Majalah Ilmiah INTI, Volume 12, Nomor 2, Mei 2017
ISSN 2339-210X

PERANCANGAN APLIKASI PENCARIAN KATA DENGAN KOMBINASI ALGORITMA KNUTH MORRIS PRATT DAN ALGORITMA BOYER MOORE

Makrina Meti Yana Daeli [1], Rivalri Kristianto Hondro [2]

[1] Mahasiswa Teknik Informatika STMIK Budi Darma
[2] Dosen Tetap STMIK Budi Darma
[1,2] Jl. Sisingamangaraja No. 338 Simpang Limun, Medan

ABSTRAK

Pada penelitian ini dibangun sebuah aplikasi yang bertujuan untuk mencari file text yang berasal dari komputer dan mengaplikasikannya dengan menggunakan algoritma pencocokan string sebagai salah satu cara untuk menemukan file teks yang terdapat pada komputer. Aplikasi ini dibangun berbasis desktop dengan menggunakan algoritma Pencocokan string Knuth-Morris-Pratt dan Boyer Moore sebagai algoritma dalam aplikasi pencarian file teks, dan dibangun dengan menggunakan bahasa pemrograman visual basic 2008, analisis perancangan sistem ini menggunakan Unified Modeling Language (UML).Dapat disimpulkan bahwa aplikasi ini dapat melakukan pencarian file dokumen yang terdapat dalam Komputer dengan menggunakan algoritma Knuth-Morris-Pratt. Hasil pencarian yang ditampilkan berupa file teks yang tersedia dalam Komputer dan informasi mengenai jumlah dari file teks yang tersedia dalam Komputer tersebut, serta menunjukkan bahwa algoritma Knuth-Morris-Pratt biasa digunakan dalam aplikasi pencarian file teks pada Komputer.

Kata Kunci: KMP, Boyer Moore, Aplikasi KMP, Aplikasi Boyer Moore.

I. PENDAHULUAN

Pencarian (*searching*) merupakan suatu pekerjaan yang sering dikerjakan dalam kehidupan sehari – hari. Ada kalanya kita mencari sesuatu dengan tujuan hanya untuk mengetahui apakah data tersebut ada dalam sekumpulan data atau tidak, sementara di lain waktu mungkin kita menginginkan posisi dari data yang dicari tersebut.

Dalam ilmu komputer terdapat bermacam – macam algoritma untuk metode pencarian (*searching*) maupun pencocokan *string*. Beberapa metode pencarian yang pernah dipelajari adalah metoda pencarian linier (*Linear / Sequential Search*), pencarian biner (*Binary Search*) pencarian interpolasi (*Interpolation Search*). Masing – masing algoritma memiliki prasyarat dan cara serta waktu pelaksanaan yang berbeda, sama halnya dengan algoritma pencocokan string yang banyak digunakan seperti *Exact String Matching, Pattern String Matching, Knuth Morris Pratt* dan *Bayer Moore* (Arief Syuhada, 2014).

Perbedaan yang signifikan terdapat pada algoritma pencocokan *string*, jika pada proses pencarian tidak perlu menerapkan algoritma pencocokan *string* untuk menguji data itu sesuai atau tidak cukup menggunakan logika IF, sedangkan pada algoritma pencocokan string secara *default* menggunakan konsep pencarian agar proses lebih baik dalam hal pencocokan *string*.

Algoritma Knuth-Morris-Pratt (KMP) adalah algoritma pencocokan *string* (*pattern*) dari kiri kekanan, sedangkan *Algoritma Bayer Moore* adalah salah satu algoritma untuk mencari suatu *string* di dalam teks,dan mencari *string* dengan melakukan perbandingan karakter paling kanan *string* yang dicari.

Penerapan *string Matching* pada fitur *Auto Correct* dan *Fitur Auto Text* di *smart Phones* yang mana pada kasus tersebut terdapat tiga algoritma pencocokkan *string* yang digunakan yaitu: *Algoritma Brute Force, Algoritma Boyer Moore*, dan *Algoritma Knuth-Morris-Pratt(KMP)*. Masalah yang dihadapi pada kasus tersebut adalah bagaimana Algoritma *Brute force*, Algoritma *Bayer Moore* dan *Algoritma Knuth-Morris-Pratt(KMP)* dapat diterapkan pada *Fitur Auto Correct* dan *Fitur Auto Text* dan ternyata ketiga Algoritma tersebut dapat diterapkan dengan baik.(Prahara Fandi,2012).Algoritma *Knuth Morris Pratt* dan *Boyer Moore* memiliki kelebihan masing-masing dan penulis mencoba menguji kelebihan kedua algoritma untuk melakukan pencarian pada komputer dengan memeriksa kata dari setiap *file text* yang ada didalam komputer.

