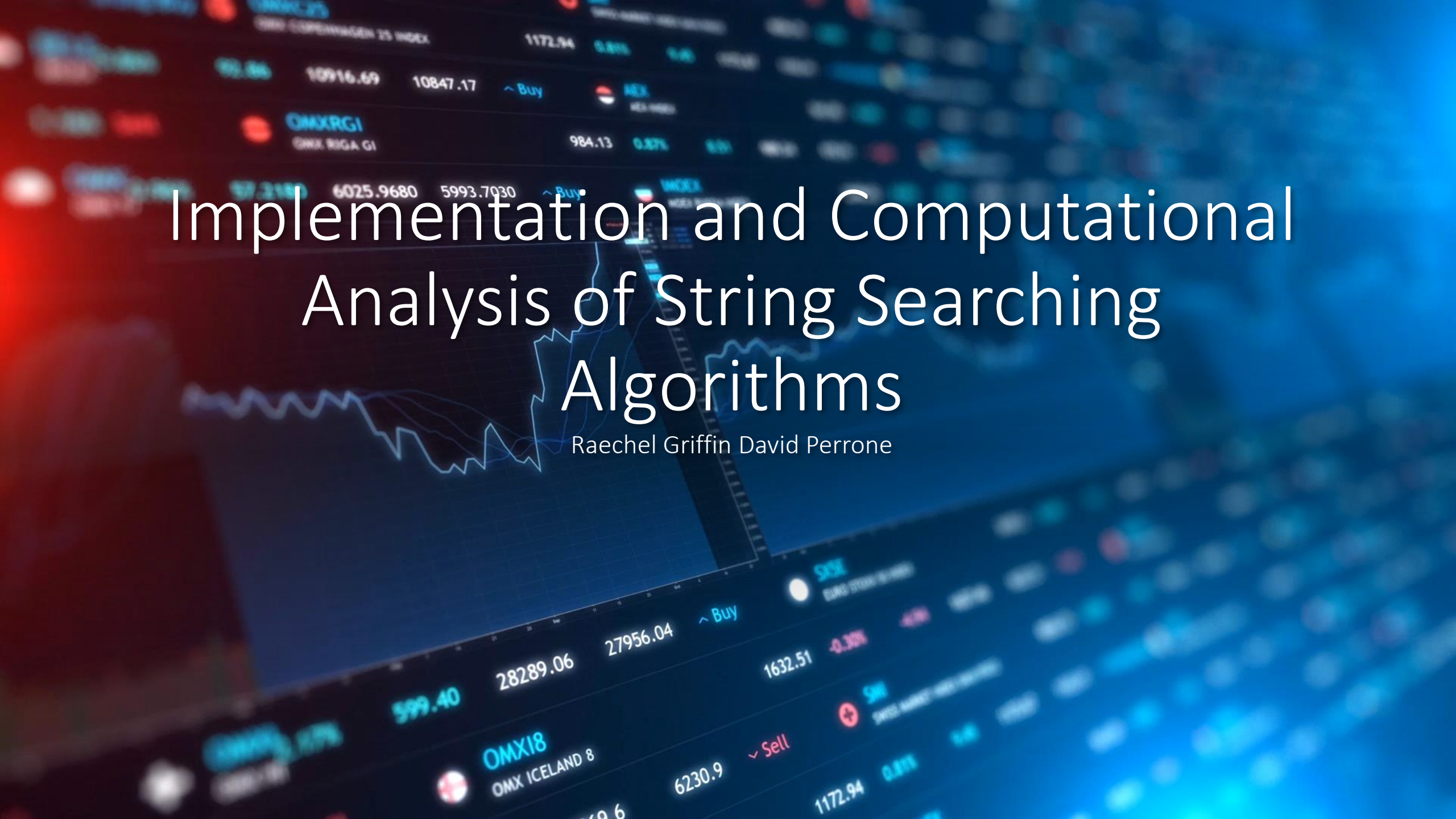


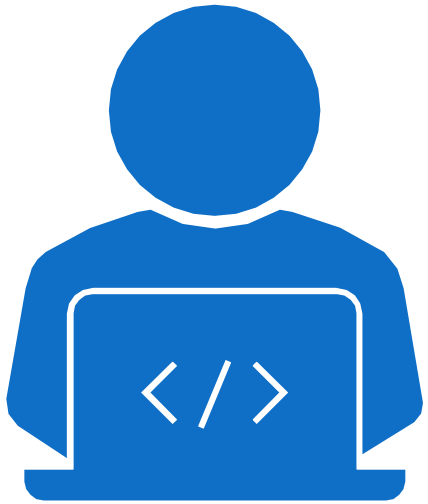
Implementation and Computational Analysis of String Searching Algorithms

Raechel Griffin David Perrone

What is String Searching

- A method of searching for a pattern in a larger text.
- Applications:
  - Most text editors will have a built-in search feature
  - Computer forensics – search for digital signatures
  - Filter spam from emails by searching for keywords
  - Bioinformatics and gene sequencing

