REPUBLIK INDONESIA
KEMENTERIAN HUKUM DAN HAK ASASI MANUSIA

SURAT PENCATATAN
CIPTAAN

Dalam rangka pelindungan ciptaan di bidang ilmu pengetahuan, seni dan sastra berdasarkan Undang-Undang Nomor 28 Tahun 2014 tentang Hak Cipta, dengan ini menerangkan:

Nomor dan tanggal permohonan	:	EC00202311545, 7 Februari 2023
Pencipta		
Nama	:	**Drs. Ir. Faisal Syafar, M.Si., M.InfTech., Ph.D., IPU.**
Alamat	:	BTN Tabaria Tower E10/23 Kelurahan Mannuruki, Kecamatan Tamalate, Makassar, SULAWESI SELATAN, 90221
Kewarganegaraan	:	Indonesia
Pemegang Hak Cipta		
Nama	:	**Drs. Ir. Faisal Syafar, M.Si., M.InfTech., Ph.D., IPU.**
Alamat	:	BTN Tabaria Tower E10/23 Kelurahan Mannuruki, Kecamatan Tamalate, Makassar, SULAWESI SELATAN, 90221
Kewarganegaraan	:	Indonesia
Jenis Ciptaan	:	**Program Komputer**
Judul Ciptaan	:	**Data Pattern Matching Menggunakan Algoritma Boyer Moore**
Tanggal dan tempat diumumkan untuk pertama kali di wilayah Indonesia atau di luar wilayah Indonesia	:	7 Februari 2023, di Makassar
Jangka waktu pelindungan	:	Berlaku selama 50 (lima puluh) tahun sejak Ciptaan tersebut pertama kali dilakukan Pengumuman.
Nomor pencatatan	:	000444468

adalah benar berdasarkan keterangan yang diberikan oleh Pemohon.
Surat Pencatatan Hak Cipta atau produk Hak terkait ini sesuai dengan Pasal 72 Undang-Undang Nomor 28 Tahun 2014 tentang Hak Cipta.

a.n Menteri Hukum dan Hak Asasi Manusia
Direktur Jenderal Kekayaan Intelektual
u.b.
Direktur Hak Cipta dan Desain Industri

Anggoro Dasananto
NIP.196412081991031002

Disclaimer:
Dalam hal pemohon memberikan keterangan tidak sesuai dengan surat pernyataan, Menteri berwenang untuk mencabut surat pencatatan permohonan.

Data Pattern Matching Menggunakan Algoritma Boyer Moore

Pernyataan masalah

Kami diberi pola input dan teks (atau string), dan kami perlu mencari dan mencetak semua kemunculan pola dalam string.

Catatan: Gunakan algoritma Boyer Moore untuk mencari pola dalam string.

Input pertama adalah ukuran string aktual yaitu n.

Input kedua adalah ukuran string pola yaitu m.

Input ketiga berisi string yang sebenarnya.

Masukan keempat berisi pola yang akan dicari.

Lihat bagian Contoh dan Penjelasan untuk detail lebih lanjut dan bagian Pendekatan untuk memahami cara kerja algoritma Boyer Moore.

Contoh

Mari kita lihat beberapa contoh yang diberikan untuk mencari pencarian pola menggunakan algoritma Boyer-Mooreoore.

Contoh 1: text = "Topik Penskala" pattern = "Topik"

Keluaran: Pola ditemukan pada indeks: 7.

Contoh 2: text = "dia melakukan apa yang disuruh" pattern = "dia"

Keluaran: Pola ditemukan pada indeks: 0. Pola ditemukan pada indeks: 17.

Penjelasan Contoh

Sebelum masuk ke penjelasan bagaimana mencari atau menemukan kemunculan pola tertentu dalam sebuah string, mari kita kenali dulu tentang string.

String adalah tipe data yang tidak dapat diubah yang digunakan untuk menyimpan urutan karakter. String adalah salah satu tipe data yang paling banyak digunakan dalam bahasa

pemrograman apa pun. Sebuah string dapat dengan mudah dibuat menggunakan tanda kutip (baik tanda kutip tunggal atau ganda).