II. LANDASAN TEORI

A. Algoritma Knuth Morris Pratt

Algoritma Knuth-Morris-Pratt dikembangkan oleh D. E. Knuth, bersama dengan J. H. Morris dan V. R. Pratt. *Algoritma Knuth-Morris-Pratt* merupakan pengembangan dari algoritma pencarian *string* sebelumnya, yaitu *algoritma Brute Force*. Algoritma *KMP* (*Knuth Morris Pratt*) merupakan algoritma yang digunakan untuk melakukan proses pencocokan *string*. Algoritma ini merupakan jenis *Exact String Matching Algorithm* yang merupakan pencocokan string secara tepat dengan susunan karakter dalam *string* yang dicocokkan memiliki jumlah maupun urutan karakter dalam string yang

Majalah Ilmiah INTI, Volume 12, Nomor 2, Mei 2017
ISSN 2339-210X

sama. Terdapat beberapa definisi pada algoritma *KMP* :

1. Misalkan *A* adalah alfabet dan $x = x1x2…xk$ adalah *string* yang panjangnya k yang dibentuk dari karakter-karakter di dalam alfabet *A*.
2. Awalan (*prefix*) dari *x* adalah upa-string (*substring*) *u* dengan $u = x1x2…xk – 1$, $k \in \{1, 2, …, k – 1\}$dengan kata lain, *x* diawali dengan *u*.
3. Akhiran (*suffix*) dari *x* adalah upa-string (*substring*) *u* dengan $u = xk – b\ xk – b + 1 …xk$, $k \in \{1, 2, …, k – 1\}$dengan kata lain, *x* di akhiri dengan *v*
4. Pinggiran (*border*) dari *x* adalah upa-string *r* sedemikian sehingga $r = x1x2…xk – 1$ dan $u = xk – b\ xk – b + 1 …xk$, $k \in \{1, 2, …, k –1\}$, dengan kata lain, pinggiran dari *x* adalahupa-string yang keduanya awalan dan juga akhiran sebenarnya dari *x*.

Fungsi Pinggiran b(j) didefinisikan sebagai ukuran awalan terpanjang dari P yang merupakan akhiran dari P[1..j]

B. Algoritma Boyer Moore

Algoritma *Boyer Moore* dipublikasikan oleh Robert S. Boyer,dan J. Strother Moore pada tahun 1977. Pencocokan karakter yang dilakukan menggunakan algoritma *Boyer Moore* dimulai dari string sebelah kanan pattern. Ide di balik algoritma ini ialah dengan memulai pencocokan karakter dari kanan, maka akan lebih banyak informasi yang didapat. (Rudi Setiawan, 2015).

Adapun langkah-langkah pencarian *string* pada *algoritma Boyer Moore* sebagai berikut :

1. Pertama diperlukan 2 tabel dengan pendekatan *Match Heuristic* (MH) dan *Occurrence Heuristic* (OH) untuk menentukan jumlah pergeseran yang akan dilakukan pada pattern (P) jika ditemukan karakter yang tidak cocok pada proses pencocokan terhadap karakter pada teks (S).
2. Jika pada proses pembandingan terdapat ketidakcocokan karakter antara karakter pada P dan S, maka pergeseran dilakukan dengan melihat kedua table dengan nilai pergerseran paling besar yang dipilih.
3. Kemungkinan penyelesaian dalam melakukan pergeseran pada P adalah jika pada pencocokan sebelumnya tidak ada karakter yang cocok maka pergeseran dilakukan dengan melihat nilai pergeseran pada tabel *occurrence heuristic.* Jika karakter yang sedang dibandingkan tidak terdapat pada P maka pergeseran dilakukan sebanyak jumlah karakter yang terdapat pada P, tetapi jika karakter yang tidak cocok tersebut terdapat pada *string* P, maka pergeseran dilakukan berdasarkan tabel.
4. Jika karakter pada teks yang sedang dibandingkan cocok dengan karakter pada string P, maka posisi pengecekan karakter pada P dan S masing-masing bergeser kekiri 1 posisi dari posisi sebelumnya, kemudian lanjutkan dengan pencocokan pada posisi tersebut dan seterusnya, kemudian jika terjadi ketidakcocokan karakter pada P dan S, maka pergeseran dilakukan dengan melihat tabel match heuristic dan occurrence heuristic dimana nilai pergeseran yang terbesar yang akan dipilih dikurangi dengan jumlah karakter yang telah cocok.

