Rabin-Karp for string matching

...

Endre Valen, Martin Toft, Sergey Jakobsen
endrelv@ifi.uio.no, matoft@ifi.uio.no, sergeyj@ifi.uio.no

History and application

Michael O. Rabin and Richard M. Karp (1987)

Example application: Detecting plagiarism

String matching

$T \leftarrow$ Text

$P \leftarrow$ Pattern

Problem: $P \in T$?

Naive algorithm

$i, j \leftarrow 0, 0$

1. Compare T[i] with P[j]
2. If match: increase i and j
    2.1. If j = len(P): $P \in T$ ✔
3. If not match: increase i, $j \leftarrow 0$
4. If i = len(T): $P \notin T$ ❌
5. Repeat step 1

Rabin-Karp

What makes Rabin Karp special? Rolling hash!

1. Hash P
2. Hash |P| first characters of T
3. After this, we can find the next hash values in T in constant time by applying Horner's method!

Importance of hash function

We want to limit the amount of collisions

The hash function should map every unique input to a unique value

Easier said than done in practice

Demonstration

Assume we have -

- a rolling hash function: hash(string) → Int
- a pattern: “ABA”
- a text: “AABABCABACABBA”

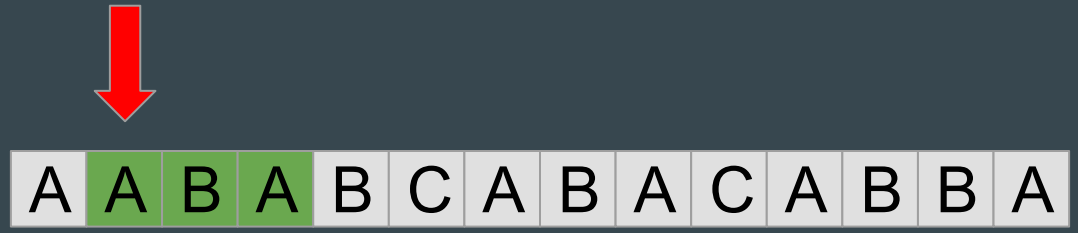
hash(“ABA”) → 33

hash(“AAB”) → 17

hash(“ABA”) → 33

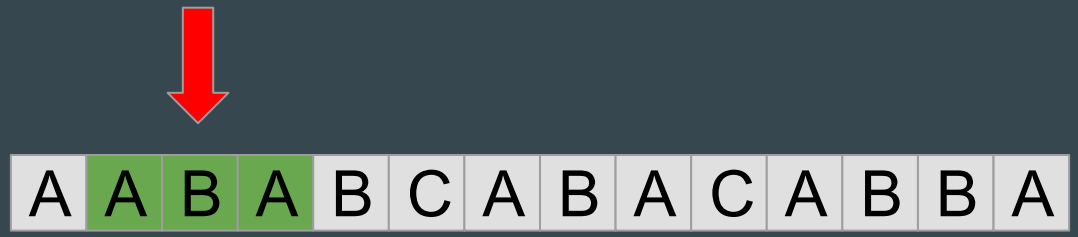
hash(“ABA”) → 33

hash(“ABA”) → 33



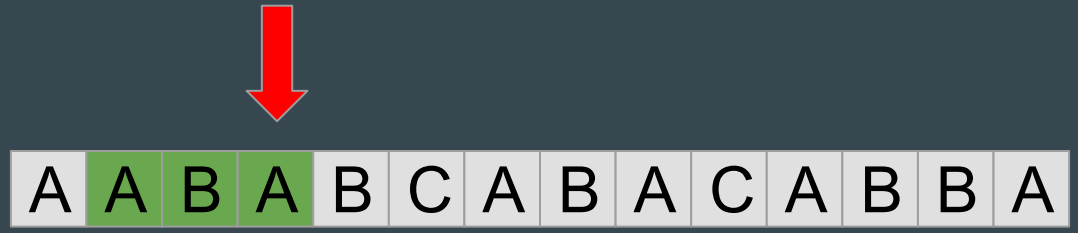
hash(“ABA”) → 33

hash(“ABA”) → 33



hash(“ABA”) → 33

hash(“ABA”) → 33



hash(“ABA”) → 33

hash(“ABA”) → 33

hash(“ABA”) → 33

hash(“ABA”) → 33

hash(“BAB”) → 45

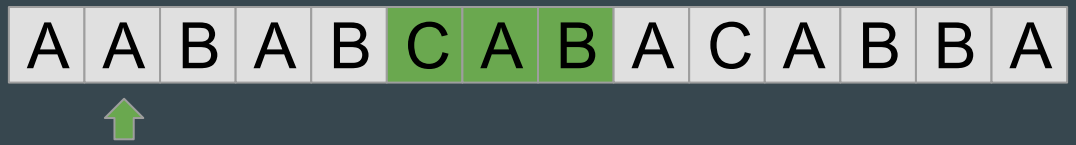
hash(“ABA”) → 33

hash(“ABC”) → 28

hash(“ABA”) → 33

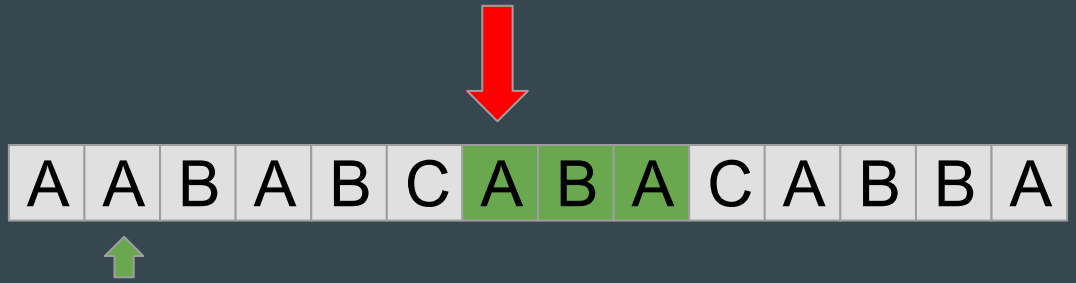
hash(“BCA”) → 13

hash(“ABA”) → 33



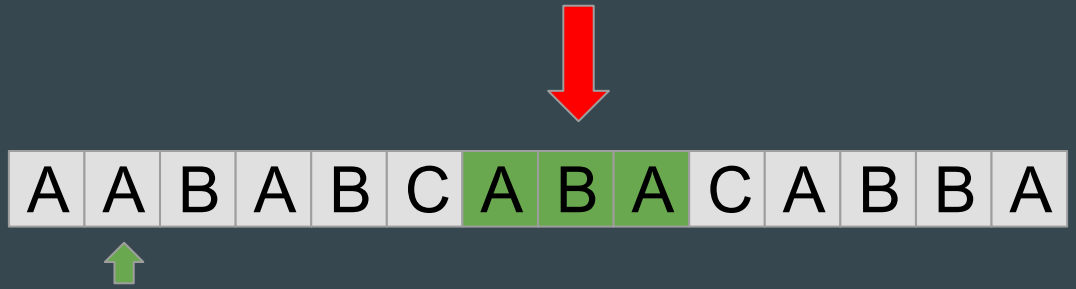
hash(“CAB”) → 17

hash(“ABA”) → 33



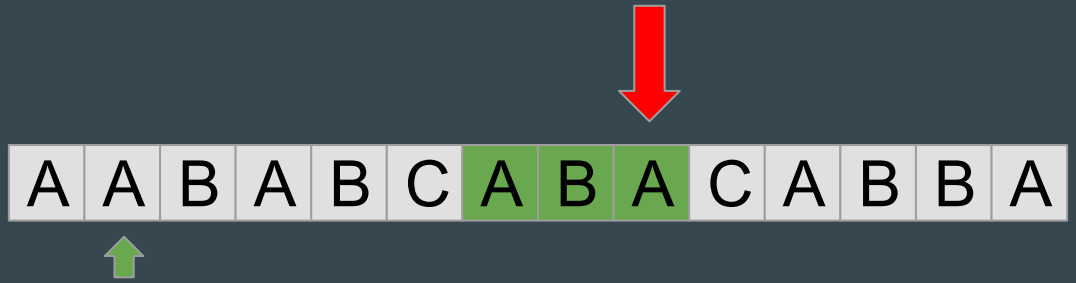
hash(“ABA”) → 33

