Penerapan Algoritma Boyer-Moore dan Algoritma Rabin-Karp dalam Mendeteksi Aksi Plagiatisme

Arif Prasetiya 13508090
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
arifwng08@students.itb.ac.id, if18090@students.if.itb.ac.id

Abstraksi— Plagiatisme sering menjadi pada banyak institusi pendidikan termasuk Perguruan Tinggi. Praktek Plagiatisme biasa dilakukan terhadap konten digital, dapat berupa dokumen maupun source code program. Plagiatisme dilakukan dengan melakukan copy-paste, atau dengan melakukan modifikasi konten digital yang asli. Untuk dapat mencegahnya, diperlukan cara yang dapat menganalisis, mengecek/mendeteksi teknik-teknik plagiatisme yang dilakukan. Salah satu pendekatannya adalah dengan melakukan penerapan algoritma Boyer-Moore atau Algoritma Rabin-Karp. Makalah ini akan mencoba menganalisis penerapan dari kedua algoritma Boyer Moore dan algoritma Rabin-Karp dalam mendeteksi plagiatisme pada suatu source code atau dokumen.

***Kata Kunci* —Boyer-Moore, Pattern maching, Plagiatisme, Rabin-Karp.**

I. PENDAHULUAN

Source code dan Dokumen dalam praktik pendidikan adalah konten digital yang paling sering dijiplak atau dilakukan plagiatisme. Praktik plagiatisme ini sangat sering dilakukan terutama di kalangan akademisi baik sekolah maupun di perguruan tinggi. Tindakan Plagiatisme ini dapat berdampak pada penurunan kreatifitas siswa maupun mahasiswa. Teknik Plagiatisme yang paling sering dilakukan dengan melakukan copy-paste-edit suatu dokumen baik merupakan hasil pekerjaan teman ataupun dokumen lain yang diperoleh baik dari meng-copy dari pemiliknya maupun mengambil dari Internet agar tidak diketahui biasanya proses editing yang sangat menentukan agar menjadi dokumen yang berbeda dari aslinya.

Upaya melakukan pengececekan atau pendeteksian plagiatisme yang pertama adalah dilakukan secara manual, hal yang terpenting yang dilakukan adalah mengetahui bagian inti dari suatu dokumen. Metode lainnya adalah dengan mengumpulkan hasil dokumen-dokumen terdahulu dan kemudian menyusunnya menjadi suatu katalog. Hal ini biasanya dilakukan oleh institusi besar yang memiliki mahasiswa dengan karya yang sangat banyak dan dimungkinkan terjadi penjiplakan atau plagiatisme. Selain itu pengajar bisa juga melakukan pendeteksian terhadap gaya penulisan dokumen mahasiswanya. Dari tiga metode manual tersebut biasanya memiliki masalah terhadap efisensi dalam melakukan pendeteksian. Dibutuhkan waktu yang cukup lama dan mungkin akan sulit dilakukan jika pemeriksaan harus dilakukan terhadap ribuan atau puluhan ribu dokumen lainnya dibandingkan dengan yang sudah tersimpan di dalam katalog yang ada. Terlebih lagi dimungkinkan terdapat gaya penulisan yang mungkin bisa salah tafsir.

Metode kedua adalah menggunakan software yang bisa melakukan pencarian atau pencocokan string. Langkahnya adalah memasukkan judul atau kata kunci tertentu pada dokumen dan membandingkan dengan dokumen yang pernah ada dan tersimpan di katalog dokumen. Hal ini akan berguna jika penjiplakan yang dilakukan adalah copy-paste langsung tanpa dilakukan edit. Namun dirasa kurang efektif juga apabila pelaku plagiatisme hanya menggunakan sebagian dari dokumen atau pecahan dokumen. Metode kedua ini juga dirasa memiliki kelemahan yang hampir sama dengan metode manual yaitu membutuhkan waktu yang mungkin tidak efektif. Bedanya kali ini pendeteksiannya dilakukan dengan bantuan software.