A Brute Force Approach

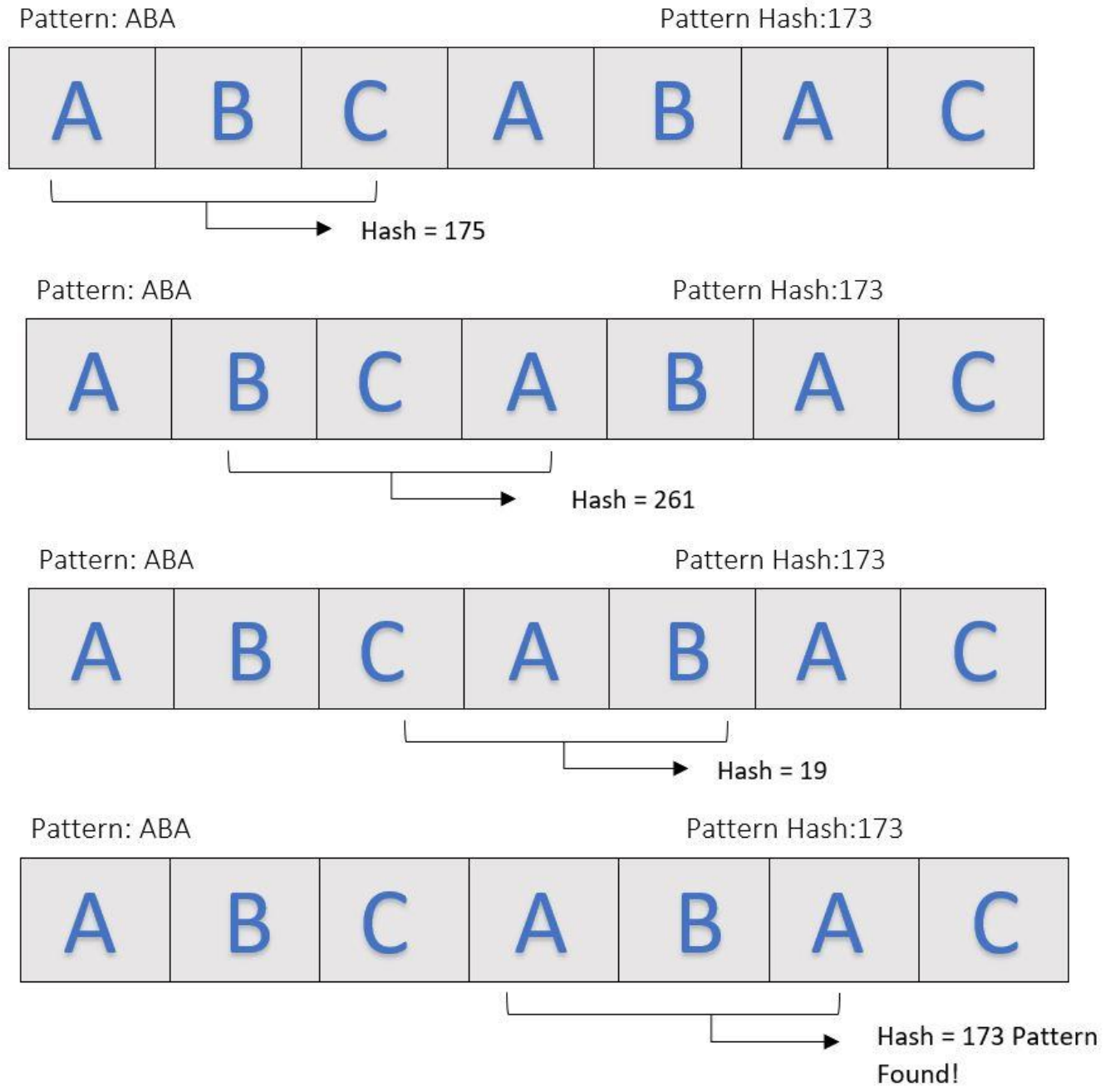


- Check every character in a string between indices 0 and n-m
- Worst-case runtime - O(nm).
- Downfalls:
  - Always shifts the comparison window 1 position to the right
  - False-Positives results in extra work done
  - Maybe there is a better way to do this!

