JURNAL TIMES
TECHNOLOGY INFORMATICS & COMPUTER SYSTEM

Volume VI No 1, Juni 2017
*p*ISSN : 2337 – 3601
*e*ISSN : 2549 – 015X

Tersedia *online* di http://ejournal.stmik-time.ac.id

Analisa Perbandingan Boyer Moore Dan Knuth Morris Pratt Dalam Pencarian Judul Buku Menerapkan Metode Perbandingan Eksponensial (Studi Kasus : Perpustakaan STMIK Budi Darma)

Alwin Fau[1], Mesran [2], Guidio Leonarde Ginting [3]

Sekolah Tinggi Manajemen Informatika dan Komputer (STMIK) Budidarma[1,2,3]
Jl. Sisingamangaraja No. 338 Simpang Limun Medan
e-mail : alwin.fau@stmik-budidarma.ac.id[1], mesran.skom.mkom@gmail.com[2], guidio.leonardo626@gmail.com[3]

Abstrak
Analisa adalah merupakan suatu proses merinci terhadap objek dengan alat bantu tertentu, kedalam beberapa komponen yang saling berhubungan dengan menilai dan mengetahui perbedaan dari kedua objek tersebut yang berbeda. Dalam proses pencarian ada beberapa algortima yang dibutuhkan untuk menyelesaikan masalah yang sedang dihadapi. Adapun permasalahannya yaitu dalam proses pencarian judul buku pada perpustakaan Perpustakaan STMIK Budidarma Medan dimana proses pencarian yang dilakukan masih membutuhkan waktu yang sangat lama. String matching adalah proses pencarian semua kemunculan query yang selanjutnya disebut pattern kedalam string yang lebih panjang (teks). Algortima adalah urutan atau langkah-langkah yang disusun secara sistematis untuk myelesaikan sebuah masalah. Adapun algortima yang digunakan dalam menyelesaiakan masalah tersebut yaitu Algortima boyer moore dan algortima knuth morris pratt (KMP). Algortima boyer moore adalah sebuah algortima pencarian yang dimana proses atau cara pencariannnya dilakukan dari kanan pattern sehingga hasil pecarian lebih cepat ditemukan. Algortima Knuth morris pratt (KMP) adalah sebuah algoritma pencarian string yang bekerja dengan memanfaatkan pergeseran pattern dalam teks dari sebeleh kiri kekanan dalam melakukan pencocokan pattern dalam teks. Analisa dalam perbandingan dari kedua algortima pada penelitian ini dilakukan untuk mengetahui algoritma yang mana proses pencarian dan cara kerjanya lebih cepat dengan memanfaatkan metode perbandingan eksponensial (MPE) sebagai metode pengambilan keputusan dalam menentukan hasil perbandingannya.
Kata Kunci : String Matching, Boyer Moore, Knuth Morris Pratt (KMP), Perbandingan Ekponensial (MPE)

1. Pendahuluan

String matching atau pencocokan string merupakan suatu proses pencarian semua kemunculan *query* yang selanjutnya disebut dengan *pettern* ke dalam *strin*g untuk kesesuaian kebutuhan informasi yang dibutuhkan[1].

Pencocokan *string* atau *string matching* memiliki sifat yaitu mencari sebuah *string* yang terdiri dari beberapa karakter (yang biasa disebut dengan *pattern*) dan sejumlah besar text. *String matching* dapat juga digunakan untuk mencari pola bit dalam jumlah besar *file binery*. *String matching* dapat di implementasikan dalam berbagai bidang diantaranya[2][3][4] sebagai berikut :

1. Pada aplikasi kamus
2. Pada permaian *Word Search* digunakan untuk menemukan solusi
3. Dan juga digunakan untuk mesin pencarian seperti *Google*, *Yahoo* dan situs yang lainnya yang dapat digunakan untuk pencarian informasi.

String matching atau pencocokan string secara garis besar dibedakan menjadi dua yaitu pencocokan *string* secara eksak/sama persis (*exact string matching*) dan pencocokan string berdasarkan kemiripan (*inexact string matching*). *Inexact string matching* merupakan pencocokan *string* yang melakukan pencarian terhadap *string* yang sama dan juga *string* yang mendekati dengan *string* yang lain yang terkumpul dalam sebuah penampung. Pencocokan *string* berdasarkan kemiripan masih dibedakan menjadi dua yaitu berdasarkan kemiripan penulisan (*approximate string matching*) dan berdasarkan ucapan kemiripan (*phonetic string matching*). Dalam dalam penelitian ini, penulis menggunakan *exact string matching* yang dimana proses pencarian yang dilakukan yaitu secara eksak.

Algoritma pencarian terbagi dalam beberapa alternatif dimana diantaranya adalah algoritma *boyer moore*, algoritma *knuth morris pratt*, algoritma *bruce force* . dari algortima yang telah ada, algoritma tersebut memiliki fungsi dan cara kerja yang berbeda-beda dalam proses pencarian atau pencocokan string. Dengan munculnya perbedaan dari berbagai algortima tersebut maka akan sangat membingungkan dalam memilih algoritma manakah yang dapat digunakan dalam pencarian yang dimana proses pencarian dan memiliki cara kerja yang cepat.