Metode berikutnya yang dapat dilakukan dalam upaya pendeteksian plagiatisme adalah dengan melakukan pembandingan dengan dokumen asli. Terdapat dua algoritma yang bisa digunakan dalam melakukan pembandingan ini yang pertama adalah dengan Algoritma Boyer-Moore (biasa juga digunakan untuk algoritma pencocokan string). Dalam penggunaan algoritma Boyer-Moore ini teknik pendeteksiannya dikhususkan pada source code program dengan karakteristik yang sedikit berbeda dengan text biasa pada umumnya, karena pendeteksiannya berdasar pada alur program bukan kesamaan tekstual. Algoritma yang kedua adalah Algoritma Rabin-Karp adalah algoritma yang ditemukan oleh Michael Rabin dan Richard Karp. Algoritma ini menggunakan *Hashing* untuk menemukan sebuah substring dalam sebuah teks. Hashing adalah metode yang menggunakan metode yang menggunakan fungsi hash untuk mengubah suatu jenis data menjadi beberapa bilangan bulat seberhana. Disebut slgoritma “pencarian string” bukan “pencocokan string” seperti Algoritma

Boyer Moore karena memang algoritma Rabin Karp tidak bertujuan menemukan string yang cocok dengan string masukan, melainkan menemukan pola yang sekiranya sesuai dengan teks masukan yang dalam hal ini adalah dokumen yang akan dibandingkan..

II. ALGORITMA BOYER-MOORE

A. Prinsip Kerja Algoritma Boyer- Moore

Algoritma Boyer-Moore adalah salah satu algoritma pencarian string, dipublikasikan oleh Robert S. Boyer, dan J. Strother Moore pada tahun 1977.

Algoritma Boyer moore merupakan variasi lain dari pencarian string dengan melompat maju sejauh mungkin. Tetapi Algoritma Boyer Moore ini berbeda dengan algoritma Knuth-Morris-Pratt(KMP) yang melakukan pembandingan mulai dari kiri atau depan, sedangkan Boyer Moore melakukan perbandingan dari belakang (kanan). Misal kita akan membandingkan teks "PLAGIATISME", algoritma ini akan melakukan pengecekan apakah karakter ke-11 dari dokumen tersebut sama dengan karakter 'E'. Jika terjadi kesamaan maka kemudian pengecekan akan dilakukan ke karakter sebelumnya apakah sama dengan karakter 'M'. Prosedur ini dalakukan hingga ditemukan kecocokan setiap karakter hingga karakter pertama yaitu 'P.

Algoritma pencocokan string Boyer-Moore didasarkan atas dua teknik :

1. Teknik looking-glass, menemukan pattern di dalam teks dengan menggerakan pattern mundur dimulai dari akhir teks.
2. Teknik character-jump, pergeseran karakter yang dilakukan saat terjadi ketidak cocokan

Karakteristik utama algoritma Boyer-Moore :

-melakukan perbandingan dari kanan ke kiri

-fase persiapan / prepocessing membutuhkan kompleksitas waktu $O(m + \sigma)$

-fase pencarian : kompleksitas waktunya O(mn)

-pada kasus terburuk, sebanyak 3n karakter teks yang dibandingkan untuk pattern yang tak berulang.

-Kasus terbaik O(n/m)

Alasan yang melatarbelakangi malakukan pengecekan dari belakang adalah akan lebih jelas jika mengamati apa yang terjadi jika pengecekan menghasilkan nilai yang tidak cocok. Misalnya untuk kasus pencarian dengan teks kata 'PLAGIATISME' dan pada karakter ke sebelas adalah 'Z', dan karakter 'Z' tidak ada di teks "PLAGIATISME" dan berarti tidak terdapat kecocokan antara karakter 'Z' dengan semua karakter dalam "PLAGIATISME" maka dengan hanya melakukan sekali pengecekan maka dapat diabaikan pengecekan dari karakter kesepuluh 'M' hingga karakter pertama 'P' dan melanjutkan pengecekan dengan menggeser pola ke karakter ke-12 dan memulai pencocokan dari karakter ke-22 pada dokumen yang dicocokkan. Dalam contoh terlihat bahwa algoritma Boyer-Moore memiliki loncatan karakter yang besar sehingga mempercepat pencarian string karena dengan hanya memeriksa sedikit karakter, dapat langsung diketahui bahwa string yang dicari tidak ditemukan dan dapat digeser ke posisi berikutnya.

