**Pencarian *String* Menggunakan Metode Boyer Moore**

**Mata Kuliah Pemrograman Berorientasi Objek**



**Disusun Oleh:**

**Kelompok 8**

**Risyad Pangestu (140810170003)**

**Muhammad Luthfiansyah (140810170023)  
Imron Madani (140810170061)**

**Program Studi Teknik Informatika**

**Fakultas Matematika dan Ilmu Pengetahuan Alam**

**Universitas Padjajaran**

**2018**

# **Kata Pengantar**

Puji syukur kami panjatkan kepada Allah SWT atas limpahan rahmat taufiq dan hidayah-Nya, sehingga penyusun mendapat kesempatan untuk menyelesaikan makalah yang berjudul “**Pencarian *String* Menggunakan Metode Boyer Moore**” sesuai dengan yang diharapkan.

Makalah ini dibuat untuk memenuhi salah satu tugas mata kuliah Pemrograman Berbasis Objek dan besar harapan penyusun makalah ini dapat menambah wawasan penyusun dan pembaca tentang apa itu pencarian string khususnya menggunakan metode Boyer Moore.

Penyusun juga ingin mengucapkan terima kasih kepada pihak-pihak yang terkait yang membantu penyusun hingga makalah ini bisa diselesaikan. penyusun juga ingin memohon maaf yang sebesar-besarnya apabila masih banyak kekurangan dalam penulisan kata, sumber, maupun informasi yang penyusun tuliskan. Untuk itu penyusun sangat mengharapkan kritik dan saran yang membangun agar kedepannya penyusun dapat lebih baik.

Sumedang, 7 Oktober 2018

Penyusun

# **Daftar Isi**

[**Kata Pengantar** ii](#_Toc526756146)

[**Daftar Isi** iii](#_Toc526756147)

[**BAB 1 PENDAHULUAN** 1](#_Toc526756148)

[**A.** **Latar Belakang** 1](#_Toc526756149)

[**B.** **Perumusan Masalah** 1](#_Toc526756150)

[**C.** **Tujuan Penulisan** 1](#_Toc526756151)

[**BAB 2 PEMBAHASAN** 1](#_Toc526756152)

[**A.** **Dasar Teori** 1](#_Toc526756153)

[**B.** **Algoritme Boyer-Moore** 2](#_Toc526756154)

[**C.** **Bagan Alur** 2](#_Toc526756155)

[**D.** **Pencarian Manual** 2](#_Toc526756156)

[**E.** **Kode Program** 3](#_Toc526756157)

[**F.** **Hasil Tangkapan Layar** 3](#_Toc526756158)

[**BAB 3 PENUTUP** 3](#_Toc526756159)

[**A.** **Kesimpulan** 3](#_Toc526756160)

[**B.** **Saran** 3](#_Toc526756161)

[**DAFTAR PUSTAKA** 3](#_Toc526756162)

# **BAB 1 PENDAHULUAN**

1. **Latar Belakang**

Pemrograman menjadi suatu hal yang penting saat ini. Tak dapat dipungkiri, dengan adanya program komputer hidup manusia makin dimudahkan. Banyak kegiatan manusia yang saat ini dapat digantikan atau dipermudah oleh adanya program komputer.

Salah satu permasalahan yang ada adalah ketika pengguna ingin mencari suatu kata atau pola dari suatu teks. Banyak cara untuk mencapai tujuan tersebut, tetapi yang dicari adalah yang paling tepat dan paling efektif diantara yang lain.

Beberapa contohnya adalah metode *Brute-Force,* metode *Rabin-Karp,* metode *Knuth-Morris-Pratt,* metode *Boyer-Moore,* dan banyak metode lainnya. Pada kesempatan kali ini, penyusun akan membahas lebih dalam tentang metode *Boyer-Moore.*

1. **Perumusan Masalah**

1. Apa yang dimaksud metode *Boyer-Moore?*

2. Bagaimana algoritme metode *Boyer-Moore?*

3. Bagaimana contoh program metode *Boyer-Moore?*

1. **Tujuan Penulisan**

1. Mengetahui definisi dari metode *Boyer-Moore.*

2. Memahami alur dan algoritme metode *Boyer-Moore.*

3. Dapat mengaplikasikan metode *Boyer-Moore* dalam bentuk program komputer.

# **BAB 2 PEMBAHASAN**

1. **Dasar Teori**

Metode Boyer-Moore merupakan sebuah cara untuk mencocokan sebuah String dari kanan ke kiri. Sebuah teks dicocokan dengan pattern tertentu untuk menentukan apakah dalam teks yang dicocokan terdapat pattern tersebut.

Algoritma Boyer-Moore dipublikasikan oleh Robert S. Boyer, dan J. Strother Moore pada tahun 1977. Algoritma ini dianggap sebagai algoritma yang paling efisien pada aplikasi umum. Algoritma BoyerMoore memulai mencocokkan karakter dari sebelah kanan pattern. Ide dibalik algoritma ini adalah bahwa dengan memulai pencocokan karakter dari kanan, dan bukan dari kiri, maka akan lebih banyak informasi yang didapat.

