**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**

**Daftar Isi**

[**A.** **Dasar Teori** 3](#_Toc526260348)

[**B.** **Algoritma Boyer-Moore** 3](#_Toc526260349)

[**C.** **Flowchart** 4](#_Toc526260350)

[**D.** **Pencarian Manual** 5](#_Toc526260351)

[**E.** **Source Code** 6](#_Toc526260352)

[**F.** **Screenshot** 9](#_Toc526260353)

[**G.** **Daftar Pustaka** 11](#_Toc526260354)

1. **Dasar Teori**

Pencarian string merupakan pencarian suatu teks yang terdiri dari beberapa karakter dalam suatu kalimat yang terdiri dari banyak teks. Ada banyak metode yang dapat digunakan untuk melakukan pencarian string tersebut, metode-metode tersebut memiliki kelebihan dan kekurangannya masing-masing dari segi algoritma maupun keefektifannya. Dalam makalah ini kami akan membahas salah satu metode yaitu metode pencarian string Boyer-Moore.

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. **Algoritma Boyer-Moore**

1. Menentukanteks(string) yang akan dicari suatu pola(pattern)

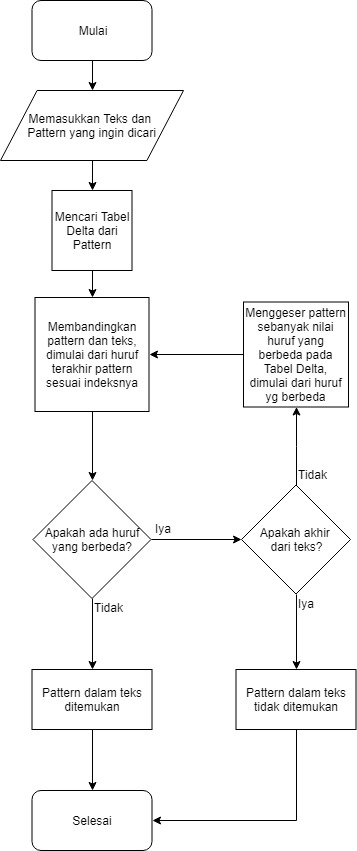
2. Mencari Tabel Delta atau Occurrence Heuristic Table, didapat dari kemunculan pertama sebuah huruf di pattern yang dihitung dari kanan, dimulai dari 0.

3. Membandingkan pattern 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. **Flowchart**

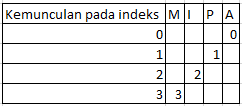
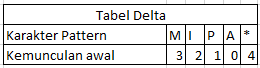


1. **Pencarian Manual**

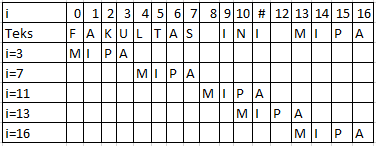
Diketahui sebuah teks : FAKULTAS INI MIPA

Dan pattern : MIPA

1. Mencari Tabel Delta atau Occurrence Heuristic Table

2. Mencocokkan teks dan pattern



2.1 membandingkan indeks 3, huruf berbeda, u bukan karakter pattern, maka pattern digeser sejauh 4 indeks

2.2 membandingkan indeks 7, huruf berbeda, s bukan karakter pattern, maka pattern digeser sejauh 4 indeks

2.3 membandingkan indeks 11, huruf berbeda, i punya nilai 2, maka pattern digeser sejauh 2 indeks

2.4 membandingkan indeks 13, huruf berbeda, m punya nilai 3, maka pattern digeser sejauh 3 indeks

2.5 membandingkan indeks 16, huruf sama

membandingkan indeks 15, huruf sama

membandingkan indeks 14, huruf sama

membandingkan indeks 13, huruf sama

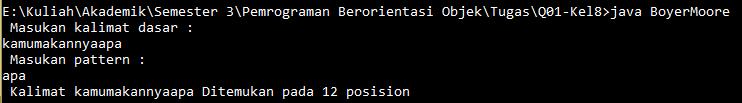
3. Pattern ditemukan

1. **Source Code**

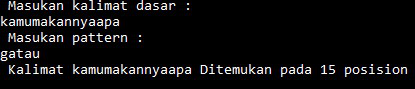
|  |
| --- |
| import java.util.Scanner;  public class BoyerMoore{  public static String kalimat;  public static String pattern;  private int[] table;  public BoyerMoore(String pattern){  int BASE = 256;  this.pattern = pattern;  table = new int[BASE];  for (int i =0; i<BASE; i++)  table[i] = -1;  for (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);  System.out.println(" Kalimat " + bomo.kalimat + " Ditemukan pada " + temp + " posision" );  }  } |

1. **Screenshot**

Jika ditemukan :



Error Handling (Tidak match) :



\*error handling belum sempurna

1. **Daftar Pustaka**

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