Langkah-langkah algoritma Boyer-Moore :

1. Buat tabel pergeseran string yang dicari (S) dengan pendekatan Match Heuristic (MH) dan Occurence Heuristic (OH), untuk menentukan jumlah pergeseran yang akan dilakukan jika mendapat karakter tidak cocok pada proses pencocokan dengan string (T).
2. Jika dalam proses pembandingan terjadi ketidakcocokan antara pasangan karakter pada S dan karakter pada T, pergeseran dilakukan dengan memilih salah satu nilai pergeseran dari dua tabel analisa string, yang memiliki nilai pergeseran paling besar.
3. Dua kemungkinan penyelesaian dalam melakukan pergeseran S, jika sebelumnya belum ada karakter yang cocok adalah dengan melihat nilai pergeseran hanya pada tabel occurence heuristic : Jika karakter yang tidak cocok tidak ada pada S maka pegeseran adalah sebanyak jumlah karakter pada S. dan jika karakter yang tidak cocok ada pada S, maka banyaknya pergeseran bergantung dari nilai pada tabel.
4. Jika karakter pada teks yang sedang dibandingkan cocok dengan karakter pada S, maka posisi karakter pada S dan T diturunkan sebanyak 1 posisi, kemudian lanjutkan dengan pencocokan pada posisi tersebut dan seterusnya. Jika kemudian terjadi ketidakcocokan karakter S dan T, maka pilih nilai pergeseran terbesar dari dua tabel analisa pattern yaitu nilai dari tabel match heuristic dan nilai tabel occurence heuristic dikurangi dengan jumlah karakter yang telah cocok.
5. Jika semua karakter telah cocok, artinya S telah ditemukan di dalam T, selanjutnya geser pattern sebesar 1 karakter.
6. Lanjutkan sampai akhir string T.

Cara Menghitung Tabel Occurence Heuristic :
contoh string : manaman
Panjang : 7 karakter
Tabel Occurence Heuristic

posisi :	1	2	3	4	5	6	7
string :	m	a	n	a	m	a	n
pergeseran (OH) :	2	1	0	1	2	1	0

Langkah-langkah perhitungannya adalah sebagai berikut :

1. Lakukan pencacahan mulai dari posisi terakhir string sampai ke posisi awal, dimulai dengan

nilai 1, catat karakter yang sudah ditemukan (dalam contoh ini karakter “n”)

2. Mundur ke posisi sebelumnya, nilai pencacah ditambah 1, jika karakter pada posisi ini belum pernah ditemukan, maka nilai pergeserannya adalah sama dengan nilai pencacah. (dalam contoh ini, karakter “a” belum pernah ditemukan sehingga nilai pergeserannya adalah sebesar nilai pencacah yaitu 2)
3. Mundur ke posisi sebelumnya, karakter “m” nilai pergeserannya 3
4. Mundur lagi, karakter “a”. karakter “a” sudah pernah ditemukan sebelumnya sehingga nilai pergeserannya sama dengan nilai pergesean karakter “a” yang sudah ditemukan paling awal yaitu 2.
5. Begitu seterusnya sampai posisi awal string.

catatan : untuk karakter selain “m”,”a” dan “n” nilai pergeseran sebesar panjang string yaitu 7 karakter.

Cara Menghitung Tabel Match Heuristic :
contoh string : manaman
Panjang : 7 karakter
Tabel Match Heuristic

posisi :	1	2	3	4	5	6	7
string :	m	a	n	a	m	a	n
pergeseran (MH) :	4	4	4	4	7	7	1

Langkah-langkah perhitungannya adalah sebagai berikut :

- jika karakter pada posisi 7 bukan “n” maka geser 1 posisi, berlaku untuk semua string yang dicari.
- jika karakter “n” sudah cocok, tetapi karakter sebelum “n” bukan “a” maka geser sebanyak 7 posisi, sehingga posisi string melewati teks.karena sudah pasti “manambn” bukan “manaman”
- jika karakter “an” sudah cocok, tetapi karakter sebelum “an” bukan “m” maka geser sebanyak 7 posisi, sehingga posisi string melewati teks.karena sudah pasti “manaban” bukan “manaman”
- jika karakter “man” sudah cocok, tetapi karakter sebelum “man” bukan “a” maka geser sebanyak 4 posisi, sehingga posisi string berada / bersesuaian dengan akhiran “man” yang sudah ditemukan sebelumnya. karena bisa saja akhiran “man” yang sudah ditemukan sebelumnya merupakan awalan dari string “manaman” yang berikutnya.
- jika karakter “aman” sudah cocok, tetapi karakter sebelum “aman” bukan “n” maka geser sebanyak 4 posisi, sehingga posisi string berada / bersesuaian dengan akhiran “man” yang sudah ditemukan sebelumnya, karena bisa saja akhiran “man” yang sudah ditemukan sebelumnya merupakan awalan dari string “manaman” yang berikutnya.
- selanjutnya sama, pergseran paling mungkin dan aman dalam tabel Match Heuristic adalah pergeseran sebanyak 4 posisi.

