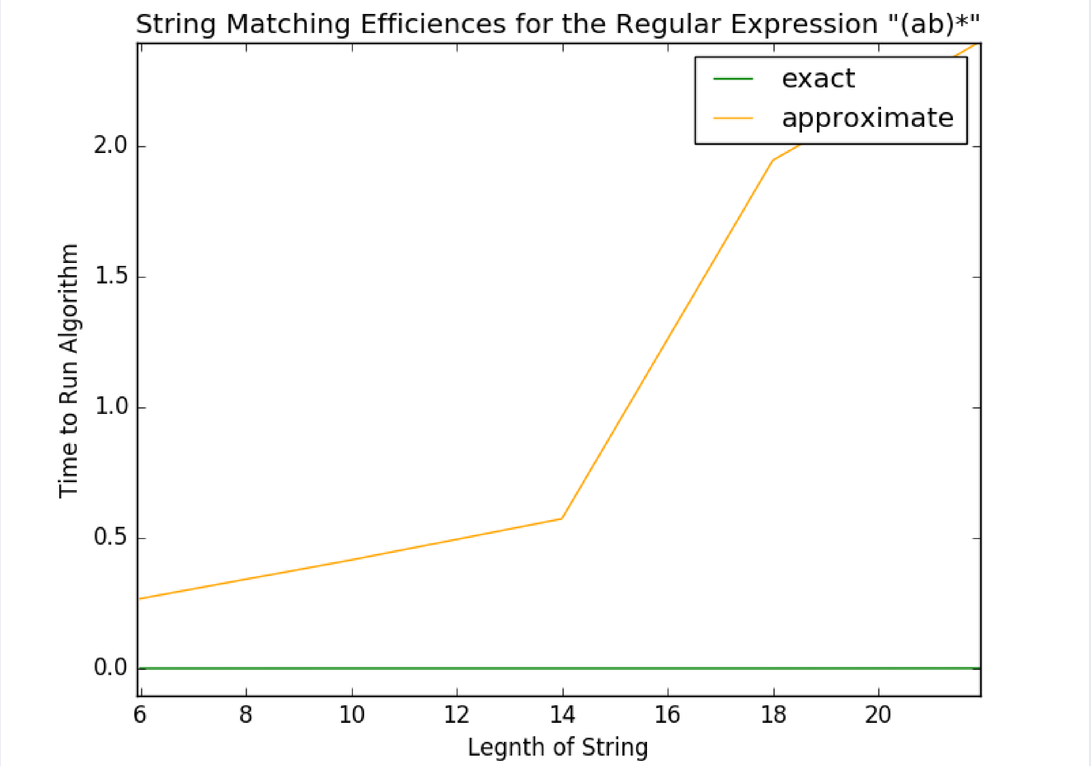

Figure 2: The time required to match strings of repeating sequences of 'ab' to the regular expression '(ab)*', as the legnth of the string increases.

Conclusions & Future Work

Based on the times computed through the experiment, it is evident that the approximate matching algorithm takes a significant amount of time. It is not worth the cost to use approximate matching for the case where $k = 0$. It is recommended to implement an exact matching algorithm to perform tasks that require exact matching.
In this experiment, only two algorithms were explored. Future research would benefit from exploring the efficiency of a plethora of different methods.

Acknowledgements

We thank Dr. Sekerinski for inspiring this project, as well as his continued guidance in our learning and endeavours. We also thank Spencer Park and Erin Varey for their invaluable feedback and support.

References

[1] D. Belazzougui and M. Raffinot, "Approximate regular expression matching with multi-strings," *Journal of Discrete Algorithms*, vol. 18, pp. 14–21, January 2013.