Pattern Searching | Set 7 (Boyer Moore Algorithm – Bad Character Heuristic)

Given a text txt[0..n-1] and a pattern pat[0..m-1], write a function search(char pat[], char txt[]) that prints all occurrences of pat[] in txt[]. You may assume that n > m.

Examples:

```
1) Input:

  txt[] =  "THIS IS A TEST TEXT"
  pat[] = "TEST"

Output:
```

```
Pattern found at index 10

2) Input:

  txt[] =  "AABAACAADAABAAABAA"
  pat[] = "AABA"

Output:

   Pattern found at index 0
   Pattern found at index 9
   Pattern found at index 13
```

Pattern searching is an important problem in computer science. When we do search for a string in notepad/word file or browser or database, pattern searching algorithms are used to show the search results.

We have discussed the following algorithms in the previous posts:

Naive Algorithm
KMP Algorithm
Rabin Karp Algorithm
Finite Automata based Algorithm

In this post, we will discuss Boyer Moore pattern searching algorithm. Like KMP and Finite Automata algorithms, Boyer Moore algorithm also preprocesses the pattern.
Boyer Moore is a combination of following two approaches.
1) Bad Character Heuristic
2) Good Suffix Heuristic
Both of the above heuristics can also be used independently to search a pattern in a text. Let us first understand how two independent approaches work together in the Boyer Moore algorithm. If we take a look at the Naive algorithm, it slides the pattern over the text one by one. KMP algorithm does preprocessing over the pattern so that the pattern can be shifted by more than one. The Boyer Moore algorithm does preprocessing for the same reason. It preporcesses the pattern and creates different arrays for both heuristics. At every step, it slides the pattern by max of the slides suggested by the two heuristics. So it uses best of the two heuristics at every step. Unlike the previous pattern searching algorithms, Boyer Moore algorithm starts matching from the last character of the pattern.

In this post, we will discuss bad character heuristic, and discuss Good Suffix heuristic in the next post.

The idea of bad character heuristic is simple. The character of the text which doesn't match with the current character of pattern is called the Bad Character. Whenever a character doesn't match, we slide the pattern in such a way that aligns the bad character with the last occurrence of it in pattern. We preprocess the pattern and store the last occurrence of every possible character in an array of size equal to alphabet size. If the character is not present at all, then it may result in a shift by m (length of pattern). Therefore, the bad character heuristic takes O(n/m) time in the best case.

```
/* Program for Bad Character Heuristic of Boyer Moore String Matching Algorit

# include <limits.h>
# include <string.h>
# include <stdio.h>

# define NO_OF_CHARS 256
```

```
// A utility function to get maximum of two integers
int max (int a, int b) { return (a > b)? a: b; }

// The preprocessing function for Boyer Moore's bad character heuristic
void badCharHeuristic( char *str, int size, int badchar[NO_OF_CHARS])
{
    int i;

    // Initialize all occurrences as -1
    for (i = 0; i < NO_OF_CHARS; i++)
         badchar[i] = -1;

    // Fill the actual value of last occurrence of a character
    for (i = 0; i < size; i++)
         badchar[(int) str[i]] = i;
}

/* A pattern searching function that uses Bad Character Heuristic of
   Boyer Moore Algorithm */
void search( char *txt,  char *pat)
{
    int m = strlen(pat);
    int n = strlen(txt);

    int badchar[NO_OF_CHARS];

    /* Fill the bad character array by calling the preprocessing
       function badCharHeuristic() for given pattern */
    badCharHeuristic(pat, m, badchar);

    int s = 0;  // s is shift of the pattern with respect to text
    while(s <= (n - m))
    {
        int j = m-1;

        /* Keep reducing index j of pattern while characters of
           pattern and text are matching at this shift s */
        while(j >= 0 && pat[j] == txt[s+j])
            j--;

        /* If the pattern is present at current shift, then index j
           will become -1 after the above loop */
        if (j < 0)
        {
            printf("\n pattern occurs at shift = %d", s);

            /* Shift the pattern so that the next character in text
               aligns with the last occurrence of it in pattern.
               The condition s+m < n is necessary for the case when
               pattern occurs at the end of text */
            s += (s+m < n)? m-badchar[txt[s+m]] : 1;
```

```
        }

        else
            /* Shift the pattern so that the bad character in text
               aligns with the last occurrence of it in pattern. The
               max function is used to make sure that we get a positive
               shift. We may get a negative shift if the last occurrence
               of bad character in pattern is on the right side of the
               current character. */
            s += max(1, j - badchar[txt[s+j]]);
    }
}

/* Driver program to test above funtion */
int main()
{
    char txt[] = "ABAAABCD";
    char pat[] = "ABC";
    search(txt, pat);
    return 0;
}
```