Jika semua karakter telah cocok, artinya P telah ditemukan didalam S, selanjutnya geser pattern sebesar 1 karakter, lanjutkan sampai akhir pattern S.

II. ANALISISA DAN PEMBAHASAN

Algoritma *Knuth Morris Pratt* tidak seperti algoritma Brute Force yang mencocokkan string dengan melakukan pengecekan dan melakukan pergeseran setiap satu karakter, pada algoritma *Knuth Morris Pratt* informasi ketidak cocokan *pattern* dengan teks disimpan untuk menentukan jumlah pergeseran. Sehingga algoritma *Knuth Morris Pratt* melakukan pergeseran lebih jauh sesuai dengan informasi yang disimpan, yang menyebabkan waktu pencarian dapat dikurangi secara signifikan, berikut adalah langkah kerja dari algoritma *Knuth Morris Pratt*:

1. Algoritma *Knuth Morris Pratt* mulai mencocokkan pattern pada awal teks.
2. Dari kiri ke kanan, algoritma ini akan mencocokkan karakter per karakter pattern dengan karakter di teks yang bersesuaian, sampai salah satu kondisi berikut dipenuhi:
   a. Karakter di pattern dan di teks yang dibandingkan tidak cocok (*mismatch*).
   b. Semua karakter di pattern cocok. Kemudian algoritma akan memberitahukan penemuan di posisi ini.
3. Algoritma kemudian menggeser pattern berdasarkan tabel next, lalu mengulangi langkah 2 sampai pattern berada di ujung teks.

Berikut adalah proses penerapan algoritma *Knuth Morris Pratt* untuk mencari *string* dalam suatu *string*, adapun prosesnya tampak sebagai berikut:

1. Diketahui variabel *string S* dengan *array* huruf sebagai berikut:

D	A	E	L	Y	M	M	A	K	R	I	N	A	A	M	R	S	W

2. Diketahui variabel *pattern P* merupakan kata yang akan dicari didalam variabel *S*

R	I	N	A

3. Langkah Pertama bandingkan *pattern P*[1] dengan *string S*[1], berikut adalah hasilnya

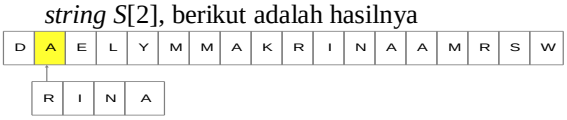
D	A	E	L	Y	M	M	A	K	R	I	N	A	A	M	R	S	W

R	I	N	A

Pattern [1] tidak cocok dengan *string* [1] maka *pattern* akan bergeser satu posisi ke kanan.

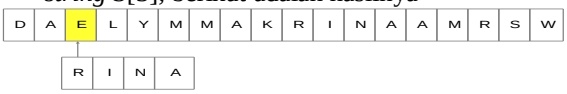
Majalah Ilmiah INTI, Volume 12, Nomor 2, Mei 2017
ISSN 2339-210X

4. Langkah Dua bandingkan *pattern P*[1] dengan *string S*[2], berikut adalah hasilnya



Pattern [1] tidak cocok dengan *string* [2] maka *pattern* akan bergeser satu posisi ke kanan.

5. Langkah Tiga bandingkan *pattern P*[1] dengan *string S*[3], berikut adalah hasilnya

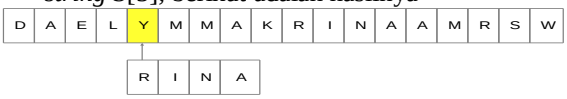


Pattern [1] tidak cocok dengan *string* [3] maka *pattern* akan bergeser satu posisi ke kanan.