Algoritma *Knuth-Morris-Pratt* merupakan sebuah Algoritma pencarian string yang bekerja dengan memanfaatkan pergeseran *pattern* teks dari kiri kekanan dalam pencocokkan string dalam teks. Algoritma *Knuth-Morris-Pratt* dikembangkan pertama sekali oleh Donald E. Knuth pada tahun 1967 dan kemudian oleh James H. Morris bersama Vaughan R. Pratt pada tahun 1966. Kemudian ditahun 1977 Algoritma tersebut dipublikasikan secara bersamaan[1].

Algoritma *Boyer-Moore* adalah salah satu Algoritma pencarian *string* yang dipublikasikan oleh Robert S. Boyer dan J. Strother Moore pada tahun 1997. Tidak seperti dua algortima sebelumnya, Adapun cara kerja yang dilakukan Algoritma *Boyer-Moore* yaitu dengan mencocokan karakter dari sebelah kanan *pattern* sehingga proses pencariannya lebih cepat[3]. Dibalik algortima ini dengan melakukan pencocokan karakter dari kanan, dan bukan dari kiri, maka akan lebih banyak informasi yang didapatkan. Jadi kita bisa melompati atau tidak melakukan perbandingan-perbandingan karakter yang diprediksikan akan gagal. Karakter paling kanan pada pola merupakan karakter pertama yang akan dicocokan dengan teks. Sehingga prinsip dari algortima *boyer moore* ini yakni :

1. Pembacaan karakter dilakukan dari kanan ke kiri.
2. *Preprocessing* dimana terdapat kondisi *bad karakter preprocessing* dan *good suffix preprocessing*.

Untuk dapat menganalisa perbandingan dan perbedaan cara kerja dari kedua algoritma tersebut, maka penulis menggunakan Metode Perbandingan Eksponensial (MPE) sebagai metode untuk membandingkan kedua algoritma tersebut.

2. Landasan Teori

String Matching

String Matching adalah proses pencarian semua kemunculan *query* yang selanjutnya disebut *pattern* kedalam *string* yang lebih panjang (teks)[1]. *String Matching* dirumuskan sebagai berikut :

x = x[0...m-1] ..... (1)
y = y[0...n-1] ..... (2)

Keterangan :

x adalah *pattern*
m adalah panjang *pettern*
y adalah text
n adalah panjang teks

Exact Matching

Exact Matching digunakan untuk menemukan *patern* yang berasal dari teks dari satu teks. Contoh pencarian *exact matching* adalah pencarian kata "pelajar" dalam kelimat "Saya seorang pelajar" atau "Saya seorang siswa". Sistem akan memberikan hasil bahwa kalimat pertama mengandung kata "pelajar" sedangkan pada kalimat kedua tidak, meskipun kenyataannya kata pelajar dan siswa adalah kata yang bersinonim, algoritma *exact matching* diklasifikasikan menjadi tiga bagian menurut arah pencariannya, yaitu

1. Arah pembacaaan dari kiri kekananan. Algoritma yang termasuk kategori ini adalah *Bruce Force, Knuth Morris Pratt* (yang dikembangkan oleh *Knuth,Morris* dan *Pratt*).
2. Arak pembacaaan dari kanan ke kiri. Algoritma yang termasuk kategori ini adalah *Boyer Moore* yang kemudian dikembangkan menjadi algoritma *Turbo Boyer-Moore, Tuned Boyer-Moore,* dan *Zhu-Takaoka*.

Arah pembacaan yang ditentukan oleh pemrogram. Algoritma yng termasuk pada kategori ini adalah Algoritma *Collusi, Crochemore-Perrin*.

Heuristic Matching

Heuristic Matching adalah teknik yang digunakan untuk menghubungkan dua kata terpisah ketika *exact matching* tidak mampu mengatasi karena ada pembatasan pada data yang tersediaa[1]. *Heuristic Matching* dapat dilakukan dengan perhitungan *distance* antara *pattern* dan teks.

Algoritma Boyer Moore

Bad Character

Bad Character menunjukkan seberapa banyak pergeseran karakter dapat melompat kedepan dalam teks setelah ketidak cocokan. *Heuristic Bad Character* dibuat dalam *array* dimana posisi masing-masing mewakili karakter dalam |S| dan nilai masing-masing adalah jarak minimal dari karakter yang ke akhir *pattern* (ketika karakter muncul lebih dari sekali dalam *pattern*, hanya hal-hal kejadian terakhir). Dalam *pattern*, misalnya yang terakhit diikuti oleh satu karakter lagi, sehingga posisi ditugaskan kedalam *array* berisi nilai 1[5][6].

Pattern Position	: 1 2 3 4 5
Pattern Character	: d a b a b
Character	: a b c d
Bad-Character Heuristic	: 1 0 3 4

Sebelum terjadi ketidakcocokan dalam *pattern*, semakin jauh ketidak cocokan disebabkan oleh karakter yang memungkinkan kita untuk melewati. Karakter *Mismatch* tidak terjadi dama sekali dalam pattern memungkinkan kita untuk melewati dengan kecepatan maksimal, Heuristik membutuhkan ruang |∑|.