hash(“ABA”) → 33



hash(“ABA”) → 33

hash(“ABA”) → 33



hash(“ABA”) → 33

hash(“ABA”) → 33

hash(“ABA”) → 33

hash(“ABA”) → 33

hash(“BAC”) → 51

hash(“ABA”) → 33

hash(“ACA”) → 43

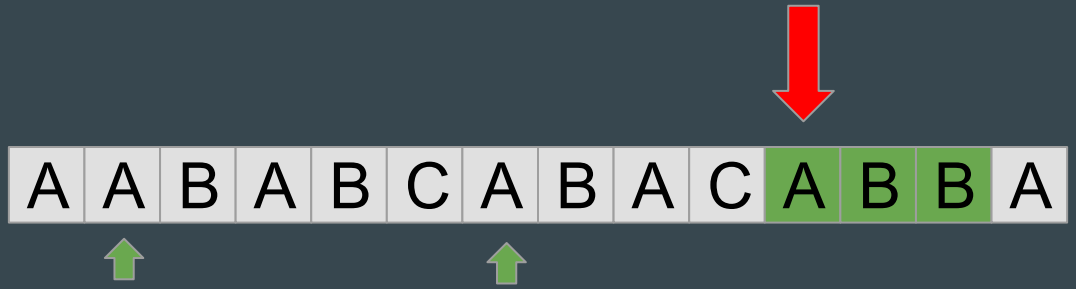
hash(“ABA”) → 33

hash(“CAB”) → 17

hash(“ABA”) → 33

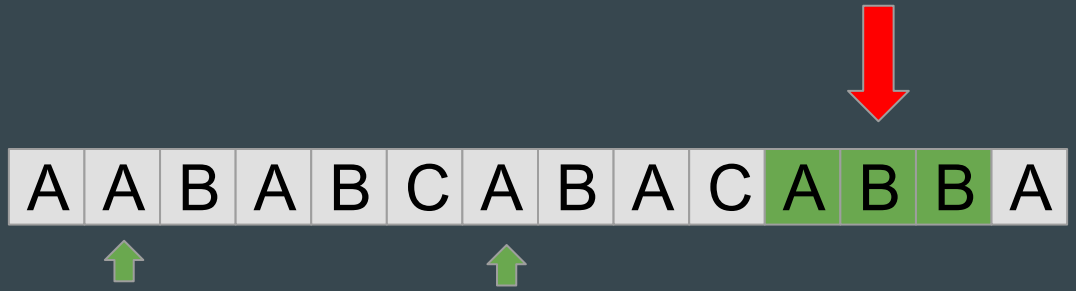
hash(“ABB”) → 33

hash(“ABA”) → 33



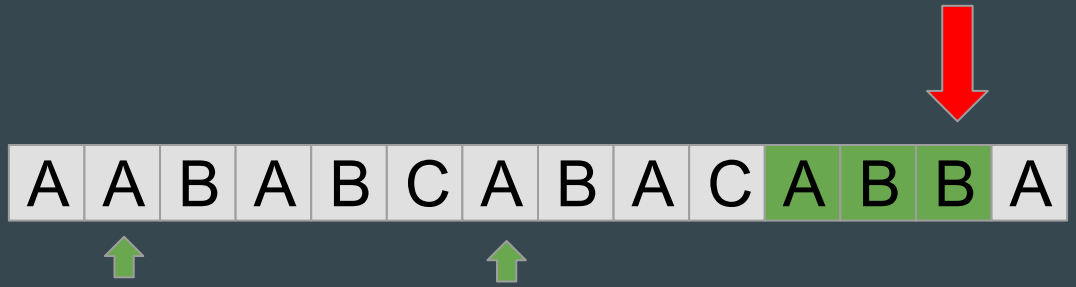
hash(“ABB”) → 33

hash(“ABA”) → 33



hash(“ABB”) → 33

hash(“ABA”) → 33



hash(“ABB”) → 33

hash(“ABA”) → 33

hash(“BBA”) → 47

hash(“ABA”) → 33

Complexity and advantages/disadvantages

Average/best case: $O(n)$

Worst case: $O(mn) = O(m) + O(m) * O(n)$

Hash collisions and false positives!

Thanks for watching!

Sources:

- https://en.wikipedia.org/wiki/Rabin%E2%80%93Karp_algorithm
- https://www.uio.no/studier/emner/matnat/ifi/IN3130/h21/slides/forelesning-3---string-search-pk.pdf