6. Langkah Empat bandingkan *pattern P*[1] dengan *string S*[4], berikut adalah hasilnya

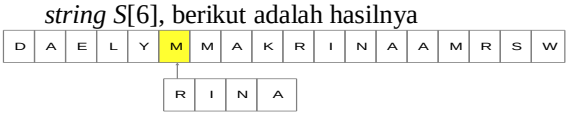
Pattern [1] tidak cocok dengan *string* [4] maka *pattern* akan bergeser satu posisi ke kanan.

7. Langkah Lima bandingkan *pattern P*[1] dengan *string S*[5], berikut adalah hasilnya



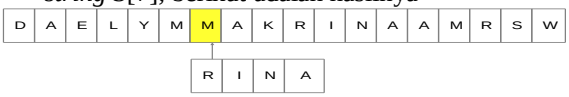
Pattern [1] tidak cocok dengan *string* [5] maka *pattern* akan bergeser satu posisi ke kanan.

8. Langkah Enam bandingkan *pattern P*[1] dengan *string S*[6], berikut adalah hasilnya



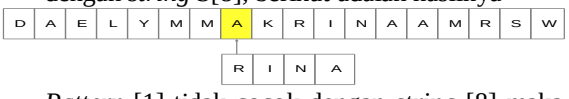
Pattern [1] tidak cocok dengan *string* [6] maka *pattern* akan bergeser satu posisi ke kanan

9. Langkah Tujuh bandingkan *pattern P*[1] dengan *string S*[7], berikut adalah hasilnya



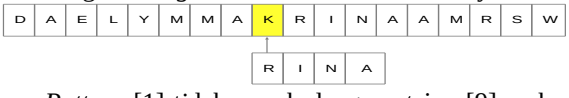
Pattern [1] tidak cocok dengan *string* [7] maka *pattern* akan bergeser satu posisi ke kanan

10. Langkah Delapan bandingkan *pattern P*[1] dengan *string S*[8], berikut adalah hasilnya



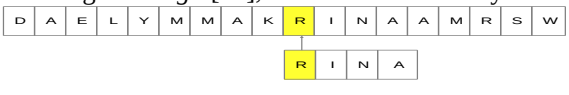
Pattern [1] tidak cocok dengan *string* [8] maka *pattern* akan bergeser satu posisi ke kanan

11. Langkah Sembilan bandingkan *pattern P*[1] dengan *string S*[9], berikut adalah hasilnya



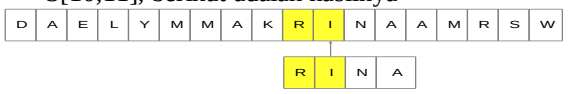
Pattern [1] tidak cocok dengan *string* [9] maka *pattern* akan bergeser satu posisi ke kanan

12. Langkah Sepuluh bandingkan *pattern P*[1] dengan *string S*[10], berikut adalah hasilnya



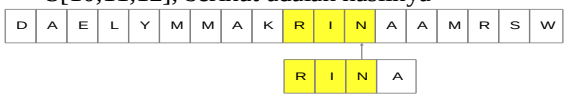
Pattern [1] cocok dengan *string* [10]. Karena ada kecocokan, maka algoritma *Knuth Morris Pratt* akan
menyimpan informasi ini, dan pattern tidak akan melakukan pergeseran dan melanjutkan pencocokan *pattern* [2] dengan *string* [11].

13. Langkah Sebelas
Bandingkan *pattern P*[1,2] dengan *string S*[10,11], berikut adalah hasilnya



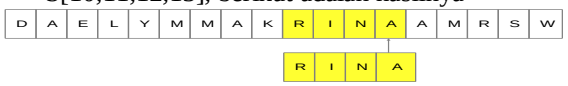
Pattern [1,2] cocok dengan *string* [10,11]. Karena ada kecocokan, maka algoritma *Knuth Morris Pratt* akan menyimpan informasi ini, dan pattern tidak akan melakukan pergeseran dan melanjutkan pencocokan *pattern* [3] dengan *string* [12].

14. Langkah Duabelas
Bandingkan *pattern P*[1,2,3] dengan *string S*[10,11,12], berikut adalah hasilnya



Pattern [1,2,3] cocok dengan *string* [10,11,12]. Karena ada kecocokan, maka algoritma *Knuth Morris Pratt* akan menyimpan informasi ini, dan pattern tidak akan melakukan pergeseran dan melanjutkan pencocokan *pattern* [4] dengan *string* [13].