Good Suffix

Good Suffix adalah cara lain untuk mengatakan berapa banyak karakter yang kita dapat lewati jika tidak ada suatu ketidakcocokan. Heuristik didasarkan pada urutan mundur pencocokan *boyer moore*[5]. Heuristik tersebut disimpan dalam *array* dimana posisi masing-masing mewakili posisi dalam *pattern*. Hal ini dapat ditemukan membandingkan pola terhadap dirinya sendiri, seperti yang dilakukan pada *Knuth-Morris-Pratt*. *Good Suffix* membutuhkan ruang *m* dan diindeks oleh ketidakcocokan dalam *pattern*. Jika ketidakcocokan dalam posisi (0-based) 3 *pattern*, kita mencari heuristik yang akhirnya baik dari posisi *array* yang ke-3 :

Posisi Pattern	: 0 1 2 3 4
Pattern Character	: d a b a b
Good Suffix Heuristik	: 5 5 5 2 1

Algoritma *Knuth Morris Pratt*

Algoritma *Knuth Morris Pratt* merupakan proses pencocokan *string* [1]. Bila terjadi ketidak cocokan pada saat *pattern* sejajar dengan teks [i...i + n -1], kita bisa menganggap ketidak cocokan pertama terjadi diantara teks [i+j] dan *pattern* [j], dengan <j<n. Berarti, teks [i...i + j] = *pattern* [0..j + 1] dan a = teks [i + j] tidak sama dengan b = *pattern* [j], ketika kita menggeser.

Dengan kata lain, pencocokan string akan berjalan secara efesien bila kita mempunyai tabel yang menentukan berapa panjang seharunya menggeser seandainya terdeteksi ketidak cocokan di karakter ke-j dari *pattern*. Tabel itu harus memuat next [j] yang merupakan posisi karakter *pattern* setelah digeser, sehingga kita menggeser pattern secara besar j-next [j] relatif terhadap teks.

Secara sistematis, langkah-langkah yang dilakukan dalam *Knuth Morris Pratt* pada saat mencocokan *string*[7], sebagai berikut :

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

Metode Perbandingan Eksponensial (MPE)

Metode perbandingan eksponensial adalah salah satu metode untuk menentukan urutan prioritas alternatif keputusan dengan kriteria jamak[8][9].

$$Total\ Nilai\ (TN_i) = \sum_{j=1}^{m} (RK_{ij})\mathrm{TKK_j} \quad \text{..... (3)}$$

Keterangan :

TN_i	= Total nilai alternatif ke-i
RK_{ij}	= Derajat kepentingan relatif alternatif ke-j pada pilihan keputusan i
TKK_j	= Derajat kepentingan kriteria keputusan ke-j; $TKK_j > 0$; bulat
M	= Jumlah Kriteria Keputusan
N	= Jumlah pilihan keputusan
j	= 1,2,3,... ; m = jumlah kriteria
i	= 1,2,3,...,n ; n = jumlah pilihan alternatif

3. Analisa dan Perancangan

Analisa yang dilakukan dalam membandingkan proses cara kerja dari algoritma *Boyer Moore* dan algortima *Knuth Morris Pratt* hanya mengarah pada pencarian judul buku pada perpustakaan STMIK Budi Darma Medan. Secara garis besar penulis melakukan analisa perbandingan dari kedua algortima tersebut karena dilatar belakangi dari proses pencarian buku pada perpustakaan STMIK Budi Darma Medan. Proses pencarian buku pada perpustakaan STMIK Budi Darma Medan masih dilakukan dengan proses pencarian yang memakan waktu cukup lama atau belum ada aplikasi yang bisa digunakan dalam melakukan pencarian judul buku pada perpustakaan. sehingga dari permasalahan yang diatas penulis berusaha menganalisa algortima *Boyer Moore* dan algortima *Knuth Morris Pratt*, algortima yang manakah cara atau proses pencariannya lebih cepat sehingga dari analisa yang diperoleh dapat dijadikan sebagai acuan untuk membuat aplikasi pencarian judul buku pada perpustakaan STMIK Budi Darma Medan. Metode perbandingan eksponensial (MPE) merupakan metode yang digunakan untuk melakukan membandingkan dan perengkingan dari algoritma *Boyer Moore* Dan algoritma *Knuth Morris Pratt* tersebut.

Dalam menganalisa cara kerja kedua algoritma tersebut maka perlu adanya sebuah aplikasi yang akan mengetahui bagaimana proses yang dihasilkan dari masing-masing algortima tersebut dalam melakukan pencarian.

aplikasi yang dirancang ini yaitu aplikasi yang berbasis dekstop. *tools* yang digunakan untuk merancang aplikasi analisa perbandingan tersebut yaitu dengan menggunakan *Microsoft Visual Studio. Net* 2008 sebagai alat bantu dalam melakukan *design* dan *source code*.

Manfaat yang diperoleh dalam menganalisa perbandingan kedua algortima tersebut adalah dapat dengan mudah memahami bagaimana proses pencarian yang dalakukan dan bagaimana kecepatan yang dilakukan dari algoritma tersebut dalam melakukan pencarian.