Output:

```
pattern occurs at shift = 4
```

The Bad Character Heuristic may take O(mn) time in worst case. The worst case occurs when all characters of the text and pattern are same. For example, txt[] = “AAAAAAAAAAAAAAAAA” and pat[] = “AAAAA”.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

Related Topics:

- Recursively print all sentences that can be formed from list of word lists
- Check if a given sequence of moves for a robot is circular or not
- Find the longest substring with k unique characters in a given string
- Function to find Number of customers who could not get a computer
- Find maximum depth of nested parenthesis in a string
- Find all distinct palindromic sub-strings of a given string
- Find if a given string can be represented from a substring by iterating the substring “n” times
- Suffix Tree Application 6 – Longest Palindromic Substring

Tags: Pattern Searching

Writing code in comment? Please use ideone.com and share the link here.

Recommend 1 Share Sort by Newest

Join the discussion…

helper • 6 months ago

a text of length n and a patten of length m:

the pattern can occur at shifts s=0,1,2,.....,n-m-1,n-m.

after having checked for a shift s, we can "skip" some of the shifts.

Some persons were asking the doubt why we initialize badarray[i] by -1 and not by 0.... i have tried to explain the funda behind this algo and also clarify taht initialization thing.

http://postimg.org/image/aexxe...

or

http://i.imgur.com/gqvIRwd.png...

4 • Reply • Share ›

rachit singhal • 8 months ago

Please share the link of good suffix part also.

• Reply • Share ›

clinton • a year ago

i'm getting d following warnings for

s += maximum(1, j - badchar[T[s+j]]);

[Warning] array subscript has type 'char' [-Wchar-subscripts]

it says T[s+j] is a character type for badchar["char"]...how can be that possible?
plz help

• Reply • Share ›

Zheng Luo • a year ago

I think this code is better to understand

http://ideone.com/jmIbwu

2 • Reply • Share ›

Abhijeet Sachdev ➚ Zheng Luo • 7 days ago

I think there is some problem in code

table[(size_t)t[i] should be :

table[(size_t)t[i + k]

• Reply • Share ›

Rahul Kumar Zheng Luo • 10 months ago

can you explain your code.....

• Reply • Share ›

Code_Addict • a year ago

java version of above(running for all cases:)
http://ideone.com/UOullP

1 • Reply • Share ›

Anitesh Kumar • a year ago

Initialization of badchar[] array should be with 0, not with -1.

Please consider the following:

char txt[] = "ABAAAABAACD";
char pat[] = "AA";

Matches should happen for 2,4 and 7.
If the initialization of badchar[] is done with -1, then matches happen for 2 and 7 only. 4th place is getting skipped.

• Reply • Share ›

GeeksforGeeks Mod Anitesh Kumar • a year ago

Please take a closer look. It finds all occurrences. See http://ideone.com/FhJok5 for a sample run.

1 • Reply • Share ›

annonymoe • a year ago

can you improve bad character heuristic as this one is very limited by early mismatch as being the best case or else if j< last occurance of bad character u increment by 1 only

• Reply • Share ›

alien • 2 years ago

@GeeksforGeeks: Could you please post explanation and implementation of Good Suffix Heuristic.