15. Langkah Tigabelas
Bandingkan *pattern P*[1,2,3,4] dengan *string S*[10,11,12,13], berikut adalah hasilnya



Pattern [4] cocok dengan *string* [13]. Karena ada kecocokan, maka algoritma *Knuth Morris Pratt* akan menyimpan informasi ini, dan pattern tidak akan melakukan pergeseran dan melanjutkan pencocokan *pattern* [5] dengan *string* [14]. Namun karena jumlah *pattern* hanya 4 huruf, maka pencarian akan dihentikan dan diperoleh hasil bahwa *pattern P* terdapat kecocokan dengan *string S* sebesar 100 persen.

A. Analisa Masalah String Matching Boyer Moore

Berikut adalah proses penerapan algoritma *Boyer Moore* untuk mencari *string target* dari dari *string input.*

1. Deklarasi *string input* dan *string target*

2. Langkah pertama
Karakter terakhir dari *string target* ‘A’ tidak sesuai dengan karakter ‘L’ dari kata ‘DAELYMMAKRINAAMRSQ’, proses pemeriksan pertama sekali dilakukan dengan menghitung panjang *string target n* dan

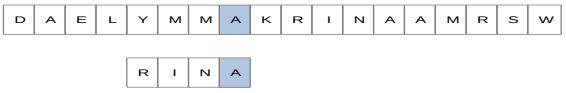
Majalah Ilmiah INTI, Volume 12, Nomor 2, Mei 2017
ISSN 2339-210X

kemudian karakter terakhir dari *string target* disesuikan dengan karakter ke *n* dari *string input*.

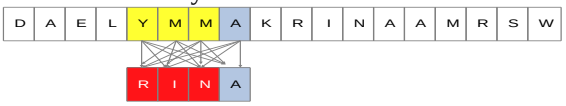
Tampak pada proses diatas huruf A != L maka proses berikutnya adalah menguji L dengan semua huruf yang ada didalam *string target* seperti dibawah ini

Karena tidak ada yang sesuai maka *string target* bergeser ke kanan sebanyak jumlah karakter.

3. Langkah Kedua karena pada langkah pertama tidak ada karakter yang sesuai maka *string target* bergeser seperti tampak pada proses berikut ini

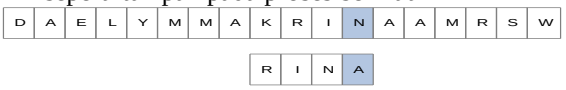


Tampak pada proses diatas huruf A == A maka proses berikutnya adalah menguji kesemua karakter yang terdapat pada *string target* sebanyak n *string target* dari *string input*, berikut hasilnya

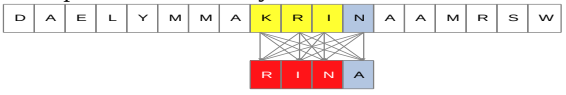


Tampak pada proses diatas bahwa karakter yang sesuai hanya karakter A dan tidak untuk karakter lainnya maka proses berikutnya *string target* maju sebanyak *n* karakter *string target*

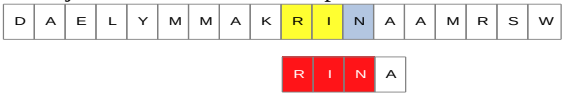
4. Langkah ketigakarena pada langkah kedua hanya karakter 'A' saja yang sesuai maka *string target* bergeser
seperti tampak pada proses berikut ini



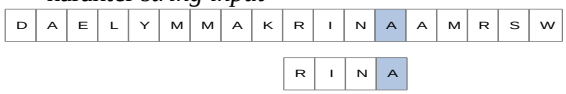
Tampak pada proses diatas huruf A tidak sama dengan N maka proses berikutnya adalah menguji kesemua karakter yang terdapat pada *string target* sebanyak n *string target* dari *string input*, berikut hasilnya



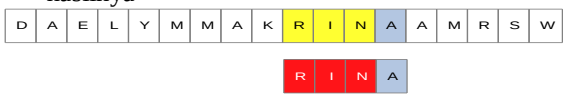
Tampak pada proses diatas terdapat beberapa karakter yang sama antara *string target* dengan *string input*, berdasarkan fungsi algoritma *boyer moore* jika terdapat karakter yang sama maka posisi karakter antara *string target* dan string tujuan harus disamakan seperti dibawah ini