Penerapan Algoritma *Boyer Moore*

Contoh penggunaan algortima *Boyer Moore* Dalam melakukan pencarian dalam teks :
Teks (S) : PENGOLAHAN CITRA DIGITAL
Pattern (P) : CITRA
Tahapan pencarian *pattern* (P) dalam Teks (S) :

Tabel 1. Analisa Penentuan Nilai OH dan MH

Pattern (P):	C	I	T	R	A
Occurence Heuristic (OH) :	4	3	2	1	0
Match Heuristic (MH) :	5	5	5	5	1

Langkah-Langkah :

a. Pada pergeseran pertama karakter “A” pada *pattern* tidak cocok dengan karakter “O” pada teks, maka pergeseran selanjutnya berdasarkan nilai dari tabel OH. Pada tabel OH karakter O tidak terdapat pada tabel, sehingga pergeseran selajutnya adalah sebanyak jumlah karakter “A” pada *pattern* yaitu 5.

Langkah Ke -1																								
Posisi Teks (S) :	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
Teks (S) :	P	E	N	G	O	L	A	H	A	N		C	I	T	R	A		D	I	G	I	T	A	L
Pattern (P) :	C	I	T	R	A																			

b. Pada pergeseran Ke-2 karakter “A” pada *pattern* tidak cocok dengan karakter “N” pada teks, maka pergeseran selanjutnya berdasarkan nilai dari tabel OH. Pada tabel OH karakter N tidak terdapat pada tabel, sehingga pergeseran selajutnya adalah sebanyak jumlah karakter “A” pada *pattern* yaitu 5.

Langkah Ke -2																								
Posisi Teks (S) :	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
Teks (S) :	P	E	N	G	O	L	A	H	A	N		C	I	T	R	A		D	I	G	I	T	A	L
Pattern (P) :						C	I	T	R	A														

c. Pada pergeseran Ke-3 karakter “A” pada *pattern* tidak cocok dengan karakter “R” pada teks, maka pergeseran selanjutnya berdasarkan nilai dari tabel OH. Pada tabel OH karakter R terdapat pada tabel, sehingga pergeseran selajutnya adalah sebanyak jumlah karakter “R” pada tabel OH pada yaitu 1.

Langkah Ke -3																								
Posisi Teks (S) :	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
Teks (S) :	P	E	N	G	O	L	A	H	A	N		C	I	T	R	A		D	I	G	I	T	A	L
Pattern (P) :											C	I	T	R	A									

d. Pada pergeseran Ke-4 karakter “A” pada *pattern* cocok dengan karakter “A” pada teks, maka pergeseran selanjutnya dimundurkan satu langkah.

Langkah Ke -4																								
Posisi Teks (S) :	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
Teks (S) :	P	E	N	G	O	L	A	H	A	N		C	I	T	R	A		D	I	G	I	T	A	L
Pattern (P) :												C	I	T	R	A								

e. Pada pergeseran selanjutnya dilakukan sampai pada pergeseran ke-12 karakter “C” pada *pattern* dengan karakter “C” pada text cocok.

Penerapan Algoritma Knuth Morris Pratt (KMP)

Contoh penggunaan algortima *Knuth Morris Pratt* Dalam melakukan pencarian dalam teks :
Teks (S) : PENGOLAHAN CITRA DIGITAL
Pattern (P) : CITRA

Langkah Ke - 1																								
Teks (S)	P	E	N	G	O	L	A	H	A	N		C	I	T	R	A		D	I	G	I	T	A	L
Pattern (P)	C	I	T	R	A																			
Indeks	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24

Keterangan : Karakter C dengan indeks pertama pada *pattern* tidak cocok dengan karakter P pada teks, maka pergeseran dimajukan 1 langkah ke kanan menuju indeks berikutnya.

Langkah Ke - 2																								
Teks (S)	P	E	N	G	O	L	A	H	A	N		C	I	T	R	A		D	I	G	I	T	A	L
Pattern (P)		C	I	T	R	A																		
Indeks	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24

Keterangan : Karakter C dengan indeks ke-2 pada *pattern* tidak cocok dengan karakter E pada teks, maka pergeseran dimajukan 1 langkah ke kanan menuju indeks berikutnya.

Langkah Ke - 3																								
Teks (S)	P	E	N	G	O	L	A	H	A	N		C	I	T	R	A		D	I	G	I	T	A	L
Pattern (P)			C	I	T	R	A																	
Indeks	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24

Keterangan : Karakter C dengan indeks ke-3 pada *pattern* tidak cocok dengan karakter N pada teks, maka pergeseran dimajukan 1 langkah ke kanan menuju indeks berikutnya.

Langkah Ke - 4																								
Teks (S)	P	E	N	G	O	L	A	H	A	N		C	I	T	R	A		D	I	G	I	T	A	L
Pattern (P)				C	I	T	R	A																
Indeks	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24

Keterangan : Karakter C dengan indeks ke-4 pada *pattern* tidak cocok dengan karakter G pada teks, maka pergeseran dimajukan 1 langkah ke kanan menuju indeks berikutnya.

Langkah Ke - 5																								
Teks (S)	P	E	N	G	O	L	A	H	A	N		C	I	T	R	A		D	I	G	I	T	A	L
Pattern (P)					C	I	T	R	A															
Indeks	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24

Keterangan : Karakter C dengan indeks ke-5 pada *pattern* tidak cocok dengan karakter O pada teks, maka pergeseran dimajukan 1 langkah ke kanan menuju indeks berikutnya.