Rabin Karp: How it Works

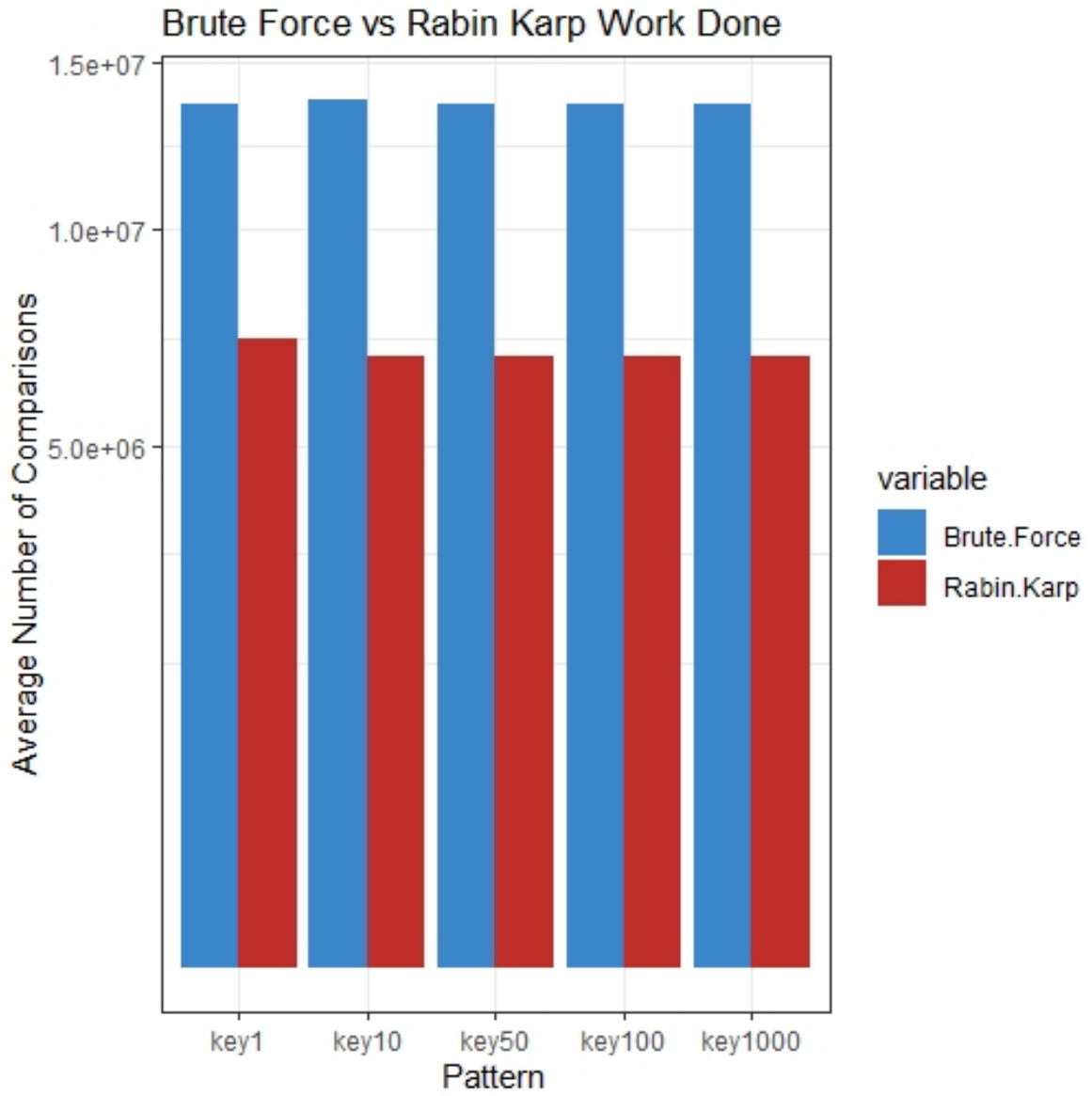
Reduce	Do	Minimize
Reduce comparisons by converting strings into hash values	Do the strings look similar? If so, check.	A good hashing function will minimize the number of collisions which reduces work done

Rabin Karp: How it Works



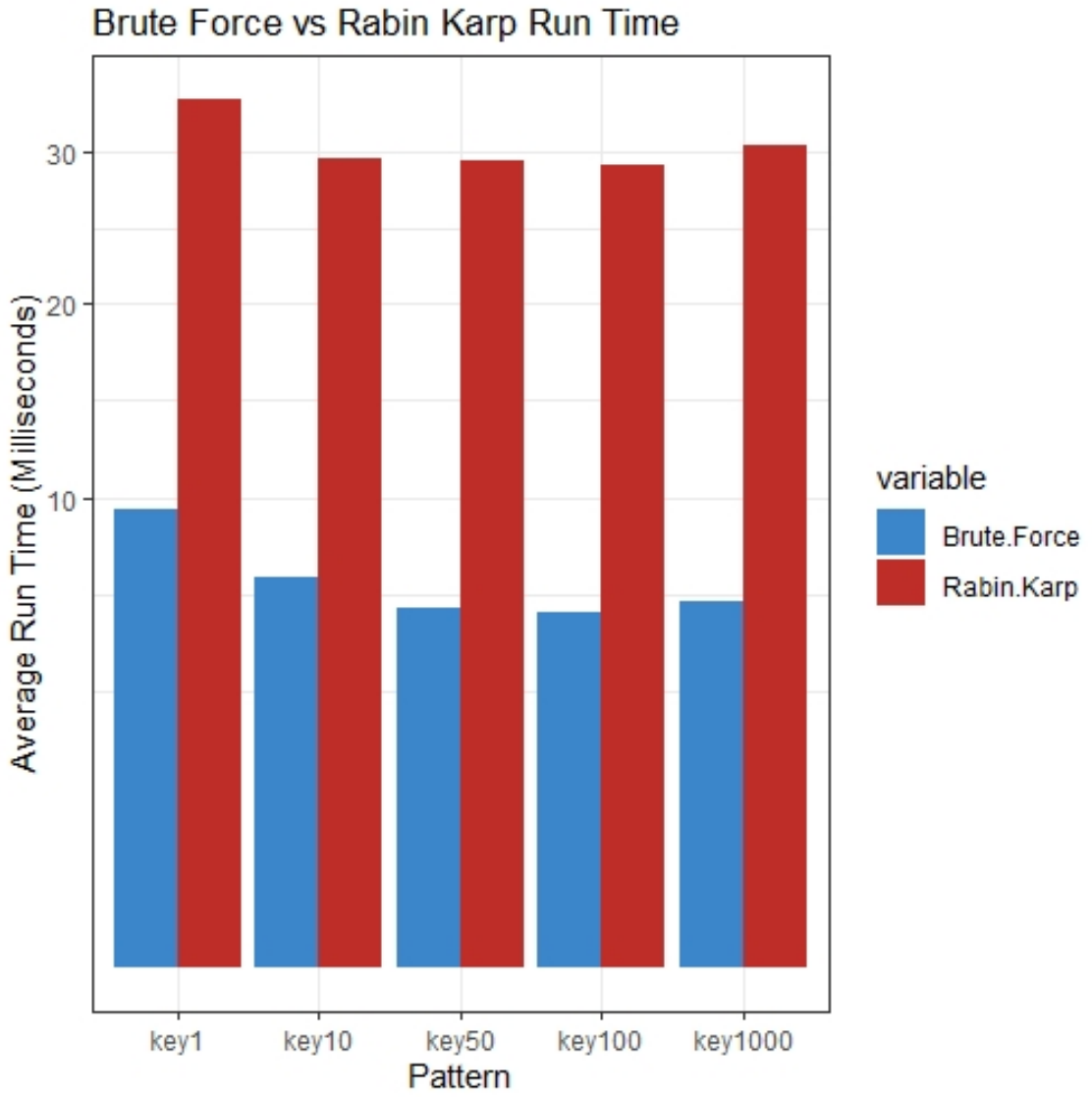
Rabin Karp: A Good Algorithm?