Sekarang, untuk mencari pola pada string, kita dapat menggunakan algoritma pencarian pola Boyer Moore (atau algoritma B-M). Intuisi algoritma Boyer Moore sangat sederhana, dua pointer disejajarkan pada indeks ke-0 string teks dan string karakter. String pola kemudian dibandingkan karakter demi karakter dengan bagian string teks saat ini, dimulai dengan karakter paling kanan.

Nah, jika karakter tidak cocok maka algoritma Boyer-Moore menggeser karakter menggunakan dua strategi secara bersamaan. Jadi, dapat dikatakan bahwa algoritma Boyer Moore merupakan gabungan dari dua pendekatan yaitu:

Heuristik Karakter Buruk, dan

Akhiran Heuristik yang Baik.

Lihat bagian berikut untuk memahami kedua pendekatan tersebut.

Sekarang, kembali ke contoh, teks yang disediakan adalah: text = "Scaler Topics" dan pola yang akan dicari adalah: pattern = "Topics". Sekarang, dalam contoh ini, kita akan memulai pencarian dengan karakter pertama yaitu indeks-0, dan pola ditemukan dari indeks-7 hingga indeks-12, tetapi kita perlu mencetak indeks tempat kemunculan pola dimulai. Jadi, kami akan mencetak indeks-

7. Demikian pula pada contoh kedua: text = "dia melakukan apa yang diperintahkan." pattern = "he" Kemunculan pertama he ditemukan pada indeks-0 dan sekali lagi pada indeks-17. Jadi, kami mencetak kedua indeks tersebut.

Kendala

$$−1≤|\text{?}|≤105−1≤|s|≤10^5$$

Di sini, s menunjukkan string pola dan string teks.

Dalam beberapa masalah, Anda mungkin menemukan jumlah kasus uji yang diwakili oleh t Jadi, kita hanya perlu memanggil fungsi BoyerMooreAlgorithm `to-times.