Langkah Ke - 6																								
Teks (S)	P	E	N	G	O	L	A	H	A	N		C	I	T	R	A		D	I	G	I	T	A	L
Pattern (P)						C	I	T	R	A														
Indeks	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24

Keterangan : Karakter C dengan indeks ke-6 pada *pattern* tidak cocok dengan karakter L pada teks, maka pergeseran dimajukan 1 langkah ke kanan menuju indeks berikutnya.

Langkah Ke - 7																								
Teks (S)	P	E	N	G	O	L	A	H	A	N		C	I	T	R	A		D	I	G	I	T	A	L
Pattern (P)							C	I	T	R	A													
Indeks	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24

Keterangan : Karakter C dengan indeks ke-7 pada *pattern* tidak cocok dengan karakter A pada teks, maka pergeseran dimajukan 1 langkah ke kanan menuju indeks berikutnya.

Langkah Ke - 8																								
Teks (S)	P	E	N	G	O	L	A	H	A	N		C	I	T	R	A		D	I	G	I	T	A	L
Pattern (P)								C	I	T	R	A												
Indeks	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24

Keterangan : Karakter C dengan indeks ke-8 pada *pattern* tidak cocok dengan karakter H pada teks, maka pergeseran dimajukan 1 langkah ke kanan menuju indeks berikutnya.

Langkah Ke - 9																								
Teks (S)	P	E	N	G	O	L	A	H	A	N		C	I	T	R	A		D	I	G	I	T	A	L
Pattern (P)									C	I	T	R	A											
Indeks	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24

Keterangan : Karakter C dengan indeks ke-9 pada *pattern* tidak cocok dengan karakter A pada teks, maka pergeseran dimajukan 1 langkah ke kanan menuju indeks berikutnya.

Langkah Ke - 10																								
Teks (S)	P	E	N	G	O	L	A	H	A	N		C	I	T	R	A		D	I	G	I	T	A	L
Pattern (P)										C	I	T	R	A										
Indeks	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24

Keterangan : Karakter C dengan indeks ke-10 pada *pattern* tidak cocok dengan karakter N pada teks, maka pergeseran dimajukan 1 langkah ke kanan menuju indeks berikutnya.

Langkah Ke - 11																								
Teks (S)	P	E	N	G	O	L	A	H	A	N		C	I	T	R	A		D	I	G	I	T	A	L
Pattern (P)											C	I	T	R	A									
Indeks	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24

Keterangan : Karakter C dengan indeks ke-11 pada *pattern* tidak cocok dengan spasi pada teks, maka pergeseran dimajukan 1 langkah ke kanan menuju indeks berikutnya.

Langkah Ke - 12																								
Teks (S)	P	E	N	G	O	L	A	H	A	N		C	I	T	R	A		D	I	G	I	T	A	L
Pattern (P)												C	I	T	R	A								
Indeks	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24

Keterangan : Karakter C dengan indeks ke-12 sampai dengan karakter A pada *pattern* dengan indeks ke-16 cocok dengan karakter pada teks, maka *pattern* telah ditemukan dan proses pencarian *pattern* berhenti dan tidak ada lagi pergeseran ke indeks berikutnya.

Metode Perbandingan Eksponensial (MPE)

Dalam menghitung dan membandingkan proses pencarian dari kedua algoritma tersebut adalah sebagai berikut:

1. Menentukan alternatif
   Untuk menganalisa perbandingan kecepatan antara algortima *Boyer Moore* dan algoritma *Knuth Morris Pratt* dalam melakukan pencarian maka perlu dilakukan penentuan algortima yang mana yang akan digunakan sebagai algoritma pencarian.
2. Menentukan kriteria
   Untuk dapat mebandingkan kedua alternatif tersebut, maka selanjutnya perlu dilakukan penentuan kriteria dalam menganlisa proses dan cara kerjanya. Untuk kriterianya dapat dilihat pada tabel berikut :

Tabel 2. Penentuan Kriteria

Kriteria	Keterangan
Besar memori yang digunakan saat melakukan pencarian	Perhitungan pemakaian memori terjadi pada saat algoritma malakukan pencocokan *string*
Jumlah Waktu Yang Digunakan Dalam Melakukan Pencarian	Perhitungan waktu diperoleh pada saat algoritma melakukan pencocokan *string* dari awal pencocokan sampai selesai

3. Menentukan bobot kriteria
   Penentuan bobot merupakan salah satu komponen yang sangat berpengaruh terhadap nilai analisa, untuk itu menetepkan bobot kriteria berdasarkan tingkatan pengaruh dalam menentukan kecepatan dalam melakukan pencarian. Pembobotan kriteria dapat dilihat pada tabel dibawah ini :

Tabel 3. Pembobotan Masing-Masing Kriteria

Kriteria	Presentase pengaruh kriteria	Bobot R*ange* (0-1)	Keterangan
Jumlah pemakaian memori	50%	0,5	Tingkat pengaruh penggunan memori sangat berpengaruh dalam menentukan kecepatan sebuah algoritma dalam melakuakan pencarian karena semakin banyak kapasitas memeri tang digunakan saat melakukan pencarian, makan maka akan semakin lambat juga suatu algoritma menyelesaikan masalah
Wktu yang dibutuhkan	50 %	0,5	Penilaian terhadap waktu dalam melakukan proses pencarian merupakan komponen yang dapat memberikan Suatu nilai terahadap algoritma dalam melakukan pencarian.