On average Rabin Karp makes 49.43% less comparisons than brute force!



Rabin Karp: A Good Algorithm?

- Best-case runtime of O(n+m)
  - No collisions/matches
- Worst-case runtime of O(nm)
  - Collision/match at every position

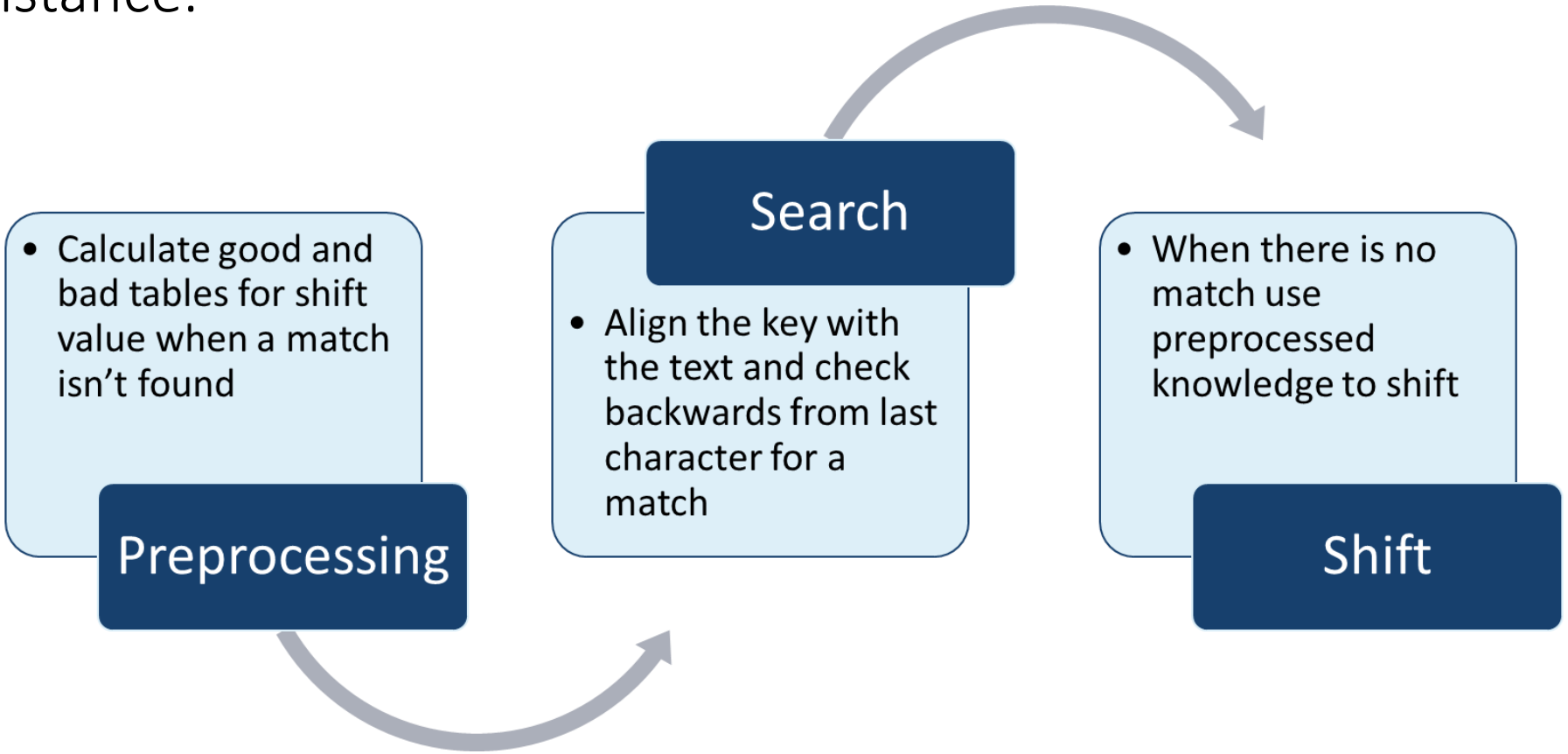


Rabin Karp: Computational Costs

- Rolling the hash in Theory:
  - 123…4 → 234
- In Practice:
  - $h = ((123 + q - \text{‘1’} * b^{m-1} \% q) * b + \text{‘4’}) \% q$
- “Even though the Karp-Rabin algorithm needs fewer symbol comparisons … the cost of computing the hashing function outweighs the advantage of performing fewer symbol comparisons.” -Christian Lovis, MD, Robert H. Baud, PhD

Boyer Moore: How it Works

The Boyer-Moore algorithm aims to be more efficient by maximizing the shift distance.