B. Pendeteksian Plagiatisme dengan Algortima Boyer Moore

Pada pendeteksian ini diambil contoh kasus pendeteksian pada soruce code. Pada prinsipnya pendeteksian plagiatisme ini menggunakan penghitungan jumlah prosedur atau fungsi, perulangan, if-else dan variabel dengan pembobotan sederhana. Aplikasi melakukan pen-scan-an kata kunci-kata kunci tertentu (menggunakan Boyer Moore) untuk menentukan bagian yang cocok atau memenuhi kondisi tertentu.

Aplikasi akan mendeteksi kata kunci yang menyatakan pendeklarasian sebuah prosedur atau fungsi (syntax disesuaikan bahasa yang digunakan). Kemudian menghitung nilai perulangan (loop), if, dan variabel dengan menghitung jumlah masing-masing tipe alur program. Setelah melakukan penghitungan untuk salah satu prosedur/ fungsi kemudian aplikasi akan lanjut menghitung untuk seluruh prosedur yang terdapat pada kedua source code program dan menyimpannya masing-masing ke dalam sebuah array.Kemudian melakukan pembandingan elemen array dari source yang dicurigai malakukan plagiatisme dengan source yang asli atau sudah ada sebelumnya. Setiap elemen akan dicari kesamaan jumlah elemen-elementnya dengan source asli, jika didapati ada prosedur yang sama maka ia akan mencatat sebagai suatu kesamaan. Setelah selesai maka hasil akhir kemungkinan tingkat kesamaan source code dapat diatampilkan dalam bentuk prosentase yang dihitung dari jumlah kesamaan dibagi total prosedur source yang dicurigai.

Contohnya sebagai berikut :

Elemen I array I (objek yang diteliti): jumlah loop = 4, jumlah if-else = 7;

Elemen ke-N array II (pembanding / dokumen asli):jumlah loop = 4, jumlah if-else = 7.

Hal tersebut di atas dihitung sebagai sebuah kesamaan. Algoritma ini akan melakukan pencarian masing-masing elemen array objek dengan pembanding dan jika telah selesai akan dapat dihitung kemungkinan plagiatisme yang terjadi.

Tingkat plagiatisme dihitung dengan rumus :

Jumlah kesamaan / jumlah elemen objek X 100% (1)

Misal dari sepuluh elemen dalam array objek yang diteliti terdapat 8 kesamaan dengan elemen pembanding, maka didapatkan nilai 80%. Nilai ini menggambarkan tingkat kesamaan sebuah source dengan source lain. Batasan penentuan sebuah sorce merupakan plagiatisme atau bukan sangat relatif dan ditentukan sesuai kebutuhan

pemakai.

III. ALGORITMA RABIN-KARP

A. Hashing

Hashing adalah suatu cara untuk mentransformasi sebuah string menjadi suatu nilai yang unik dengan panjang tertentu (fixed-length) yang berfungsi sebagai penanda string tersebut. Fungsi untuk menghasilkan nilai ini disebut fungsi hash, sedangkan nilai yang dihasilkan disebut nilai hash. Contoh sederhana hashing adalah:

```
Firdaus, Hari   Munir, Rinaldi
   Rabin, Michael      Karp, Richard
```

menjadi

```
7864 Firdaus,Hari 9802 Munir, Rinaldi
   1990 Rabin, Michael     8822 Karp,
Richard
```

Contoh di atas adalah pengunaan hashing dalam pencarian pada database. Apabila tidak di-hash, pencarian akan dilakukan karakter per karakter pada nama-nama yang panjangnya bervariasi dan ada 26 kemungkinan pada setiap karakter. Namun pencarian akan menjadi lebih mangkus setelah di-hash karena hanya akan membandingkan empat digit angka dengan cuma 10 kemungkinan setiap angka.

Nilai hash pada umumnya digambarkan sebagai fingerprint yaitu suatu string pendek yang terdiri atas huruf dan angka yang terlihat acak (data biner yang ditulis dalam heksadesimal).

