Penggunaan String Matching Dalam Mencari Kata Dalam Permainan Mencari Kata Dari Sebuah Matriks Huruf

Luthfi Kurniawan 13514102[1]
Program Studi Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
[1]*luthkur@s.itb.ac.id*

***Abstract*—Makalah ini menganalisa mengenai penggunaan variasi algoritma string matching untuk memenukan kata dalam matriks huruf dengan metode perbandingan waktu runtime dari tiap variasi algoritma. Penulis melakukan pengadaptasian variasi algoritma String matching sehingga bisa dijalakan di matriks huruf. Adaptasi dilakukan dengan pembagian matriks menjadi array biasa yang dibagi menurut orientasi horizontal, vertikal dan diagonal. Untuk varian algoritma String Matching yang paling bagus performanya adalah algoritma Boyer-Moore.**

Keywords—String; Matching; Brute force; matriks huruf; KMP; boyer-moore; bahasa alami;

I. PENDAHULUAN

String Matching adalah suatu proses pencocokan sebuah string yang panjang dengan sebuah string yang lain. String Matching banyak digunakan untuk menemukan pola-pola dari sebuah data yang direpresentasikan sebagai sebuah string.

Biasanya String Matching dapat digunakan untuk menemukan pola di sebuah string yang normal. Tetapi String Matching dapat digunakan juga untuk menyesaikan sebuah permainan mencari kata yang kata-katanya tersembunyi dalam sebuah matriks huruf, matriks huruf ini membentuk sebuah string yang bisa dibaca secara horizontal atau vertikal. Umumnya jika dibaca secara biasa yaitu dari kiri ke kanan kata-kata tidak bisa ditemukan. Di makalah ini akan dibahas bagaimana caranya untuk menemukan kata-kata yang tersembunyi di dalam matriks huruf dapat ditemukan. Penulis akan menggunakan Brute Force, KMP, dan boyer moore untuk menyelesaikan permasalahan.

String Matching dapat dikembangkan menjadi banyak variasi untuk penyelesaian berbagai masalah tertentu. Di makalah ini akan dibahas masalah dan penyelesaian pencarian kata dalam matriks huruf yang bisa dicari secara horizontal, vertikal maupun diagonal. Penulis akan menggunakan Algoritma String Matching Brute Force, KMP, Dan Boyer-Moore yang sudah dimodifikasi untuk menyelesaikan masalah. Penulis akan menganalisa penerapan dan perfoma masing masing versi Algortima String Matching.

Matriks huruf yang digunakan dalam makalah ini akan berukuran kecil untuk mempersingkat waktu penerapan dan analisa, kata-kata yang digunakan adalah kata-kata umum dalam bahasa indonesia yang biasa diguanakan untuk permainan mencari kata dalam sebuah matriks huruf.

II. LANDASAN TEORI

A. String Matching

Pencarian String Matching dilakukan untuk menemukan sebuah rangkaian string dalam string yang lain. Dengan String yang dicari memiliki panjang string yang lebih pendek daripada string yang diolah isinya untuk ditemukan pola string yang menjadi bahan pencarian.

String yang menjadi pola pencarian disebut String Pattern. Sementara String yang menjadi bahan Pencarian disebut String Text. Panjang String Pattern dilambakan dengan huruf m. Sedangkan panjang String teks dilambangkan oleh huruf n. Syarat yang harus dipenuhi oleh String pattern dan string teks adalah ($m < n$) yang berarti panjang String Teks harus lebih panjang daripada String Pattern agar algoritma String Matching bisa bekerja untuk mencari pattern di dalam teks.

Diasumsikan String Teks sudah diload dalam memori, sehingga tidak perlu dilakukan operasi untuk meload String Teks ke dalam Memori dalam algoritma String Matching. Jika pattern ditemukan berkali-kali di dalam teks maka yang akan dikeluarkan hanyalah indeks dimana pattern pertama kali ditemukan di dalam teks. Untuk membedakan dengan pattern yang menjadi masukan algoritma String Matching, maka pattern yang ditemukan dalam akan teks akan dinamakan sebagai Target.

String-String teks dan pattern terdiri dari anggota-anggota himpunan sebuah alphabet yang menjadi huruf-huruf penyusun dari teks dan pattern. Alphabet bisa terdiri dari huruf huruf latin, kode biner (0 dan 1) ataupun kode kode penyusun dari DNA.

String Matching banyak diaplikasin di banyak bidang bukan hanya untuk mencari sebuah kata khusus dari sebuah rangkain huruf-huruf.