Permasalahan yang dicari adalah pencarian pola menggunakan algoritma Boyer-Moore.

```
#include <bits/stdc++.h>
using namespace std;

// defining a macro
#define TOTAL_CHARACTERS 256

// The preprocessing function that fills the actual value of the last occurrence of a character
void badCharacterHeuristic(string s, int n,
                           int badCharacter[TOTAL_CHARACTERS]) {
  // First initializing all occurrences with -1
  for (int i = 0; i < TOTAL_CHARACTERS; i++)
    badCharacter[i] = -1;

  // Filling the actual value of the last occurrence of a character in the bad character array.
  for (int i = 0; i < n; i++)
    badCharacter[(int)s[i]] = i;
}

/*
A function that searches the string and returns the occurrence.
*/
void BoyerMooreAlgorithm(string text, string pattern) {
  // getting the size of the pattern and the string.
  int m = pattern.size();
  int n = text.size();

  // defining a a character array.
  int badCharacter[TOTAL_CHARACTERS];
```

```
/*
Filling the badCharacter array with the current pattern using a helper function.
*/
badCharacterHeuristic(pattern, m, badCharacter);

// initially there is no shift
int shifts = 0;

// defining the boundaries of the searching
while (shifts <= (n - m)) {
  int j = m - 1;

  /*
  If the pattern and text are matching, keep on reducing the j variable.
  */
  while (j >= 0 && pattern[j] == text[shifts + j])
    j--;

  /*
  If the pattern is present at current
  shift, then j will become -1.
  */
  if (j < 0) {
    cout << "Pattern found at index: " << shifts << endl;

    /*
```

```
        Shifting the pattern so that the next character of the text is aligned with its last occurrence in the pattern.

        */

        shifts += (shifts + m < n) ? m - badCharacter[text[shifts + m]] : 1;

      }


      /*

      Shifting the pattern so that the bad character of the text is aligned with its last occurrence in the pattern. The max function helps to get the positive shift only.

      */

      else

        shifts += max(1, j - badCharacter[text[shifts + j]]);

    }

}


int main() {

  string text = "Scaler Topics";

  string pattern = "Topics";


  BoyerMooreAlgorithm(text, pattern);


  return 0;

}

public class Test {


        static int TOTAL_CHARACTERS = 256;


        // The preprocessing function that fills the actual value of the last occurrence of a character
```

```
static void badCharacterHeuristic(char s[], int n,
int badCharacter[]) {
	// First initializing all occurrences with -1
	for (int i = 0; i < TOTAL_CHARACTERS; i++)
		badCharacter[i] = -1;

	// Filling the actual value of the last occurrence of a character in the badCharacter array.
	for (int i = 0; i < n; i++)
		badCharacter[(int)s[i]] = i;
}

/*
A function that searches the string and returns the occurrence.
*/
static void BoyerMooreAlgorithm(char text[], char pattern[]) {
	// getting the size of the pattern and the string.
	int m = pattern.length;
	int n = text.length;

	// defining a bacCharacter array.
	int badCharacter[] = new int[TOTAL_CHARACTERS];

	/*
	Filling the badCharacter array with the current pattern using a helper function.
	*/
	badCharacterHeuristic(pattern, m, badCharacter);

	// initially there is no shift
```

```
int shifts = 0;

// defining the boundaries of the searching
while (shifts <= (n - m)) {
    int j = m - 1;

    /*
    If the pattern and text are matching, keep on reducing the j variable.
    */
    while (j >= 0 && pattern[j] == text[shifts + j])
        j--;

    /*
    If the pattern is present at current
    shift, then j will become -1.
    */
    if (j < 0) {
        System.out.println("Pattern found at index: " + shifts);

    /*
    Shifting the pattern so that the next character of the text is aligned with its last occurrence in the pattern.
    */
    shifts += (shifts + m < n) ? m - badCharacter[text[shifts + m]] : 1;
    }

    /*
```

```
                        It is shifting the pattern so that the bad character of the text is aligned with its last occurrence in the pattern. The max function helps to get the positive shift only.
                        */
                        else
                                shifts += Math.max(1, j - badCharacter[text[shifts + j]]);
                }
        }

        public static void main(String args[]) {

                char text[] = "Scaler Topics".toCharArray();
                char pattern[] = "Topics".toCharArray();

                BoyerMooreAlgorithm(text, pattern);
        }
}
```

```
TOTAL_CHARACTERS = 256

"""
The preprocessing function that fills the actual value of the last occurrence of a character
"""
def badCharacterHeuristic(string, size):
   # First initializing all occurrences with -1
   Bachar = [-1]*TOTAL_CHARACTERS

   # Filling the actual value of the last occurrence of a character in the badCharacter list (array).
   for i in range(size):
```

```
        badChar[ord(string[i])] = i

    return badChar


"""
A function that searches the string and returns the occurrence.
"""
def boyerMooreAlgorithm(text, pattern):
    # getting the size of the pattern and the string.
    m = len(pattern)
    n = len(text)

    # defining a bacCharacter array.
    # Filling the badCharacter array with the current pattern using a helper function.
    badCharacter = badCharacterHeuristic(pattern, m)

    # initially there is no shift
    shifts = 0

    # defining the boundaries of the searching
    while(shifts <= n-m):
        j = m-1

        # If the pattern and text are matching, keep on reducing the j variable.
        while j >= 0 and pattern[j] == text[shifts+j]:
            j -= 1
```

```
        # If the pattern is present at the current shift, then j will become -1.
        if j < 0:
            print("Pattern found at index: ", shifts)

            '''
            Shifting the pattern so that the next character of the text is aligned with its last occurrence in the pattern.
            '''
            shifts += (m-badCharacter[ord(text[shifts+m])]
                    if shifts+m < n else 1)
        else:
            """
            Shifting the pattern so that the bad character of the text is aligned with its last occurrence in the pattern. The max function helps to get the positive shift only.
            """
            shifts += max(1, j-badCharacter[ord(text[shifts+j])])


text = "Scaler Topics"
pattern = "Topics"
boyerMooreAlgorithm(text, pattern)
```

Kesimpulan

Pola ditemukan pada nilai index: 7