1 • Reply • Share ›

srinivas • 2 years ago

& displayed instead of ampersand symbol

" displayed instead of quotes
Pls check this displaying issue.

• Reply • Share ›

GeeksforGeeks ➦ srinivas • 2 years ago

Thanks for pointing this out. We have updated the post.

• Reply • Share ›

abhishek08aug • 2 years ago

Intelligent :D

• Reply • Share ›

Ramesh.Mxian • 2 years ago

Following is the Java source code for Boyer Moore Algorithm with 2D array for Bad Heuristic .

```
public int[][] getBadHeuristics2D(String pattern) {
                final int MAX_CHAR = 256;
                int[][] badHeuristics = new int[256][pattern.length()];
                for (int i = 0; i < MAX_CHAR; i++) {
                        for (int j = 0; j < pattern.length(); j++) {
                                badHeuristics[i][j] = -1;

                        }
                }

                for (int i = 0; i < pattern.length(); i++) {
                        badHeuristics[(int) pattern.charAt(i)][i] = i;
                }

                int lastIndex;
                for (int i = 0; i < MAX_CHAR; i++) {
```

see more

• Reply • Share ›

atul • 3 years ago

i guess this checking is not required.
s += (s+m < n)? m-badchar[txt[s+m]] : 1;

because if s+m < n then total number character in patten is greater than total number of text , which are required to be traversed.
so we can break it anyway
[sourcecode language="C"]
/* Paste your code here (You may delete these lines if not writing code) */

• Reply • Share ›

dheeraj • 3 years ago

A 2D array can be used to always get a positive shift from the bad character heuristic.

• Reply • Share ›

GeeksforGeeks dheeraj • 3 years ago

@dheeraj: yes, we can use a 2D array to always get a positive shift. We can build a 2D array of size m*NO_OF_CHARS where we store the last occurrence of the character before every possible index (0 to n-1) in pattern. We will be covering that in a separate post.

• Reply • Share ›

Ramesh.Mxian GeeksforGeeks • 2 years ago

Hi GeeksForGeeks,

I have posted the java code for 2D based implementation. Kindly review it and give your opinion. If it is good then please post it as separate post.

• Reply • Share ›

azee GeeksforGeeks • 2 years ago

Have you posted algorithm using good heuristic and also the two dimensional array ?

```
/* Paste your code here (You may delete these lines if not writing code) */
```

• Reply • Share ›

GeeksforGeeks azee • 2 years ago

@azee: No, we have posted yet.

• Reply • Share ›

cracker • 3 years ago

Comments in code are awesome, made it easy to understand.

• Reply • Share ›

Popular Posts

- All permutations of a given string
- Memory Layout of C Programs
- Understanding “extern” keyword in C
- Median of two sorted arrays
- Tree traversal without recursion and without stack!
- Structure Member Alignment, Padding and Data Packing
- Intersection point of two Linked Lists
- Lowest Common Ancestor in a BST.
- Check if a binary tree is BST or not
- Sorted Linked List to Balanced BST

Recent Comments

- lt_k

  i need help for coding this function in java...

  Java Programming Language · 2 hours ago

- Piyush

  What is the purpose of else if (recStack[*i])...

  Detect Cycle in a Directed Graph · 2 hours ago

- Andy Toh

  My compile-time solution, which agrees with the...

  Dynamic Programming | Set 16 (Floyd Warshall Algorithm) · 2 hours ago

- lucy

  because we first fill zero in first col and...

  Dynamic Programming | Set 29 (Longest Common Substring) · 2 hours ago

- lucy

  @GeeksforGeeks i don't n know what is this long...

  Dynamic Programming | Set 28 (Minimum insertions to form a palindrome) · 3 hours ago

- manish

  Because TAN is not a subsequence of RANT. ANT...

  Given two strings, find if first string is a subsequence of second · 3 hours ago