String Matching memilki berbagai versi pengembangan, dari sekadar bruteforce hingga berbagai versi yang memakai

berbagai macam cara preprocessing untuk menghindari waktu runtime seburuk brute force yang melakukan matching secara naif.

B. Brute Force

Brute Force dilakukan dengan asumsi bahwa teks berada di dalam array T[1..n] dan pattern berada di dalam array P[1..m], maka algoritma brute force String Matching adalah sebagai berikut :

1. Mula-Mula pattern dicocokan pada awal teks T.
2. Dengan bergerak ke kiri ke kana, membandingkan setiap setiap karaktek di dalam pattern P dengan karakter yang bersesuaian di dalam teks T sampai :
    a. Semua karakter yang dibandingkan cocok atau sama (pencarian berhasil)
    b. Atau Dijumpai sebuah ketidakcocokan karakter(pencarian belum berhasil).
3. Bila pattern P belum ditemukan dan iterasi di array Teks belum selesai maka geser mulainya pencocokan pattern P satu karakter ke kanan di teks T dan ulangi langkah 2.

Kompleksitas algortima pencocokan string secara brute force adalah O(m) untuk kasus terbaik untuk kasus terburuk waktu runtime akan menjadi O(mn) karena pencocokan karakter akan dilakukan sebanyak m kali dengan n kali pergesaran. Algortima brute force tidak banyak dipakai di dunia nyata karena sangat tidak efisien dalam pencariannya. Pergesaran karakter yang dilakukan hanya sekali jika perbandingan pattern dengan suatu lokasi gagal dilakukan walau di bagian teks itu benar-benar tidak ada kecocokan secara suffix atau prefix. Algoritma brute force akan dilakukan dengan cepat jika alphabet dari teks besar seperti alfabet latin atau angka-angka. Brute force akan jadi lebih lambat ketika jumlah alfabetnya kecil seperti kode biner di file biner atau di file gambar.

C. KMP

Algoritma KMP dikembangkan oleh D. E Knuth, bersama-sama dengan dengan J.H Morris dan V. R. Pratt. Algoritma ini membedakan diri dengan brute force melalui beberapa hal berikut :

1. Sebelum dilakukan pencarian Algoritma KMP akan melakukan Preprosses terhadap Pattern P yang akan melakukan komputasi fungsi failure yang akan menandakan berapa pergeseran maksimal yang bisa dilakukan oleh pattern ketika gagal pada suatu lokasi karakter di dalam array patter P. Yang dilakukan di fungsi failure dalam menandakan panjang prefiks tertentu p yang merupakan juga suffiks dari pattern P. Fungsi pinggiran hanya tergantung pada karakter-karakter di p bukan pada karakter-karakter di teks sehingga preprosess bisa dilakukan sebelum pencarian string dilakukan.
2. Pergesaran pattern ketika terjadi ketidakcocokan karakter akan dilakukan berdasarkan nilai yang ada di dalam array yang indeksnya menandakan lokasi di mana kegagalan pencocokan terjadi di dalam pattern. Perhitungan besar pergeseran adalah besar jumlah panjang pattern yang sudah dibandingkan dikurangi dengan nilai yang ada di array fungsi failure.

Kmp melakukan pergesaran dengan pintar untuk menghindari perbandingan yang sudah pasti salah, pencatatan nilai fungsi failure adalah untuk mencatat awalan dan akhiran yang sama sehingga kita bisa mementukan apakah sebagian dari pattern yang sudah cocok di satu sekuens perbandingan itu apakah adalah awalan dari sekuens perbandingan yang cocok dengan pattern yang dibandingkan.

Waktu runtime dari algoritma KMP adalah O(n) untuk menghitung fungsi failure dan O(n) untuk bagian pencarian string. Sehingga totalnya algoritma KMP memilki kompleksitas waktu sebesar O(m+n), kompleksitas KMP jauh lebih baik daripada kompleksitas waktu brute force.

Keunggulan KMP dalah kmp tidak pernah perlu bergrak balik di dalam sebuah input teks, hal ini sangat bagus untuk memproses file yang sangat besar yang distream-kan dari sebuah jaringan atau device eksternal.

D. Boyer-Moore

Boyer-Moore adalah algortima yang dianggap oleh kalangan praktisi sebagai algoritma String Matching yang paling efisien di penggunaan string matching pada umumnya. Versi yang cukup simpel dari algoritma ini sering diimplementasikan di dalam editor teks untuk fungsi mencari dan fungsi mengganti sebuah pola dalam teks dengan pola lain.