Perbedaan pencocokan String Boyer-Moore dengan pencocokan String secara Brute Force adalah pada Algoritma Boyer-Moore tidak semua String dicocokan seperti pada cara Brute Force. Ketika Ada text dan pattern yang terjadi Mismatch, maka pattern akan mencocokan text meloncat menurut nilai pada tabel delta atau sebanyak jumlah karakter yang telah dicocokan. Hal ini tergantung nilai maksimum yang terdapat dari keduanya.

1. **Algoritme Boyer-Moore**

1. Menentukanteks yang akan dicari suatu pola

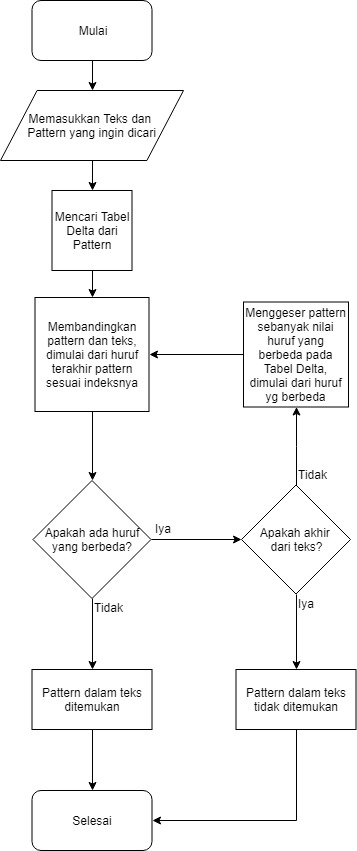
2. Buat Tabel Delta, buat Occurrence Heuristic Table, didapat dari kemunculan pertama sebuah huruf di pattern yang dihitung dari kanan, dimulai dari 0. Dan juga tabel Math Heuristic. Kedua tabel tersebut berguna untuk menentukan pergesaran saat terjadi ketidakcocokan.

3. Membandingkan pola dan teks, dimulai dari huruf terakhir pattern sesuai indeks teksnya.

4. Apabila hurufnya sama maka diteruskan membandingkan huruf kedua dari terakhir sampai huruf pertama. Jika sama semua maka pattern ditemukan.

5. Apabila ada huruf yang berbeda, maka pattern digeser sebesar nilai pada Tabel Delta, penggeseran menggunakan huruf yang berbeda sebagai patokan. Setelah digeser kemudian mengulangi membandingkan huruf sesuai indeksnya dimulai dari huruf terakhir pattern.

1. **Bagan Alur**

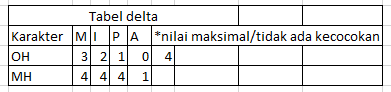
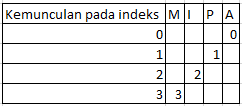
****

1. **Pencarian Manual**

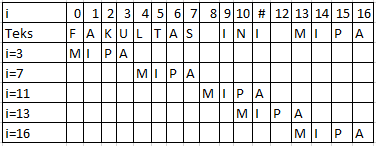
Diketahui sebuah teks : FAKULTAS INI MIPA

Dan pattern : MIPA

1. Mencari Tabel Delta buat Occurrence Heuristic Table dan Math Heuristic Table



2. Mencocokkan teks dan pattern



2.1 membandingkan indeks 3, huruf berbeda, u bukan karakter pola, maka pattern digeser sejauh 4 nilai pada tabel OH

2.2 membandingkan indeks 7, huruf berbeda, s bukan karakter pola, maka pattern digeser sejauh 4 nilai pada tabel OH

2.3 membandingkan indeks 11, huruf berbeda, i punya nilai 2, maka pola digeser sejauh 2 nilai pada tabel OH

2.4 membandingkan indeks 13, huruf berbeda, m punya nilai 3, maka pola digeser sejauh 3 nilai pada tabel OH

2.5 membandingkan indeks 16, huruf sama lanjutkan pencocokan karakter sebelumnya

2.6 membandingkan indeks 15, huruf sama huruf sama lanjutkan pencocokan karakter sebelumnya

2.7 membandingkan indeks 14, huruf sama huruf sama lanjutkan pencocokan karakter sebelumnya

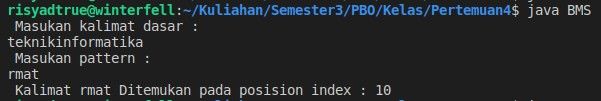
2.8 membandingkan indeks 13, huruf sama, pola ditemukan.