B. Prinsip Kerja Algoritma Rabin Karp

Pada dasarnya, algoritma Rabin-Karp akan membandingkan nilai hash dari string masukan dan substring pada teks. Apabila sama, maka akan dilakukan perbandingan sekali lagi terhadap karakter-karakternya. Apabila tidak sama, maka substring akan bergeser ke kanan. Kunci utama performa algoritma ini adalah perhitungan yang efisien terhadapat nilai hash substring pada saat penggeseran dilakukan.

Dari hasil perhitungan, kompleksitas waktu yang dibutuhkan adalah O(m+n) dengan m adalah panjang string masukan dan n adalah jumlah looping yang dilakukan untuk menemukan solusi. Hasil ini jauh lebih mangkus daripada kompleksitas waktu yang didapat menggunakan algoritma brute-force yaitu O(mn).

Secara garis besar, algoritma Rabin-Karp dapat dijelaskan dengan pseudocode berikut:

```
function RabinKarp (input s: string[1..m], teks: string[1..n]) → boolean
{
 Melakukan pencarian string s pada string teks dengan algoritma Rabin-Karp
}

Deklarasi
i : integer
ketemu = boolean

Algoritma:
ketemu ← false
hs ← hash(s[1..m])
for i ← 1 to n-m+1 do
   hsub ← hash(teks[1..i+m-1])
   if hsub = hs then
          if teks[i..i+m-1] = s then
          ketemu ← true
   else
       hsub ← hash(teks[i+1..i+m])
endfor
return ketemu
```

C. Pendeteksian Plagiatisme dengan Algoritma Rabin- Karp

Pada pendeteksian plagiatisme kali ini digunakan contoh kasus pendeteksian pada dokumen. Algoritma Rabin-Karp digunakan dalam mendeteksi plagiat sebab memungkinkan untuk mencari pola tulisan yang didapat dari substring-substring pada sebuah teks dalam dokumen, di mana algoritma pencarian string tunggal sangat tidak efisien dan praktis. Yang digunakan tentunya adalah varian algoritma Rabin-Karp untuk pencarian berpola banyak. Contoh perbandingan sebuah dokumen mahasiswa yang merupakan hasil plagiat dengan dokumen aslinya dari suatu website.

Tabel 1 Dokumen mahasiswa hasil plagiat dan dokumen asli dari website

Dokumen yang diteliti
Even if none of these things happen to you, youd be surprised at what you might find lurking on your computer!
What are malware, adware, and spyware? Malware is a short for malicious software and is a catch all phrase to describe any software that is designed to damage a computer.
Dokumen asli dari http://www.jellico.com/spyware.html
Even if none of these things happen to you, you'ld be surprised at what you might find lurking on your computer!
What is it? Malware (short for "malicious software") is any software developed for the purpose of doing harm to a computer.

Dari tabel 1, sekilas terlihat bahwa mahasiswa tidak melakukan plagiat. Namun, bagi yang mengerti cara kerja algoritma Rabin-Karp dalam mencari string berpola banyak, hal plagiat dapat dengan jelas diketahui.

Pertama-tama adalah melakukan filtering dengan menghilangkan beberapa tanda baca yang tidak penting. Proses ini juga dilakukan terhadap dokumen asli. Berikut contoh filtering (hanya diambil sebagian kecil dari keseluruhan teks).

```
Even if none of → evenifnone
What is it? → whatisit
```

Dari hasil filtering, kata-kata yang akan dijadikan string masukan diambil dengan melakukan pemisahan sebanyak k, misalnya, dan dicari seluruh kemungkinan yang mungkin dibentuk dari setiap kata-kata, sehingga didapat pecahan kata-kata. Hal ini disebut dengan k-gram. Berikut contoh k-gram dengan k = 4.

```
even veni enif nifn ifno fnon none
what hati atis tisi isit
```

Kemudian gunakan hasil ini sebagai masukan himpunan string dari algoritma RabinKarpSet seperti yang sudah dijelaskan pada bab sebelumnya. Diperoleh masukan algoritma RabinKarpSet-nya seperti ini:

```
teks:
evenifnoneofthesethings...computer
set s: {even, veni, enif, nifn,
...,uter}
m: 4
```

Selanjutnya, sesuai dengan algoritma, lakukan hashing ke seluruh pecahan string pada set s. Fungsi hash yang diberikan inilah yang merupakan kunci dalam menemukan pola kalimat pada teks, sehingga pastikan agar fungsi hash memadai untuk setiap string pada k-gram yang dihasilkan. Berikut contoh hashing untuk string-string pada set s (agar memudahkan, nilai hash string yang diberikan acak dan berharga kecil).

```
12 22 13 15 35 26 18
```