Algoritma ini melakukan dua hal yang menyebabkan dia menjadi algoritma yang lebih efiseien. Yaitu dia menggunakan teknik looking glass dan juga character jump. Kedua teknik ini saling membantu membuat Boyer-moore menjadi algoritma yang sangat efisien untuk melakukan penncarian string di dalam teks.

1. Looking glass adalah teknik yang melakukan perbandingan string secara terbalik berlawanan dengan KMP dan brute force yang membandingkan karakter dari pattern dan teks dari awal pattern.
2. Character jump dilakukan dengan cara jika terjadi ketidakcocokan karakter di satu posisi maka akan dilakukan shift yang terbagi menjadi tiga kasus yaitu :
    a. Jika terjadi mismatch dan karakter x yang ada di teks ada di dalam pattern yang indeksnya lebih kecil dari indeks pattern yang sedang dibandingkan maka dilakukan pergesaran ke kanan hingga indeks karakter x yang ada di teks akan

dibandingkan dnegan karakter x yang ada di pattern.

b. Jika terjadi mismatch dan karakter x yang ada di teks ada di dalam pattern tapi kemunculan terakhir dari x relatif dari akhir pattern ada di indeks yang lebih besar daripada indeks yang sedang dibandingkan maka indeks i yang di teks digeser sebesar 1, dan indeks j pada pattern di mulai lagi dari akhir pattern sehingga di sekuens pencocokan selanjutnya i dibandingkan dengan j.

c. Jika karakter x yang mengakibatkan ketidakcocokan tidak ada di dalam pattern maka yang dilakukan adalah pergeseran di mana indeks i di teks digeser sebesar 1 dan perbandingan P[1] dengan karakter yang di indeks tersebut, maka perbandingan sekuens karakter yang selanjutnya dilakukan dengan membandingkan karakter Pattern dari belakangan dengan teks dimulai indeks i+panjang pattern.

3. Agar Character jump bisa dilakukan dengan efisien dan bisa berjalan dengan baik maka dilakukan fungsi lastoccurance yang menandai tiap alfabet yang ada di dalam pattern dengan indeks di mana alafabet itu terakhir muncul di pattern. Indeks yang ditandai adalah indeks dari sebelah kiri alias indeks terbesar karakter itu muncul, jika tidak ada karakter/alfabet tersebut di dalam pattern maka diberikan nilai sebesar -1. Nilai-nilai ini disimpan sebagai Array yang berkorespodensi dengan alfabet yang digunakan dalam teks dan pattern. Fungsi lastOccurance dijalankan ketika pattern pertama kali dibca seperti yang dilakukan oleh Algoritma KMP.

Boyer-Moore memiliki performa yang sangat baik ketika mencari kata di dalam teks yang merupakan bahasa natural seperti bahasa inggris. Perfomanya semakin baik jika jumlah karakter di dalam himpunan alfabet yang muncul diteks semakin banyak, hal ini dikarenakan jika ditemukan karakter yang ada di teks tapi tidak ada di pattern maka persgeseran dilakukan dengan sangat besar sebesar panjang teks. Sehingga efisiensi pencarian pattern di tteks sangat tinggi. Walau hal ini tidak berlaku jika alfabet yang ada di teks dan pattern kecil seperti mencari di string biner yang menyusun file biner atau file gambar dan sebagainya.

Kompleksitas dari boyer-moore adalah O(mn+A) untuk kasus terburuk untuk kasus rata-rat kompleksitasnya O(n / m) untuk kasus terbaik.

E. Matriks Huruf

Matriks huruf adalah array 2 dimensi yang menampung huruf-huruf yang memiliki pengaturan yang awalnya terkesan acak. Walau sebenarnya matriks ini menyimpan sebuah kata-kata yang bisa ditemukan jika dibaca horizontal, vertikal, diagonal miring kiri atau diagonal miring kanan. Umumnya matriks ini dipakai untuk teka-teki/permainan anak anak yang sedang belajar bahasa sehingga melatih kemampuan mencari pola kata-kata dalam bahasa tersebut. Dalam makalah ini akan digunakan matriks berukuran 13x13 yang karakter-karakternya adalah karakter bahasa latin yang menyusun kata-kata dalam bahasa indonesia.