Tampak pada proses diatas posisi karakter yang sesuai sudah sama antara posisi *string target* dengan *string input*, kemudian proses diatas diperiksa kembali untuk menguji apakah karakter *string target* sudah sama dengan karakter *string input*



Pada proses diatas tampak karakter terakhir *string target* 'A' sudah sama dengan string input 'A' untuk posisi yang sama, setelah itu diuji semua posisi karakter yang sama antara *string target* dengan *string input*, berikut hasilnya



Tampak pada gambar sudah sesuai maka proses dihentikan dan kata "RINA" diketemukan pada karakter 10 sampai dengan 13, maka karakter sisa dari *string input* tidak diproses.

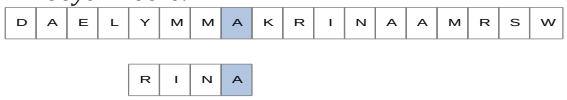
B. Analisa Kombinasi Algoritma Knuth Morris Pratt dan Boyer Moore

Berikut adalah proses penerapan kombinasi algoritma KMP dan Boyer Moore.

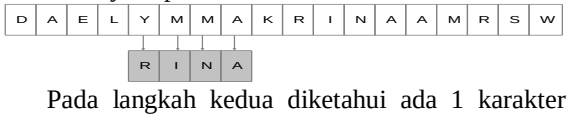
1. Diketahui *string input* adalah DAELYMMAKRINAAMRSQ dan *string target* adalah RINA

2. Langkah pertama yang dilakukan adalah memeriksa posisi yang sesuai antara *string input* dan *string target*,dengan menggunakan pencocokan kmp.

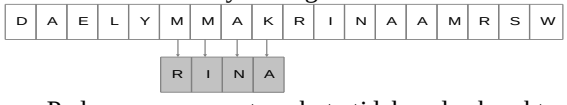
Dikarenakan tidak ada karakter yang sama, maka *string target* bergeser sebanyak panjang *string target* dengan menggunakan pencarian *boyer moore*.



3. Langkah kedua setelah bergeser dilanjutkan dengan menggunakan pencocokkan *KPM* dikarenakan tidak ada karakter yang sama, hasilnya seperti berikut.



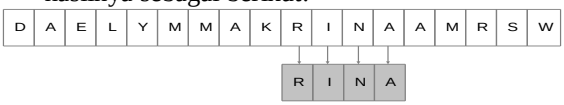
Pada langkah kedua diketahui ada 1 karakter yang sama antara *string target* dan *string input* maka string input bergeser 1 posisi ke sebelah kanan dan hasilnya sebagai berikut:



Pada pergeseran tersebut tidak ada karakter yang sama, *string target* bergeser sebanyak panjang *string target*

Majalah Ilmiah INTI, Volume 12, Nomor 2, Mei 2017
ISSN 2339-210X

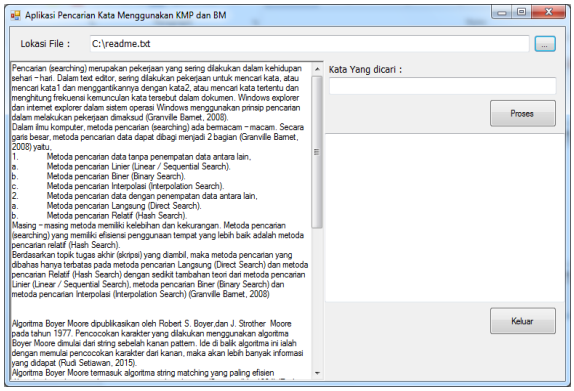
4. Langkah ketiga setelah pergeseran *string target* dikarenakan ketidaksesuaian karakter maka hasilnya sebagai berikut:



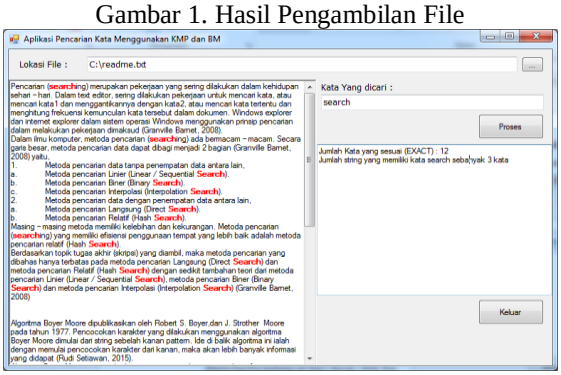
Pada langkah ini untuk setiap karakter pada *string target* sesuai dengan *string input* maka proses diberhentikan karena *string* sudah ketemu perhatikan perbedaan pencocokan *string* yang dilakukan menggunakn algoritma Boyer Moore dan KMP, prosesnya lebih cepat karena hanya mencocokan antara karakter saja dan bukan untuk keseluruhan karakter yang ada pada *string target*.

IV. IMPLEMENTASI

Implementasi dari sistem yang dirancang, menggunakan antar muka pengolahan data dari pengujian. berikut ini adalah tampilan program aplikasi.



Gambar 1. Hasil Pengambilan File



Gambar 2. Hasil Pencarian

V. KESIMPULAN

Kesimpulan dari penulisan penelitian ini adalah:

1. Pencarian kata dengan menggunakan algoritma *Knuth Morris Pratt* dan *Boyer Moore* yang dirancang mampu menampilkan *file text* yang ada didalam komputer berdasarkan kata kunci yang dimasukkan pengguna dan sesuai dengan isi dari *file text* tersebut
2. Kombinasi dilakukan dengan menggunakan pencarian algoritma *KMP* dan *Boyer Moore* secara implisit sistem dimana file text yang akan dicari akan di proses berdasarkan array yang sesuai dengan string yang dimasukan pengguna.
3. Penggunaan merancang aplikasi pencarian kata dengan menggunakan *Visual Basic.Net 2008* sebagai bahasa pemrograman dalam membuat aplikasi sudah tepat dikarenakan kemudahan dalam memproses string yang cukup cepat dan sudah lengkap dengan fungsi proses pencarian dari *Visual Basic.Net 2008*.

DAFTAR PUSTAKA

[1] Fandi Prahana, (2012). Penerapan *String* Matching pada Fitur Auto Correct dan Fitur Auto Text di Smart Phones. Bandung. Institut Teknologi Bandung.
[2] Kevin Wibowo, (2012) "Perbandingan Algoritma KMP dan Algoritma boyer moore dalam pencarian teks di Bahasa Indonesia dan inggris". Institut Teknologi Bandung.
[3] Andri Januardi, (2013) Pelita informatika budidarma Medan.
[4] Moch.Yusuf Soleh, (2011) "implementasi Algoritma KMP dan Boyer moore dalam Aplikasi Search Engine Sadarhana". Institut Teknologi Bandung.
[5] Maya Rossaria, (2015)"Implementasi Algoritma pencocokan string KMP dalam aplikasi pencarian dokumen digital berbasis android".Universitas Bengkulu.
[6] G. L. Ginting, "Implementasi Algoritma Boyer-Moore Pada Aplikasi Pengajuan Judul Skripsi Berbasis Web," *Pelita Inform.*, vol. 3, no. 1, 2014.
[7] G. L. Ginting, "Implementasi Algoritma Boyer-Moore Pada Aplikasi Pengajuan Judul Skripsi Berbasis Web," *Pelita Inform.*, 2014.
[8] F. T. Waruwu and Mesran, "IMPLEMENTASI ALGORITMA KNUTH MORRIS PRATT PADA APLIKASI KAMUS ISTILAH LATIN FLORA DAN FAUNA BERBASIS ANDROID," *Maj. Ilm. INTI*, vol. 4, no. 1, pp. 96–102, 2014.
[9] Mesran, "IMPLEMENTASI ALGORITMA BRUTE FORCE DALAMPENCARIAN DATA KATALOG BUKU PERPUSTAKAAN," *Maj. Ilm. INTI*, vol. 3, no. 1, pp. 100–104, 2014.
[10] J. I. Sinaga, Mesran, and E. Buulolo, "APLIKASI MOBILE PENCARIAN KATA PADA ARTI AYAT AL-QUR'AN BERBASIS ANDROID MENGGUNAKAN ALGORITMA STRING MATCHING," *INFOTEK*, vol. 2, no. 2, pp. 68–72, 2016.