Lazaron Shyta

Professor Emil Kelevejdjiev

COS460 - Algorithms

Topic: Implement step-by-step illustration of Boyer-Moore method of pattern matching.

String pattern matching / **Boyer-Moore Algorithm**

*Typical problem: Given a **text** find all the occurrences of a **pattern.***

Pattern searching is one of the most important problems in computer science. There are many Pattern Searching algorithms to use in finding a pattern inside a text. (KMP, Finite State Automata, Robin-Karp, Boyer-Moore etc.)

Some of the algorithms, use some preprocessed tables to help with the search of the patterns. Boyer-Moore is one of them.

Boyer-Moore Algorithm

The reason we have different algorithms other than the naïve approach is that we want to shift the pattern by more than one position.

Boyer-Moore can be a combination of two approaches:

- Bad Character Heuristic **(I will focus here)**
- Good Suffix Heuristic

Bad Char Heuristic

Bad character → The current character of the text mismatching with the current character of the pattern.

In case of a mismatch → Shift the pattern until…

1. Mismatch becomes a match.
2. Patten moves past the mismatched char.

Two cases of mismatch…

a) **Mismatch becomes a match** -→ We will check the position of the last position of the Bad Character in pattern, and if the character exists in the pattern, we will shift the pattern in order to fit with the Bad Character in Text.

Example

T: *alibdasgf*
P: *bbbf*
> *Shift to index 1* **(Bad char 'b' exists in the pattern)**

```
alibdasgf
 bbbf
```

b) **Pattern moves past the mismatched char** → If the Bad Character doesn't exist in the pattern, then we shift the all the pattern after the Bad Character (mismatched char).

Example *continuing the previous example*

T: *alibdasgf*
P: *bbbf*

> *Shift to index 5* **(Bad char 'd' does not exist in the pattern)**

```
alibdasgf
     bbbf
```

It is highly effective → In English text an average of $O(n/m)$ complexity.

Worst Case

The Bad Character Heuristic's time complexity can go up to $O(mn)$ in worst case. This would occur when all characters of the text and pattern are the same.

For example:

text = "111111111111111" *and* **pattern**="111"

Lazaron Shyta

Implementation in C++ (Algorithm only)

```
unordered_map<char,int> lastPos; // Table for storing rightmost positions


void prepTable(string pattern, string text){
    int it = 0;
    // Store last position of every character.
    for( char ch : text )
        lastPos[ch] = -1;
    for( char ch : pattern )
        lastPos[ch] = it++;
}
void searchBM(string text, string pattern){
    int n = text.length();
    int m = pattern.length();
    prepTable(pattern,text); // Prepare the table which helps shifting.
    int shift = 0, remain = n - m;
    while( shift <= remain ){
        int check = m - 1;
        while( check >= 0 && pattern[check] == text[ shift + check ])
                        check--; // Decrement [check] until a mismatch.
        if(check < 0){  // If check < 0 then we have a match.
            if( shift + m < n ) shift += m - lastPos[ text [ shift + m ] ];
                  // Shift by using the Bad Character heuristic
            else shift++;
        }
        else{  // Shift by using the Bad Character heuristic
            shift += max(1, check - lastPos[ text [ shift + check ] ]);
        }
    }
}
```

My Program

Program search a pattern in a text and find all the occurrences. It consists of a single **.cpp** file. *BM-code.cpp*

- Input: *Text* and *Pattern*
- Output: Two .txt files
  - ✓ *steps.txt*
  - ✓ *Illustrator.txt*

```
> Enter the text: sampddpd
> Enter the pattern: pd
* Files 'steps.txt' and 'illustrator.txt' created! *
```

Figure 1 Prompting user

steps.txt

This file will be created after the successful execution of *BM-code.cpp.*

It will contain all the steps performed by the Boyer Moore algorithm.

Example:

```
 + Last position table...
                          |   d   |   p   |
                          |   1   |   0   |
Steps below
     ...

 > Shift to index 2
sampddpd
  pd

 > Shift to index 3
sampddpd
   pd

         + Match at index 3
        sampddpd
           pd

 > Shift to index 5
sampddpd
     pd

 > Shift to index 6
sampddpd
      pd

         + Match at index 6
        sampddpd
              pd

******* End of steps *******
```

Figure 2 Inside steps.txt

Illustrator.txt

This file will be created after the successful execution of *BM-code.cpp.*

It will contain some illustration of the pattern found in the text performed by Boyer Moore algorithm.

Example…

```
 -> Boyer-Moore Algorithm illustration <-

******* Program starts here *******

    Text: sampddpd
    Pattern: pd

- - - - - - - - - -
Pattern 'pd' occurs 2 time(s) in the text.
Position(s): 3 6


> Matching Position(s) below...

        sampddpd
           pd
              pd
- - - - - - - - - -
```

Figure 3 Inside illustrator.txt

Sources used.

1. Wikipedia
   (https://en.wikipedia.org/wiki/Boyer%E2%80%93Moore_string-search_algorithm).

2. Handouts given in class by Prof. Emil Kelevejdjiev.

3. Boyer-Moore visualization
   (https://people.ok.ubc.ca/ylucet/DS/BoyerMoore.html).

4. Geeks for Geeks article about Boyer Moore
   (https://www.geeksforgeeks.org/boyer-moore-algorithm-for-pattern-searching/)