III. PENERAPAN STRING MATCHING

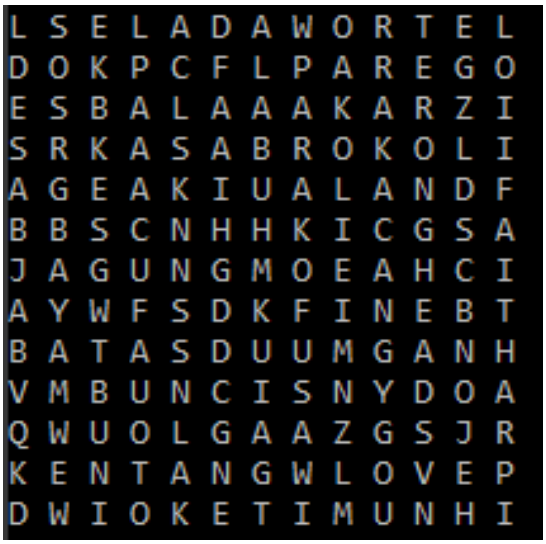
A. Penerapan Secara Umum

Penerapan String Matching secara umum adalah sebagai berikut :

1. Untuk Matriks huruf yang digunakan dalam makalah ini berukuran 13x13 dan digunakan array 2 dimensi yang menampung variabel bertiper char.
2. Bahasa pemprograman yang digunakan dalam implementasi Algoritma String matching adalah bahasa Java.
3. Untuk input data ke dalam program String Matching dilakukan di waktu runtime.
4. Untuk brute force, KMP, dan boyer-moore dicompile secara masing-masing di dalam class/program yang berbeda-beda.
5. Adaptasi dari String Matching yang biasa adalah di dalam penyelesaian pencarian pattern di dalam matriks huruf dibagi menjadi 4 tahap yaitu :
    a. Pencarian secara horizontal, yaitu tiap baris menjadi sebuah array karakter teks biasa yang bisa dilakukan string matching.
    b. Pencarian secara vertikal, yaitu tiap kolom menjadi array karakter teks yang bisa dilakukan string matching yang biasa.
    c. Pencarian secara diagonal miring kiri, perbandingan yang dilakukan secara diagonal miring kiri yang dimulai dari pojok kiri bawah, perbandingan string baru mulai dilakukan jika panjang pattern yang dicari lebih kecil dari panjang array karakter secara diagonal.
    d. Pencarian secra diagonal miring kanan, perbandingan yang dilakukan secara miring kanan, perbandingan dimulai dari pojok kanan atas. Sama seperti perbandingan diagonal miring kiri, Algoritma String matching baru dijalankan jika panjang diagonal

memilki panjang lebih besar daripada panjang pattern ayang akan dibandingkan.

6. Output yang akan dihasilkan program string matching adalah berupa :
    a. Orientasi dari kata yang ditemukan apakah orientasi secara horizontal, vertikal, diagonal miring kanan, atau diagonal miring kanan.
    b. Posisi yang dikeluarkan adalah indeks 2 dari array 2 dimensi dari teks.
7. Matriks Huruf yang akan digunakan dalam penelitian makalah ini adalah sebagai berikut :



Gambar 1 : Matriks Huruf yang digunakan

B. Kata-Kata Yang Dicari

Kata-kata yang digunakan dalam penerapan String Matching untuk mencari kata di dalam matriks huruf adalah sebagai berikut :

1. BROKOLI
2. PARE
3. WORTEL
4. KANGKUNG
5. LABU
6. KENTANG
7. BAYAM
8. BUNCIS
9. LOBAK
10. CABAI
11. KACANG
12. KETIMUN
13. TERONG
14. SAWI
15. JAGUNG
16. BAWANG
17. KOL
18. SELADA

Kata-kata yang diatas terdapat di dalam Matriks Huruf yang digunakan dalam percobaan yang dirangkum dalam makalah ini.

C. Brute Fore

Secara implementasi Algortima Brute Force masih seperti algoritma brute force yang biasa, di mana perbandingan dilakukan dari awal pattern ke akhir pattern. Jika ditemukan ketidakcocokan maka pattern digeser sebesar 1 indeks di array T teks dan perbandingan string dilakukan lagi dari awal pattern diulang lagi sampai ditemukan lokasi pattern di teks yang semua cocok dengan pattern yang dicari. Sama seperti yang disudah disebukan di implementasi secara umum. Pencarian String dilakukan secara horizontal, vertikal, diagonal miring kiri, dan diagonal miring kanan. Perbandingan dilakukan seperti biasa di array-array teks yang sudah dibagi-bagi ke bagian-bagian kecil yang dibagi secar horizontal, vertikal, diagonal miring kiri dan diagonal miring kanan.