Boyer Moore: How it WORKS

W	O	R	K	S
4	3	2	1	0

For the bad shift we used a map to represent the table. If the mismatched character from the text does not appear in the pattern, the window can be safely shifted to the right by the length of the pattern. If a character appears more than once, the rightmost occurrence is used.

The good shift rule requires two tables, each represented as a vector. This rule is quite complicated and comes into play when the key has repeated substrings like “abababababab”

To calculate the final shift value, take the max shift of all tables.

Boyer Moore: An Example

Bad Character Table

E	X	A	M	P	L
0	5	4	3	2	1

Good Character Table

E	X	A	M	P	L
1	1	1	1	1	1

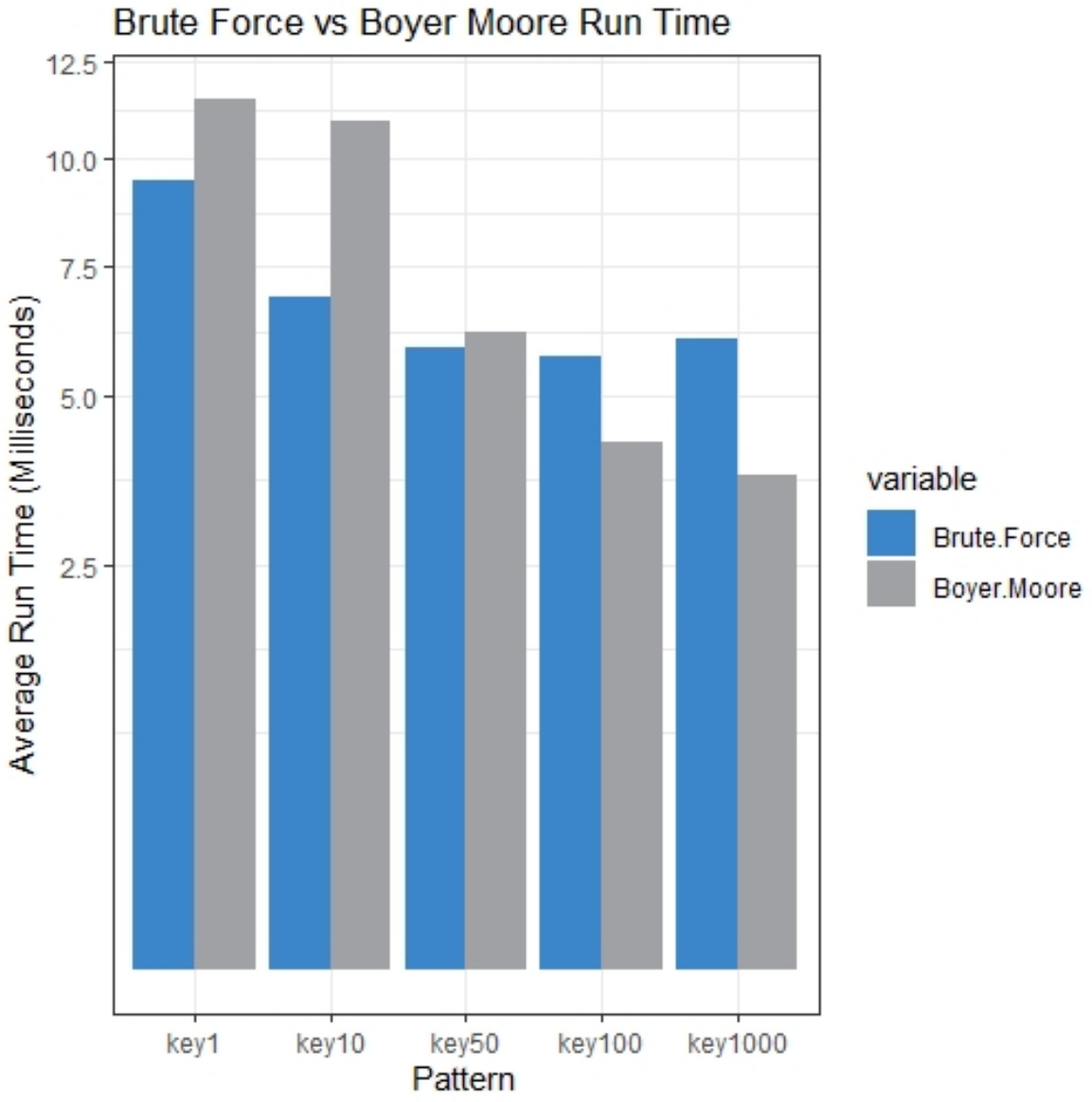
T	H	I	S		I	S		A	N		E	X	A	M	P	L	E

E	X	A	M	P	L	E

Pattern Found!!