Dengan cara pengulangan iteratif sampai mencapai akhir teks, nilai hash string-string pada set s dicocokkan dengan nilai hash penggalan string sepanjang 4 karakter pada teks (benar apabila nilai hash penggalan teks asli terdapat pada himpunan nilai hash string-string masukan). Kita sebut saja hs bagi himpunan nilai hash untuk set s.

```
hash(even) ∈ hs hash(veni) ∈ hs
hash(enif) ∈ hs hash(nifn) ∈ hs
hash(ifno) ∈ hs hash(fnon) ∈ hs
                ...
hash(youl) !∈ hs   (bukan elemen hs)
hash(ould) !∈ hs   (bukan elemen hs)
hash(uldb) !∈ hs   (bukan elemen hs)
hash(ldbe) !∈ hs   (bukan elemen hs)
hash(dbes) !∈ hs   (bukan elemen hs)
                ...
hash(pute) ∈ hs hash(uter) ∈ hs
```

Pada setiap penggalan teks asli sepanjang 4 karakter, dilakukan hashing dengan fungsi hash yang sama seperti yang dilakukan terhadap string-string set s. Pencarian nilai hash penggalan teks asli pada hs dapat dilakukan dengan algoritma pencarian biasa, sesuai kehendak. Hasil akhir yang didapat untuk perbandingan dokumen pada Tabel 1 adalah:

Tabel 2 Hasil perbandingan dokumen mahasiswa dan dokumen asli dari website

Dokumen yang diteliti
Even if none of these things happen to you, youd be surprised at what you might find lurking on your computer!
What are malware, adware, and spyware? Malware is a short for malicious software and is a catch all phrase to describe any software that is designed to damage a computer.
Dokumen asli dari http://www.jellico.com/spyware.html
Even if none of these things happen to you, you'ld be surprised at what you might find lurking on your computer!
What is it? Malware (short for "malicious software") is any software developed for the purpose of doing harm to a computer.

Dari hasil perbandingan kedua potongan dokumen, terlihat sangat jelas bahwa mahasiswa mengopi sebagian besar dokumen asli. Pola tulisan mahasiswa menunjukkan kemiripan dengan pola penulisan pada dokumen asli, sehingga dapat disimpulkan ia telah melakukan tindakan plagiat. Sekali lagi di sini penentuan plagiatisme sangatlah relatif tergantung pengguna, dalam hal ini kesamaan pola penulisan sudah digolongkan ke dalam plagiatisme.

IV. KESIMPULAN

Terdapat cara yang lebih efektif dalam mendeteksi praktek plagiatisme yaitu dengan memanfaatkan Algoritma Boyer-Moore dan Algoritma Rabin-Karp.

Algoritma Boyer Moore digunakan untuk melakukan pencocokan string dan melakukan penghitungan kesamaan. Sedangkan kemiripan pola antar dua buah dokumen dapat dideteksi dengan menerapkan prinsip kerja algoritma pencarian string Rabin Karp.

Strategi pendeteksian di atas dapat dijadikan patokan dalam menentukan apakah terjadi praktek plagiatisme disesuaikan dengan pandangan penggunanya, walaupun dapat diantisipasi jika metode pendeteksian yang digunakan diketahui oleh orang yang akan melakukan praktek plagiatisme tersebut. Namun setidaknya hal ini dapat digunakan untuk melakukan pencegahan terhadap upaya plagiatisme tersebut.

REFERENCES

[1] Munir,Rinaldi. Diktat Kuliah IF3051 Strategi Algoritma.Prodi teknik Informatika. STEI ITB, 2009.

[2] http://edwardgr.wordpress.com/2009/01/06/algoritma-boyer-moore/ diakses tanggal 7 Desember 2010.

[3] http://informatika.org/~rinaldi

[4] http://en.wikipedia.org/wiki/Rabin–Karp_string_search_algorithm

[5] William B. Frakes dan Ricardo Baeza-Yates, Information Retrieval Data Structures & Algorithms (New Jersey : Prentice-Hall, Inc. 1992), hal. 224.

[6] http://id.wikipedia.org/wiki/Algoritma_Boyer-Moore. diakses 7 Desember 2010.

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 8 November 2010

Arif Prasetiya