D. KMP

Secara implementasi Algoritma KMP masih seperti algoritma KMP yang seperti biasa. Karena KMP masih seperti brute force yang melakukan pergeseran ketika terjadi kegagalan pencocokan pattern. KMP melakukan pergesarn secara pintar melalui fungsi failure, fungsi failure yang digunakan dalam KMP penyelesaian pencarian kata di matriks huruf masih sama seperti KMP biasa karena fungsi failure hanya tergantung dari pattern kata yang ingin dicari sehingga hanya tegantung dengan pattern kata yang ingin dicari oleh algoritma KMP. Dalam pencarian String untuk menemukan kata dalam Matriks huruf pembagiannya pola algoritma sama seperti yang sudah disebutkan di implementasi secara umum akan dilakukan pencarian secara horizontal, vertikal, diagonal miring kanan, dan diagonal miring kiri. Sama seperti yang disebutkan di di implementasi secara umum, algoritma KMP akan dijalankan di array-array 1 dimensi hasil pembagian matriks huruf menjadi array-array yang dibaca secara horizotal, vertikal, diagonal miring kanan, dan diagonal miring kanan di dalam matriks huruf. Di bagaian array yang dihasilkan oleh diagonal matriks huruf algoritma baru dijalankan ketika panjang array yang dihasilkan lebih panjang daripada pattern kata yang ingin dicari.

E. Boyer-Moore

Secara Implementasi Algoritma Boyer-Moore masih seperti algortima Boyer-Moore yang digunakan untuk mencari pattern P dalam array teks T yang 1 dimensi. Sama seperti KMP preprosses yang digunakan oleh Boyer-Moore dilakukan

pada pattern saja. Fungsi lastOccurance milik Boyer-Moore dilakukan terhadap kata yang ingin dicari dengan himpunan alfabet latin biasa bahasa indonesia. Sama seperti yang disebutkan di di implementasi secara umum algoritma Boyer-Moore akan dijalankan di array-array 1 dimensi hasil pembagian matriks huruf menjadi array-array yang dibaca secara horizotal, vertikal, diagonal miring kanan, dan diagonal miring kanan di dalam matriks huruf. Di bagaian array yang dihasilkan oleh diagonal matriks huruf algoritma baru dijalankan ketika panjang array yang dihasilkan lebih panjang daripada pattern kata yang ingin dicari.

IV. Analisa Performa

A. *Analisa Secara Umum*

Performa Secara umum algoritma-algoritma yang digunakan secara umum bisa menghasilkan lokasi-lokasi kata-kata yang ingin dicari dalam matriks huruf. Waktu runtime masing-masing algoritma sesuai dengan waktu yang biasa didapatkan dengan algoritma tersebut.

Untuk waktu penyelesaian matriks waktu yang dibutuhkan tergantung dimana kata berada dalam orientasi apa, jika dalam orientasi horizontal maka eksekusi algoritma akan lebih cepat karena horizontal adalah orientasi yang pertama kali dicek. Baru ke orientasi vertikal, orientasi diagonal miring kiri dan diagonal miring kanan. Jika kata ditemukan di orientasi diagonal miring kanan maka waktu yang dibutuhkan akan lebih lama karena algoritma akan melakukan iterasi pencarian di orientasi horizontal, vertikal, dan diagonal miring kiri terlebih dahulu sehingga baru akan mengolah orientasi diagonal miring kanan di akhir.

Pada matriks huruf ini penyelesaian pencarian kata menghasilkan data orientasi kata yang ditemukan di bagian apa. Jika horizontal maka akan mengeluarkan output jika kata ditemukan secara horizontal dan ditemukan di kata awal yang mana, harus diketahui jika orientasi horizontal maka pattern dapat dibaca di matriks secara horizontal. Jika ditemukan di vertikal maka kata harus dibaca secara vertikal di dalam matriks huruf. Sedangkan jika ditemukan secara diagonal maka kata bisa dibaca secara diagonal di matriks.

Kasus terbaik untuk pencarian kata di matriks huruf sebenarnya relatif dengan perurutan iterasi orientasi yang digunakan seperti apa, jika kebanyakan kata yang ingin dicari berada di dalam orientasi yang pertama maka performa String matching untuk matriks huruf akan sangat bagus. Tetapi jika Kata-kata yang ingin dicari kebanyakan ada di dalam orientasi yang diiterasi yang terakhir maka performa algoritma string matching akan sangat buruk.