4. Pemberian Nilai Pada Setiap Kriteria
   Pada kriteria yang telah dibentuk harus diberikan nilai. Nilai tersebut dapat dilihat pada contoh dibawah ini yang dimana nilainya diambil berdasarkan analisa algoritma *Boyer Moore* dan *Knuth Morris Pratt* sebelumnya.

Tabel 4. Pemberian Nilai Terhadap Setiap Kriteria

Alternatif	Proses Ke-	*Pattern*	Kriteria	
			Kapasitas Memori (byte)	Jumlah Waktu (ms)
Algoritma *Boyer Moore*	1	Citra	698	0,4741052
	2	Digital	693	0,3240782
	3	Pemrograman	695	0,346412
	4	Animasi	692	0,3293661
	5	Algoritma	697	0,2621291
Algoritma *Knuth Morris Pratt*	1	Citra	722	0,3370435
	2	Digital	720	0,3403657
	3	Pemrograman	717	0,2768853
	4	Animasi	720	0,3359458
	5	Algoritma	719	0,3205129

5. Menghitung Nilai
Setelah melakukan pengisian nilai terhadap masing-masing kriteria, maka proses berikutnya adalah malakukan perhitungan dengan menggunakan rumus dari Metode Perbandingan Eksponensial (MPE). Proses perhitungannya sebagai berikut :

Tabel 5. Perhitungan Analisa Perbandingan Dengan Menggunakan MPE

Proses Ke-	Kriteria						Total Nilai BM	Total Nilai KMP
	Jumlah Memori			Jumlah Waktu				
	B	BM	KMP	B	BM	KMP		
		N	N		N	N		
1	0,5	698	721	0,5	0,4741052	0,3370435	27,10815	27,43195
2	0,5	693	723	0,5	0,3240782	0,3403657	26,89407	27,427
3	0,5	695	722	0,5	0,346412	0,2768853	26,95136	27,39619
4	0,5	692	720	0,5	0,3293661	0,3359458	26,8797	27,4124
5	0,5	697	719	0,5	0,2621291	0,3205129	26,9127	27,38023
						Total Value	134,74598	137,04777

Keterangan :
1. **B** : Bobot
2. **BM** : Algortima *Boyer Moore*
3. **KMP** : Algortima *Knuth Morris Pratt*
4. **N** : Nilai Kriteria
5. **Total Nilai** : $\sum$ **(N)** ^{B}

Langkah-langkah / proses perhitungan adalah sebagai berikut :
a. Proses Perhitungan total nilai pada proses ke-1 :
Nilai BM = $(698)^{0,5} + (0,4741052)^{0,5}$
= 26,4196 + 0,68855
= 27,10815
Nilai KMP = $(721)^{0,5} + (0,3370435)^{0,5}$
= 26,8514 + 0,58055
= 27,43195
b. Proses Perhitungan total nilai pada proses ke-2 :
Nilai BM = $(693)^{0,5} + (0,3240782)^{0,5}$
= 26,3248 + 0,56927
= 26,89407
Nilai KMP = $(723)^{0,5} + (0,3403657)^{0,5}$
= 26,8886 + 0,58340
= 27,427
c. Proses Perhitungan total nilai pada proses ke-3 :
Nilai BM = $(695)^{0,5} + (0,346412)^{0,5}$
= 26,3628 + 0,58856
= 26,95136

Nilai KMP = (722) ^{0,5} + (0,2768853) ^{0,5}
= 26,8700 + 0,52619
= 27,39619

d. Proses Perhitungan total nilai pada proses ke-4 :
Nilai BM = (692) ^{0,5} + (0,3293661) ^{0,5}
= 26,3058 + 0,57390
= 26,8797
Nilai KMP = (720) ^{0,5} + (0,3359458) ^{0,5}
= 26,8328 + 0,57960
= 27,4124

e. Proses Perhitungan total nilai pada proses ke-5 :
Nilai BM = (697) ^{0,5} + (0,2621291) ^{0,5}
= 26,4007 + 0,51200
= 26,9127
Nilai KMP = (719) ^{0,5} + (0,3205129) ^{0,5}
= 26,8141 + 0,56613
= 27,38023

f. Menghitung nilai prioritas keputusan
Total Nilai BM = 27,10815 + 26,89407 + 26,95136 + 26,8797 + 26,9127
= 134,74598
Total Nilai KMP = 27,43195 + 27,427 + 27,39619 + 27,4124 + 27,38023
= 137,04777

6. Menentukan Hasil Atau Prioritas Keputusan
Setelah diperoleh nilai akhir atau total nilai dari masing-masing alternatif, maka tahapan selanjutnya yang perlu dilakukan adalah menentukan prioritas keputusan berdasarkan nilai dari masing-masing alternatif. Hasil prioritas keputusan dapat dilihat pada tabel dibawah ini :