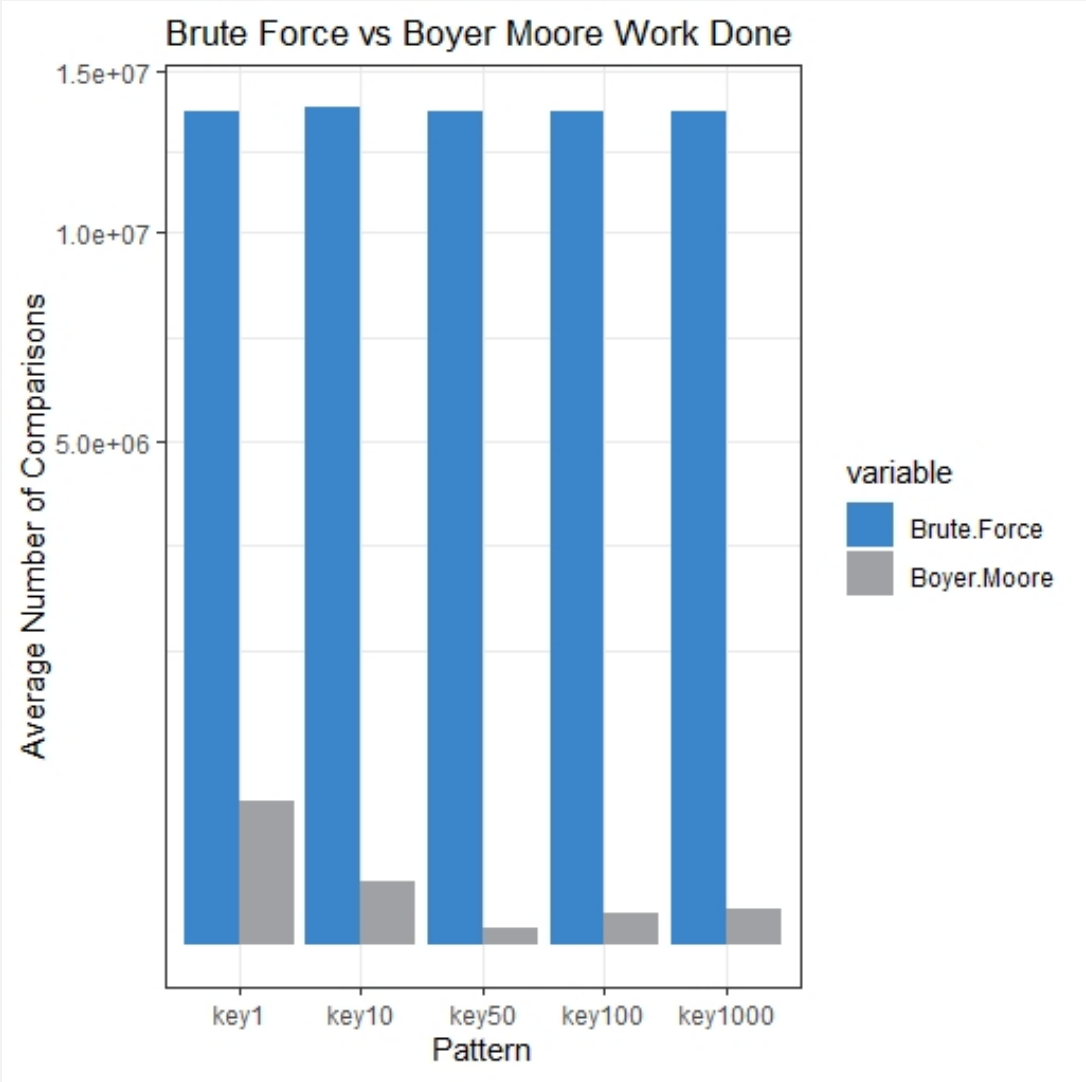
Boyer-Moore: A good Algorithm?

- On average Boyer-Moore runs 6.95% longer than brute force.
- But for longer patterns like 100 and 1000 characters, Boyer-Moore runs 63% faster than brute force
- Worst Case O(m) preprocessing and O(mn) matching
- Best Case O(m) preprocessing and O(m/n) matching

