

## 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
```

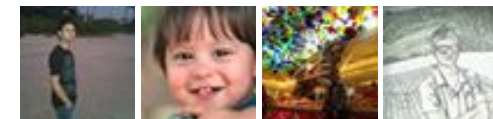
Google™ Custom Search



GeeksforGeeks



52,731 people like GeeksforGeeks.



Interview Experiences

Advanced Data Structures

Dynamic Programming

Greedy Algorithms

Backtracking

Pattern Searching

Divide & Conquer

Mathematical Algorithms

Recursion

Geometric Algorithms

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 preprocesses 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.

# ITT Tech - Official Site

[itt-tech.edu](http://itt-tech.edu)

Tech-Oriented Degree Programs.  
Education for the Future.



## 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](#)

```

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

#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--;
    }
}

```

Custom market  
research at scale.

Get \$75 off

 Google consumer surveys



## Recent Comments

affizerv Your example has two 4s on row 3, that's why it...

[Backtracking | Set 7 \(Sudoku\)](#) · 16 minutes ago

**RVM** Can someone please elaborate this Qs from above...

[Flipkart Interview | Set 6](#) · 36 minutes ago

**Vishal Gupta** I talked about as an Interviewer in general,...

[Software Engineering Lab, Samsung Interview | Set 2](#) · 36 minutes ago

**@meya** Working solution for question 2 of 4f2f round....

[Amazon Interview | Set 53 \(For SDE-1\)](#) · 1 hour ago  
sandeep void rearrange(struct node \*head)  
{...

Given a linked list, reverse alternate nodes and append at the end · 2 hours ago

Neha I think that is what it should return as, in...

[Find depth of the deepest odd level leaf node](#) · 2 hours ago

AdChoices 

[► Algorithm Java](#)

[► Pattern Matching](#)

[► Java Source Code](#)

```
/* 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 function */
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[] = "AAAAAAAAAAAAAAAAAAAA"` and `pat[] = "AAAAA"`.

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

about the topic discussed above:



Better Than Hadoop.

HPCC Systems is Big Data Processing and Analytics  
Open Source. Proven. Trusted.

 LexisNexis


Learn More 

AdChoices 

► [Java Pattern](#)

► [Pattern Shift](#)

► [Pattern File](#)

AdChoices 

► [Character Java](#)

► [Character Set](#)

► [Java Array](#)

## Related Tpoics:

- [Printing Longest Common Subsequence](#)
- [Suffix Array | Set 2 \(nLogn Algorithm\)](#)
- [Rearrange a string so that all same characters become d distance away](#)
- [Recursively remove all adjacent duplicates](#)
- [Find the first non-repeating character from a stream of characters](#)
- [Dynamic Programming | Set 33 \(Find if a string is interleaved of two other strings\)](#)
- [Remove "b" and "ac" from a given string](#)
- [Dynamic Programming | Set 29 \(Longest Common Substring\)](#)



1



Tweet

1



1

Writing code in comment? Please use [ideone.com](#) and share the link here.

17 Comments

GeeksforGeeks

Sort by Newest ▼



Join the discussion...



**Zheng Luo** · a month ago

I think this code is better to understand

<http://ideone.com/jmlbwu>

2 ^ | v · Reply · Share ›



**Code\_Addict** · 4 months ago

java version of above(running for all cases:)

<http://ideone.com/UOullP>

1 ^ | v · Reply · Share ›



**Anitesh Kumar** · 5 months 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

^ | v · Reply · Share ›



**GeeksforGeeks** Mod ➔ Anitesh Kumar · 7 days ago

Please take a closer look. It finds all occurrences. See <http://ideone.co>

^ | v · Reply · Share ›



**anonymoe** · 6 months ago

can you improve bad character heuristic as this one is very limited by early mi  
if j< last occurrence of bad character u increment by 1 only

^ | v • Reply • Share ›



**alien** • 8 months ago

@GeeksforGeeks: Could you please post explanation and implementation of C

1 ^ | v • Reply • Share ›



**srinivas** • a year ago

& displayed instead of ampersand symbol

" displayed instead of quotes

Pls check this displaying issue.

^ | v • Reply • Share ›



**GeeksforGeeks** → srinivas • a year ago

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

^ | v • Reply • Share ›



**abhishek08aug** • a year ago

Intelligent :D

^ | v • Reply • Share ›



**Ramesh.Mxian** • a year ago

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

```
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;

```

[see more](#)

^ | v • Reply • Share ›



**atul** • 2 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  
be traversed.

so we can break it anyway

[sourcecode language="C"]

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

^ | v • Reply • Share ›



**dheeraj** • 2 years ago

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

^ | v • Reply • Share ›



**GeeksforGeeks** → dheeraj • 2 years ago

@dheeraj: yes, we can use a 2D array to always get a positive shift. We  
m\*NO\_OF\_CHARS where we store the last occurrence of the charact  
1) in pattern. We will be covering that in a separate post.

^ | v • Reply • Share ›





**Ramesh.Mxian** → GeeksforGeeks · a year ago

Hi GeeksForGeeks,

I have posted the java code for 2D based implementation. Kindly  
good then please post it as separate post.

^ | v · Reply · Share ›



**azee** → GeeksforGeeks · a year ago

Have you posted algorithm using good heuristic and also the tw

```
/* Paste your code here (You may delete these lines if r
```

^ | v · Reply · Share ›



**GeeksforGeeks** → azee · a year ago

@azee: No, we have posted yet.

^ | v · Reply · Share ›



**cracker** · 2 years ago

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

^ | v · Reply · Share ›

 [Subscribe](#)

 [Add Disqus to your site](#)