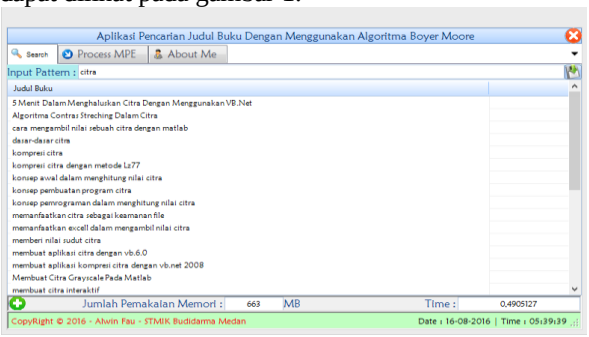
Tabel 6. Prioritas Keputusan

Alternatif	Total Nilai	Rangking
Algoritma *Boyer moore*	134,74598	1
Algoritma *Knuth Morris Pratt*	137,04777	2

4. Hasil Penelitian

Implementasi Algoritma Boyer Moore

Pada implementasi algoritma *Boyer Moore* menampilkan proses pencocok *string* yang telah dilakukan pencocokan antara *pattern* yang telah diinput dengan *field* judul dari tabel *Book* yang terdapat pada *database* DBSearchBMKMP. Implementasi algoritma *Boyer Moore* dalam pencocokan *string* pada analisa perbandingan dalam pencarian judul buku, dapat dilihat pada gambar 1.



Gambar 1. Hasil Penerapan Boyer Moore

Perhitungan Hasil Pencarian Algortima *Boyer Moore* Dengan MPE

Hasil perhitungan total nilai dari algortima *Boyer Moore* dapat dilihat pada gambar 2 dibawah ini :

Gambar 2. Hasil Total Nilai Pencarian *Boyer Moore* dalam teks setelah ditemukan

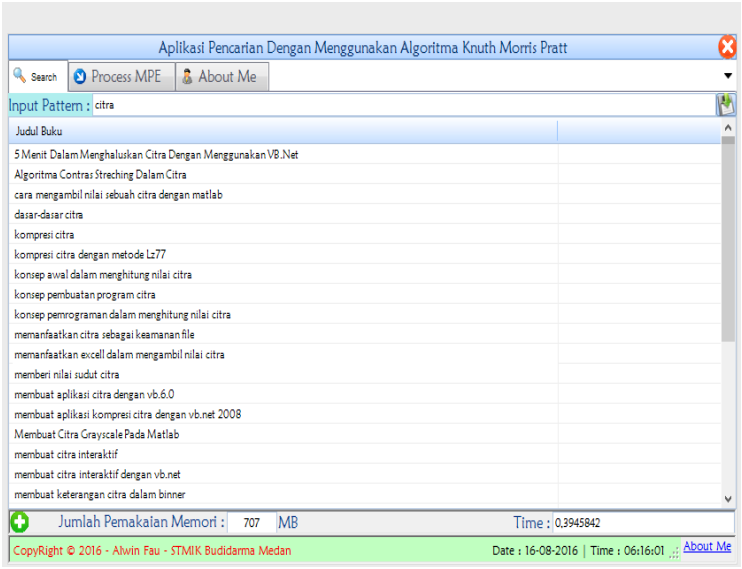
Berikut ini hasil perncarian judul buku dengan algoritma *Boyer Moore* (BM). Hasil pencarian dapat dilihat pada tabel 7.

Tabel 7. Hasil Pencarian Dengan Algoritma *Boyer Moore* (BM)

No	Cari	Hasil	Status
1	Citra	Pengolahan Citra Digital	Ketemu
		Kompresi Citra	Ketemu
		Dasar - Dasar Ditra	Ketemu
		Algoritma Contras Strching Dalam Citra	Ketemu
		Membuat Citra Interaktif	Ketemu
		Menghitung Pinggiran Citra	Ketemu
		Penghalusan Citra	Ketemu
2	Digital	Pengolahan Citra Digital	Ketemu
3	Citras	-	Tidak Ketemu
4	Pemrograman	Konsep Pemrograman Dalam Menghitung nilai Citra	Ketemu
		Pemrograman Animasi Dengan Flash 8.0	Ketemu
		Pemrograman Client Server	Ketemu
		Pemrograman Jaringan	Ketemu
		Pemrograman Java	Ketemu
5	Animasi	Membuat Animasi Sederhana Dengan Vb.Net	Ketemu
		Pemrograman Animasi Dengan Flash 8.0	Ketemu
6	Algoritma	Algoritma Contras Strching Dalam Citra	Ketemu
7	Citra Digital	Pengolahan Citra Digital	Ketemu

Implementasi Algoritma Knuth Morris Pratt (KMP)

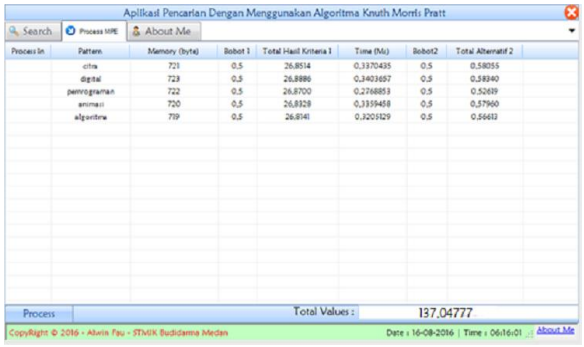
Pada implementasi algoritma *Knuth Morris Pratt* menampilkan proses pencocok *string* yang telah dilakukan pencocokan antara *pattern* yang telah diinput dengan *field* judul dari tabel *Book* yang terdapat pada *database* DBSearchBMKMP. Implementasi algoritma *Knuth Morris Pratt* (KMP) dalam pencocokan *string* pada analisa perbandingan dalam pencarian judul buku, dapat dilihat pada gambar 3.