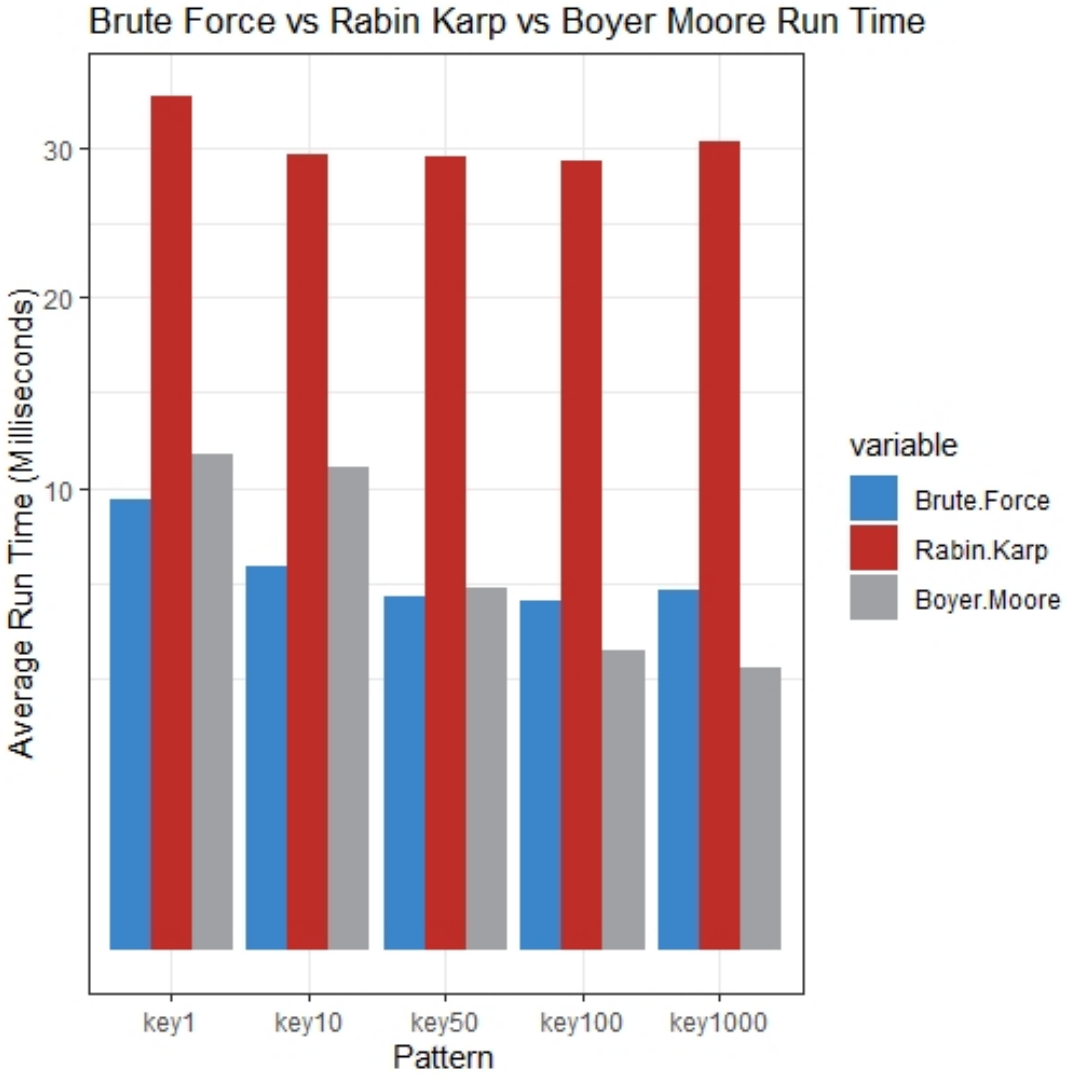


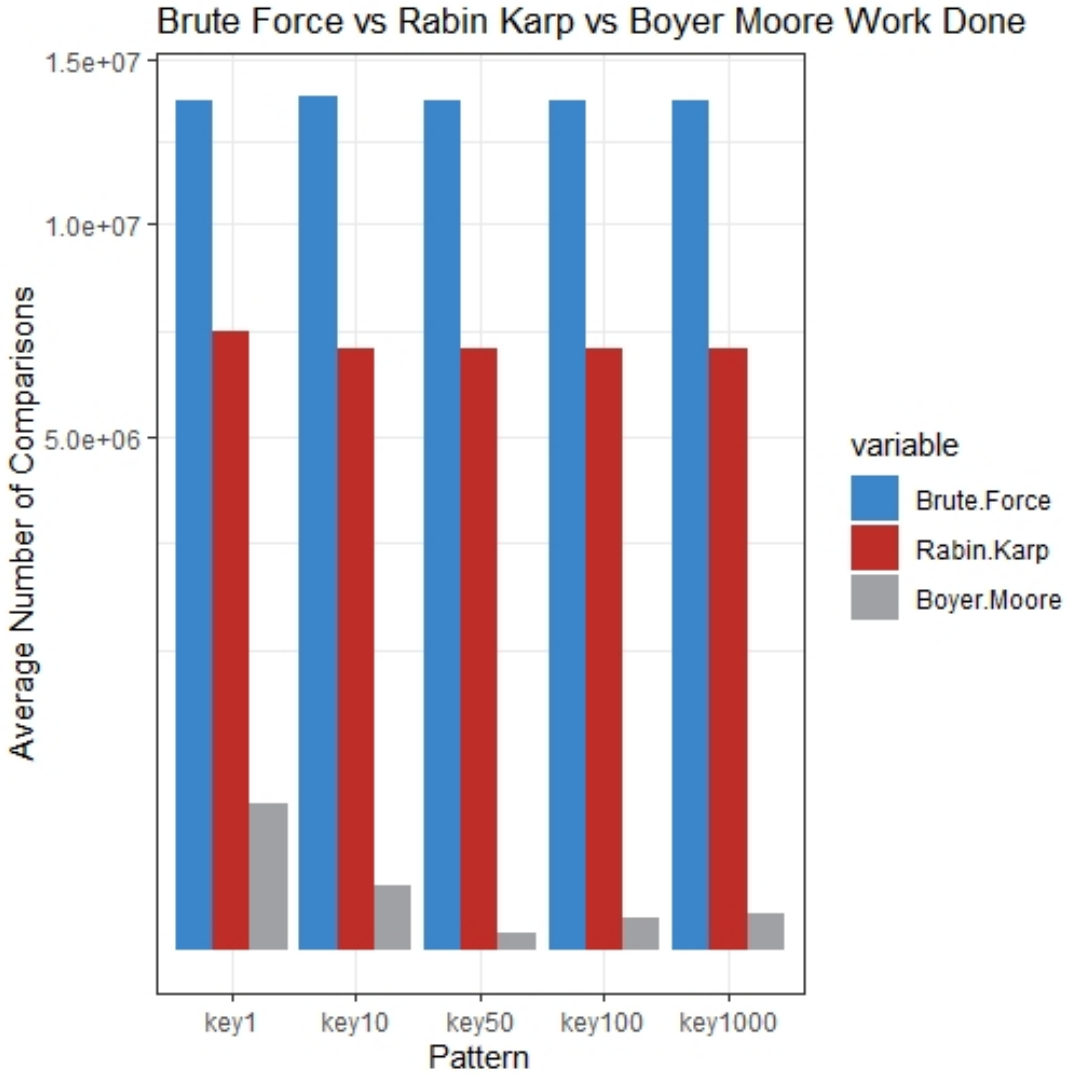
Boyer-Moore: A good Algorithm?

