

MA/CSSE 473

Day 28

Boyer-Moore

Hashing



MA/CSSE 473 Day 28

- HW 11 due now
- HW 12 due Oct 30 (available later this morning)
- Exam 2 is Friday, Oct 31
- **Student Questions**
- Boyer-Moore String search algorithm
- Hashing: Review of algorithms and analysis
- B-Trees – a quick look



Recap: Boyer-Moore

- Boyer-Moore takes into account k , the number of matched characters (from the right) before a mismatch occurs.
- If $k=0$, we simply do the same shift as Horspool's algorithm.



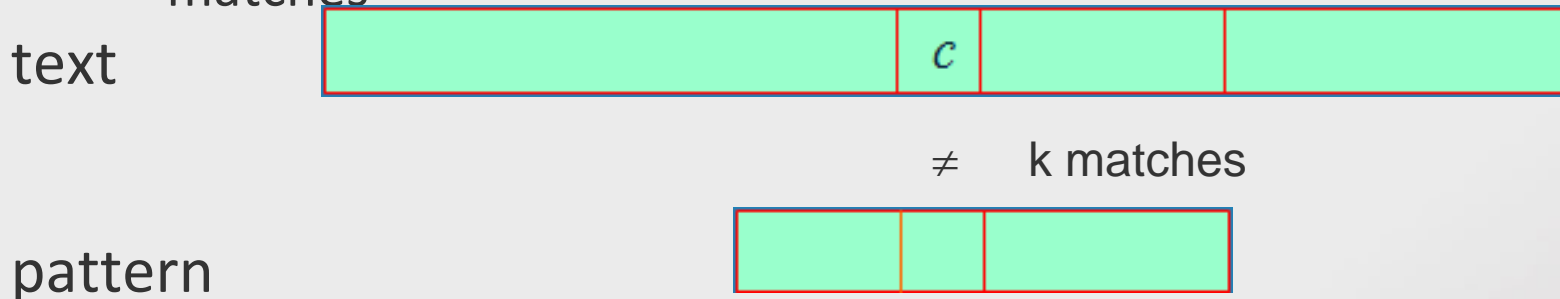
Boyer-Moore Algorithm

- Based on same two main ideas:
- compare pattern characters to text characters from right to left
- precompute the shift amounts in **two** tables
 - **bad-symbol table** indicates how much to shift based on the text's character that causes a mismatch
 - **good-suffix table** indicates how much to shift based on matched part (suffix) of the pattern



Bad-symbol shift in Boyer-Moore

- If the rightmost character of the pattern does not match, Boyer-Moore algorithm acts much like Horspool's
- If the rightmost character of the pattern does match, BM compares preceding characters right to left until either
 - all pattern's characters match, or
 - a mismatch on text's character c is encountered after $k > 0$ matches



bad-symbol shift: How much should we shift by?

$$d_1 = \max\{t_1(c) - k, 1\},$$

where $t_1(c)$ is the value from the Horspool shift table.



Good-suffix Shift in Boyer-Moore

- Good-suffix shift d_2 is applied after the k last characters of the pattern are matched
 - $0 < k < m$
- How can we take advantage of this?
- As in the bad suffix table, we want to precompute some information based on the characters in the suffix.
- We want to create a **good suffix table** whose indices are $k = 1 \dots m-1$, and whose values are how far we can shift after matching a k -character suffix (from the right).
- Spend 5 minutes talking with one or two other students. Try to come up with criteria for how far we can shift.
- Example patterns: CABABA AWOWWOW
 WOWWOW ABRACADABRA



Boyer-Moore Algorithm

After matching successfully $0 < k < m$ characters, the algorithm shifts the pattern right by

$$d = \max \{d_1, d_2\}$$

where $d_1 = \max\{t_1(c) - k, 1\}$ is bad-symbol shift

$d_2(k)$ is good-suffix shift



Boyer-Moore Example

pattern = abracadabra

text =
abracadabtabradabracadabcadaxbrabbracadabraxxxxxxabracadabracadabra

m = 11, n = 67

badCharacterTable: a3 b2 r1 a3 c6 x11

GoodSuffixTable: (1,3) (2,10) (3,10) (4,7) (5,7) (6,7) (7,7) (8,7)
(9,7) (10,7)

abracadabtabradabracadabcadaxbrabbracadabraxxxxxxabracadabracadabra
abracadabra

i = 10 k = 1 t1 = 11 d1 = 10 d2 = 3

abracadabtabradabracadabcadaxbrabbracadabraxxxxxxabracadabracadabra
abracadabra

i = 20 k = 1 t1 = 6 d1 = 5 d2 = 3

abracadabtabradabracadabcadaxbrabbracadabraxxxxxxabracadabracadabra
abracadabra

i = 25 k = 1 t1 = 6 d1 = 5 d2 = 3

abracadabtabradabracadabcadaxbrabbracadabraxxxxxxabracadabracadabra
abracadabra

i = 30 k = 0 t1 = 1 d1 = 1



On-line Boyer-Moore Example

- On Moore's home page
- <http://www.cs.utexas.edu/users/moore/best-ideas/string-searching/fstrpos-example.html>



Implementation project

- Produce an interactive animation to help students learn how Boyer-Moore works.
- Include construction of the two tables and the actual search of the text.
- You may work alone or with up to 3 other people.
- Do it and demonstrate it to me by noon on Thursday of Finals week.
- Ideally, it should run in a web browser.
- One feature should be the ability of the student to choose the pattern and text, but you should also supply some patterns and texts that illustrate various aspects of the algorithm.



Hashing Review

- What problem do we try to solve by hashing?
- What is the brute force approach?
- What alternatives have we seen?
- What is the general idea of how hashing works?
- Why does it fit into Chapter 7?
- What are the main issues to be addressed when discussing hashing implementation?



Terminology and analysis

- collision
- load factor (λ)
- separate chaining
- open addressing
- linear probing
- cluster
- quadratic probing
- rehashing
- expected lookup time (as a function of λ)
 - For successful search
 - For unsuccessful search