Untuk perbedaan performa untuk masing masing jenis algoritma String matching untuk mencari kata dalam matriks huruf yang paling baik adalah algoritma boyer-mooore karena algoritma ini sangat baik untuk mencari kata-kata dalam bahasa alami seperti bahasa indonesia. KMP cukup baik untuk menyelesaikan kata-kata yang dicari jika kata yang digunakan adalah kata-kata yang digunakan memilkik prefiks dan suffiks yang tidak berulang-ulang. Performa algoritma brute force adalah yang paling buruk karena tidak ada optimisasi melalu preprosses yang meminimalkan jumlah perbandingan karakter dengan pergeseran karakter yang lebih pintar.

B. *Brute Force*

Peforma algoritma brute force adalah yang paling buruk karena hanya melakukan pergesaran satu kali untuk pergantian sekuens perbandingan yang disebabkan oleh kegagalan pencocokan karakter antara pattern dan array teks. Jumlah perbandingan karakter yang dilakukan oleh brute force jauh lebih banyak jika dibandingkan dengan perbandingan karakter yang dilakukan oleh algoritma KMP dan Boyer-Moore.

Pada brute force perbandingan yang dilakukan sangat simpel, jika terjadi kegagalan pencocokan maka digeser sekali. Pada matriks huruf kekompleksitasnya sama kompleksnya dengan kasus terburuk di array teks biasa. Pada matriks ini pada kasus terburuk adalah jika dilakukan String Matching di semua baris, kolom dan diagonal-diagonal yang ada pada Matriks huruf.

C. *KMP*

Pada algoritma KMP memiliki performa yang cukup bagus karena kata-kata yang digunakan dan teks yang ada di dalam matriks huruf adalah kata-kata bahasa alami yang merupakan bahasa indonesia yang memilki himpunan alfabet yang besar. Himpunan alfabet yang terdiri atas huruf-huruf latin. Untuk KMP performa akan lebih cepat jika kata pattern P yang digunakan bukan kata-kata yang prefiksnya sama dengan suffiks dengan kemiripan yang cukup besar. Karena jika prefiks dan suffiksnya besar maka pergeseran karakter akan sangat minimal. Jika kata yang digunakan tidak berulang prefiks dan suffiksnya maka ketika terjadi ketidakcocokan maka pergeseran kata-kata akan besar.

Pada saat program dijalankan performa KMP cukup bagus.

D. *Boyer-Moore*

Pada saat dijalankan Algoritma Booyer-Moore memiliki performa yang paling bagus karena pergeseran yang dilakukan sangat cocok untuk mencari kata-kata bahasa alami seperti bahasa indonesia. Kata-kata yang dijalankan cocok dijalankan di pencarian kata dalam matriks huruf sehingga pergeseran yang dijalankan oleh algoritma Booyer-Moore efisien ketika terjadi mismatch pada saat sekuens perbandingan karakter.

V. Kesimpulan

Dapat disimpulkan jika algortima String Matching dapat diadaptasi menjadi algoritma yang bisa mencari kata-kata yang terkandung dalam matriks huruf. Pencarian dilakukan secara horizontal, vertikal, dan diagonal miring kiri dan diagonal miring kanan. Dengan membagi-bagi matriks huruf menjadi array 1 dimensi secara horizontal, vertikal dan diagonal, algortima String matching dapat dijalankan untuk mencari kata.

Untuk peforma yang paling baik adalah algoritma booyer-moore karena algoritma ini sangat cocok untuk mencari kata-kata yang merupakan bahasa alami seperti bahasa indonesia.

References

[1] Munir, Rinaldi. 2009. *Diktat Kuliah IF2211 Strategi Algoritma.* Bandung. Program Studi Teknik Informatika ITB

[2] Knuth-Morris-Pratt Algorithm. (n.d.). diakses 08 Mei, 2016, dari http://www.personal.kent.edu/~rmuhamma/Algorithms/MyAlgorithms/StringMatch/kuthMP.htm

[3] Boyer-Moore algorithm. (n.d.). Diakses 8 Mei, 2016, dari http://www-igm.univ-mlv.fr/~lecroq/string/node14.html

[4] A Simplified Boyer-Moore Algorithm. (n.d.). Diakses 8 Mei, 2016, dari http://www.mathcs.emory.edu/~cheung/Courses/323/Syllabus/Text/Matching-Boyer-Moore1.html

VI. 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, 07 Mei 2016

Luthfi Kurniawan (13514102)