1. **Kode Program**

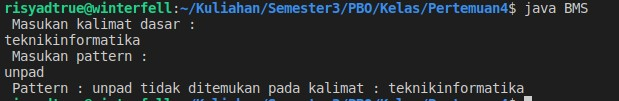
|  |
| --- |
| import java.util.Scanner;  public class BMS{  public static String kalimat;  public static String pattern;  private int[] table;  public BMS(String pattern){  int BASE = 256;  this.pattern = pattern;  table = new int[BASE];  for (int i =0; i<BASE; i++)  table[i] = -1;  for (int j = 0; j<pattern.length(); j++ )  table[pattern.charAt(j)] = j;  }  public int search(String Kalimat){  int pjgkalimat = kalimat.length();  int pjgpattern = pattern.length();  int skip;  for (int i=0; i<=pjgkalimat-pjgpattern; i+=skip){  skip = 0;  for (int j=pjgpattern-1; j>=0; j--){  if (pattern.charAt(j) != kalimat.charAt(i+j)){  skip = Math.max(1, j-table[kalimat.charAt(i+j)]);  break;  }  }  if (skip == 0) return i;  }  return pjgkalimat;  }  public static void main(String[] args) {  Scanner in = new Scanner(System.in);  System.out.println(" Masukan kalimat dasar : ");  kalimat = in.next();  System.out.println(" Masukan pattern : ");  pattern = in.next();  BoyerMoore bomo = new BoyerMoore(pattern);  int temp = bomo.search(kalimat);  if (temp < kalimat.length()){  System.out.println(" Kalimat " + pattern + " Ditemukan pada " + temp + " posision" );  }  else  System.out.println(" Pattern : " + pattern+ " tidak ditemukan pada kalimat : " + kalimat);  } |

1. **Hasil Tangkapan Layar**

Keadaan jika *pattern* ditemukan :



Keadaan jika tidak ditemukan :



Penjelasan hasil keluaran program :

1. Fungsi main akan menjalankan konstruktor yang akan membuat sebuah wadah/tabel (pada iteras bagian1), lalu pada iterasi bagian ke 2 *character* pattern akan mengisi table yang sudah ada dalam hal ini patter : rmat

( Implementasi Bad-Character Heuristic )

2. Program akan berlanjut pada fungsi search yang dapat digambarkan seperti berikut ini :



2.1 pada index ke 3 dalam hal ini ini huruf t dan n, tidak cocok maka akan ada perbandigan nilai index max (a,b) yaitu (1,3) yaitu 3 arena tidak di temukan huruf ‘n’ pada pattern.. Hasil perbandingan tersebut menjadi jumlah pergesaran.



2.2 Dikarenakan hasil pergerseran 3 index maka dilanjutkan dengan pencocokan index ke 6 yaitu t dan I, keadaan tidak cocok maka akan ada perbandigan nilai index max (a,b) yaitu (1,3) yaitu 3 arena tidak di temukan huruf ‘i’ pada pattern. Hasil perbandingan tersebut menjadi jumlah pergesaran.



2.3 Dikarenakan hasil pergerseran 3 index maka dilanjutkan dengan pencocokan index ke 9 yaitu t dan o keadaan tidak cocok maka akan ada perbandigan nilai index max (a,b) yaitu (1,3) yaitu 3 karena tidak di temukan huruf ‘o’ pada pattern. Hasil perbandingan tersebut menjadi jumlah pergesaran.



2.4 Dikarenakan hasil pergerseran 3 index maka dilanjutkan dengan pencocokan index ke 12 yaitu t dan a keadaan tidak cocok maka akan ada perbandigan nilai index max (a,b) yaitu (1,1) yaitu 1 arena di temukan huruf ‘i’ pada pattern dengan 1 pergesran .Hasil perbandingan tersebut menjadi jumlah pergesaran.



2.5 Pada kondisi ini index ke 13 yaitu t = t, maka akan di lakukan iterasi pencocokan tiap index maksimal pattern sampai 0 yang ternyata hasilnya cocok semuanya. Tampilan akhir adalah memunculkan persamaan index petama dari pattern dan kalimat index di kalimat.

# **BAB 3 PENUTUP**

1. **Kesimpulan**
2. **Saran**

# **DAFTAR PUSTAKA**

*CHAPTER 34: STRING MATCHING*. (n.d.). Retrieved from http://staff.ustc.edu.cn/~csli/graduate/algorithms/book6/chap34.htm

Hartoyo, E. G., Vembrina, Y. G., & Meilana, A. F. (n.d.). *Analisis Algoritma Pencarian String (String Matching).* Retrieved from http://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/Makalah/MakalahStmik10.pdf

Kumara, G. H. (n.d.). *Visualisasi Beberapa Algoritma Pencocokan String Dengan.* Retrieved from [http://informatika.stei.itb.ac.id/~rinaldi.munir/TA/Makalah\_TA%20Gozali.pdf](http://informatika.stei.itb.ac.id/~rinaldi.munir/TA/Makalah_TA Gozali.pdf)