Gambar 3. Hasil Pencarian *Pattern* dalam teks setelah ditemukan dengan algoritma *Knuth Morris Pratt*

Perhitungan Hasil Pencarian Algortima *Knuth Morris Pratt* Dengan MPE

Hasil perhitungan total nilai dari algortima *Knuth Morris Pratt* dapat dilihat pada gambar 4 dibawah ini :



Gambar 4. Hasil Total Nilai Pencarian *Knuth Morris Pratt* dalam teks setelah ditemukan

Berikut ini hasil perncarian judul buku dengan algoritma *Knuth Morris Pratt* (KMP). Hasil pencarian dapat dilihat pada tabel 8.

Tabel 8. Hasil Pencarian Dengan Algoritma K*nuth Morris Pratt* (KMP)

No	Cari	Hasil	Status
1	Citra	Pengolahan Citra Digital	Ketemu
		Kompresi Citra	Ketemu
		Dasar - Dasar Ditra	Ketemu
		Algoritma Contras Strching Dalam Citra	Ketemu
		Membuat Citra Interaktif	Ketemu
		Menghitung Pinggiran Citra	Ketemu
		Penghalusan Citra	Ketemu
2	Digital	Pengolahan Citra Digital	Ketemu
3	Citras	-	Tidak Ketemu
4	Pemrograman	Konsep Pemrograman Dalam Menghitung nilai Citra	Ketemu
		Pemrograman Animasi Dengan Flash 8.0	Ketemu
		Pemrograman Client Server	Ketemu
		Pemrograman Jaringan	Ketemu
		Pemrograman Java	Ketemu
5	Animasi	Membuat Animasi Sederhana Dengan Vb.Net	Ketemu
		Pemrograman Animasi Dengan Flash	Ketemu
6	Algoritma	Algoritma Contras Strching Dalam Citra	Ketemu

5. Kesimpulan

Berdasarkan hasil penelitian tentang analisa perbandingan Algoritma *Boyer Moore* dan Algoritma *Knuth Morris Pratt* (KMP) dengan menggunakan Metode Perbandingan Eksponensial (MPE) dapat disimpulkan, sebagai berikut :

1. Untuk dapat menganalisa Algoritma *Boyer Moore* dan Algoritma *Knuth Morris Pratt* (KMP), nilai perbandingannya dihitung dengan menggunakan Metode Perbandingan Eksponensial (MPE). Dalam perhitungannya yang menjadi kriteria perbandingan dari MPE adalah jumlah memori yang digunakan dan besarnya waktu yang dibutuhkan dari setiap proses pencocokan, setiap nilai yang didapat dari setiap kriteria kemudian di pangkatkan dengan bobot dari setiap kriteria sehingga mendapat nilai tetap yang menentukan algoritma mana yang menjadi algoritma tercepat.
2. Dengan adanya total nilai dari Metode Perbandingan Eksponesal (MPE) yang dilakukan pada penelitian ini maka terbukti bahwa Algoritma Boyer Moore merupakan Algoritma yang tercepat dalam melakukan perencarian.

Dari penelitian yang dilakukan oleh penulis maka dianggap perlu adanya saran yang penulis sampaikan kepada penulis selanjutnya agar penelitian ini tidak berhenti pada tahap ini melainkan akan terus dilanjutkan sebagai konsep penelitian yang ilmiah. Berikut beberapa hal yang perlu disarankan :

1. Dalam perbandingan dengan menggunakan Metode Perbandingan Eksponensial (MPE) disarankan algortima yang digunakan atau yang dibandingkan dapat ditambahkan dengan algoritma-algoritma *String Matching* lainnya.
2. Diharapkan agar dapat menjadi referensi dan bahan pembelajaran untuk melakukan penelitian dengan objek yang berbeda.

Daftar Pustaka

[1] R. Samo, Y. Anistyasari, and R. Fitri, *Simantic Search*. Yogyakarta: Andi, 2012.

[2] 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.

[3] G. L. Ginting, "Implementasi Algoritma Boyer-Moore Pada Aplikasi Pengajuan Judul Skripsi Berbasis Web," *Pelita Inform.*, 2014.

[4] Mesran, "IMPLEMENTASI ALGORITMA BRUTE FORCE DALAMPENCARIAN DATA KATALOG BUKU PERPUSTAKAAN," *Maj. Ilm. INTI*, vol. 3, no. 1, pp. 100–104, 2014.

[5] dan J. M. Jon Orwant, Jarkko Hietaniemi, *Mastering Algorithms R'ith Perl*. O'Reilly, 1999.

[6] K. W. Argakusumah and S. Hansun, "Implementasi Algoritma Boyer Moore Pada Aplikasi Kedokteran Berbasi Android," 2011.

[7] 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.

[8] Marimin, *Teknik dan Aplikasi Pengambilan keputusan dengan Kriteria majemuk*. 2005.

[9] Didie Nanda Pribadi, "Sistem Pendukung Keputusan Pemberian Reward kepada Karyawan Menggunakan Metode Perbadingan Eksponensial."