On average Boyer-Moore makes 99.22% less comparisons than brute force!



Putting it all together…



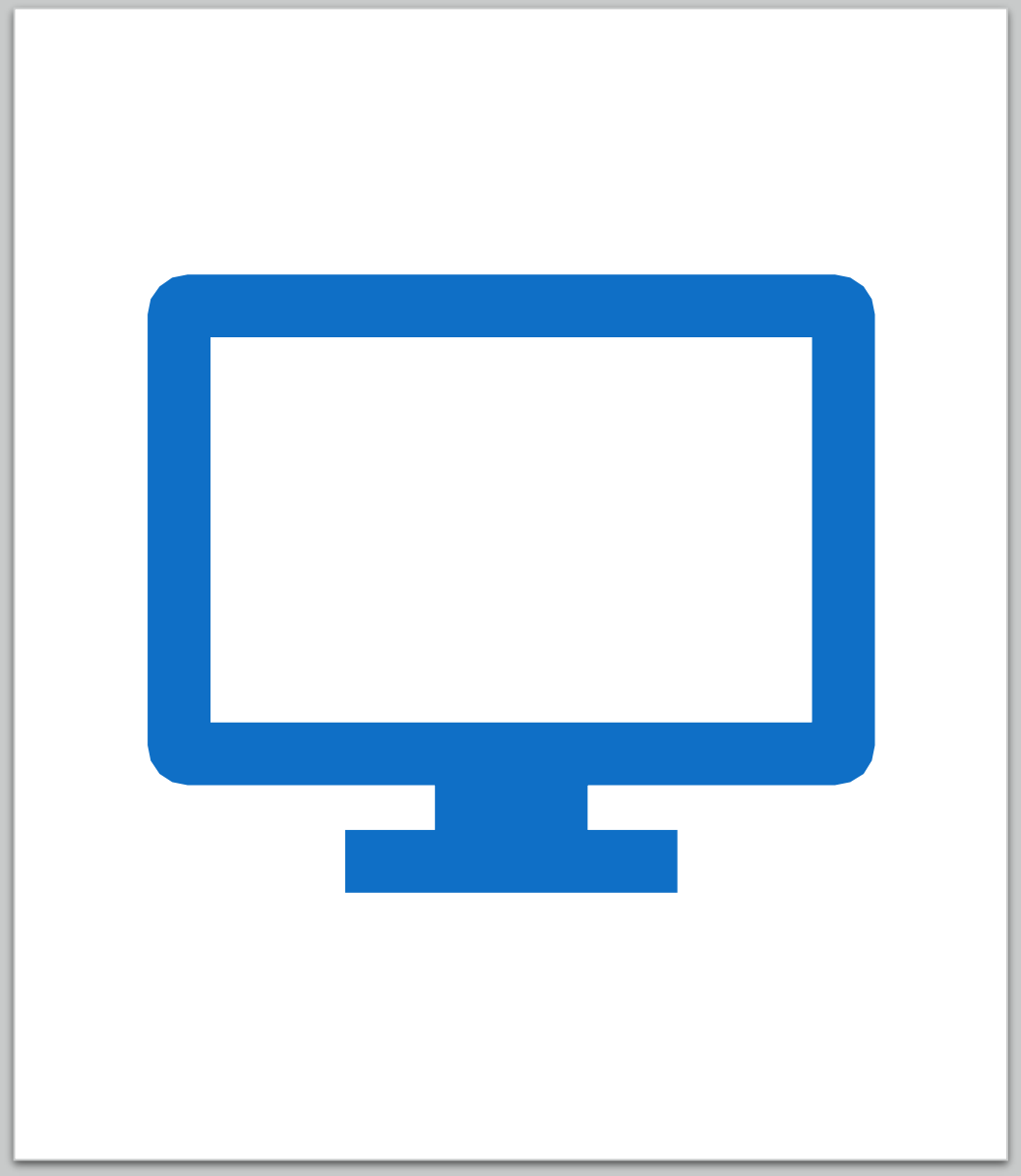


Putting it all together…

- Boyer-Moore is the most efficient algorithm minimizing the work done by a naïve brute force algorithm by nearly 100%
- Its computational cost doesn’t greatly worsen the run times
- Rabin Karp does less work than brute force, but the computational costs of the hash function worsen runtimes

Live Demo

Let’s see how our code works when we search out test files with Rabin Karp and Boyer Moore!



Questions?

https://github.com/djperrone/212final.git