Variasi-Variasi Algoritma Boyer-Moore dan perbandingannya dalam pencarian String

Flora Monica Mirabella 13510094[1]
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
[1]13510094@std.stei.itb.ac.id

***Abstraksi*—pattern matching merupakan salah satu algoritma yang diperlukan dalam banyak hal di bidang informatika khususnya dalam pencarian string. Salah satu algoritma pattern matching yang sering digunakan untuk diimplementasikan dalam program adalah algoritma Boyer-Moore. Boyer-Moore dianggap sebagai salah satu algoritma pencarian string paling efektif dan telah memiliki banyak varasi yang menyempurnakan kerja algoritma ini. Variasi tersebut diantaranya adalah algoritma Turbo-BM, Apostolico-Giancarlo, dan Horspool.**

***Index Terms*—pattern matching, Boyer-Moore, variasi, algoritma pencarian string.**

I. Latar Belakang

Algoritma telah menjadi bagian yang tidak terpisahkan di bidang ilmu informatika. Banayk algoritma yang telah berkembang dari dulu, sejak jaman komputer masih berukuran sebesar satu ruangan. Dari algoritma yang digunakan untuk menentukan optimasi suatu persoalan sampai algoritma pencarian string.

Apapun jenis algoritma tersebut, tujuan utama pembuatannya adalah untuk meningkatkan tingkat keefisiensian dalam penyelesaian suatu persoalan. Dengan adanya algoritma yang baru, maka diharapkan dapat membantu memecahkan persoalan khususnya persoalan kecepatan proses komputer yang selalu dibutuhkan untuk berkembang.

Perkembangan tersebut juga termasuk di dalam kemajuan teknologi aplikasi yang memanfaatkan algortima-algorita yang ada. Banyak perusahaan pengembang perangkat lunak, berlomba-lomba membuat sebuah aplikasi yang menarik dan juga yang paling baik menerapkan strategi algortima yang paling mangkus.

Salah satu aplikasi yang sedang sangat digemari dan berkembang sekarang adalah aplikasi yang dibantu dari pemanfaatan algoritma pattern matching/string matching. Mulai dari aplikasi yang dapat digunakan untuk berkomunikasi seperti *chatbox*, sampai aplikasi yang digunakan sebagai mesin pencari (*search engine*).

Dengan adanya kebutuhan akan algoritma-algoritma yang lebih mangkus dari perusahaan perangkat lunak lainnya yang menyediaka aplikasi sejenis, maka sekarang ini banyak sekali algoritma-algoritma baru yang muncul. Untuk algoritma-algoritma pencarian string saja kurang lebih sudah terdapat sebanyak 35 algoritma yang memiliki implementasi yang berbeda satu dengan lainnya.

Algoritma-algoritma tersebut seringkali muncul dari proses penyederhanaan dari algoritma sebelumnya sehingga, dengan adanya proses penyederhanaan tersebut, didapatkan sebuah algoritma yang lebih mangkus. Selain itu, berasal dari ide algoritma yang sebelumnya, banyak juga algoritma yang muncul sehingga bisa dikatakan menjadi variasi dari algoritma yang ada.

Algoritm pencarian string sendiri merupakan algoritma yang paling penting dalam pemrosesan teks. Algoritma ini juga merupakan komponen dasar dalam implementasi perangkat-perangkat lunak dalam kebanyakan sistem operasi yang ada saat ini. Algoritma pencarian string ini dapat dibagi menjadi dua jenis yaitu algoritma yang mencari string dari kiri-ke-kanan dan juga algoritma yang mencari string dari kanan-ke-kiri.

Algoritma Boyer-Moore merupakan algoritma pencarian string jenis kedua, yaitu algoritma pencarian string dari kanan-ke-kiri. Algoritma ini telah dianggap oleh banyak pakar informatika menjadi algoritma pencarian string yang paling efisien dalam penggunaan sehari-hari. Hal ini mungkin yang menjadikan algoritma Boyer-Moore memiliki banyak sekali variasi dan penyederhanaannya. Variasi-variasi tersebut tidak hanya berbeda namun dapat juga dikatakan menyempurnakan algritma ini sehingga menjadi lebih mangkus.

II. Algoritma Boyer-Moore

Nama Boyer-Moore diambil dari nama pembuat algoritma ini. Pembuatnya adalah Robert S. Boyer dan J Strother Moore. Sebelum melakukan pencarian string, algoritma ini melakukan proses terlebih dahulu ke string yang akan dicari, bukan string yang menjadi teks tempat pencarian. String yang dicari ini akan menjadi pattern untuk pencarian dalam teks tempat pencarian. Konsep ini sangat cocok untuk diaplikasikan pada teks yang tidak bertahan dibeberapa pencarian.

Dengan penggunaan proses pada string yang dicari ini maka, algoritma ini tidak harus mencari disemua karakter yang ada pada teks tempat string dicari. Dalam prosesnya, perbandingan yang dilakukan bisa melompat di dalam teks sehingga tidak terjadi proses iterasi perbandingan

satu per satu antar karakter yang ada.

Lompatan tersebut terbagi menjadi dua jenis pada algortima ini. Pertama adalah lompatan yang diakibatkan oleh akhiran-baik dan yang kedua disebabkan alaha karakter-buruk. Akhiran-baik yang dimaksudkan disini adalah akhiran pada pattern yang saat dilakukan perbandingan sama dengan karakter pada teks.

Konsep akhiran dipakai pada algoritma Boyer-Moore karena pencarian string dilakukan dari kanan-ke-kiri. Dimana pada penulisan romawi yang dipakai oleh kita, teks dibaca dari kiri-ke-kanan, sehingga pencarian dari kanan berarti pencarian dimulai dari akhiran dari suatu kata.

Karakter-buruk yang dimaksudkan disini adalah karakter yang tidak sama dengan karakter yang ada didalam teks. Dengan ditemukannya karakter-buruk ini maka pergeseran pattern akan berbeda.

Kita akan melakukan algoritma Boyer-Moore pada teks : gcatcgcagagagtatacattacg. Dengan pattern: gcagagag. Proses dari pencarian tersebut dilakukan dengan pertama-tama memproses pattern terlebih dahulu. Proses ini akan kita sebut dengan proses prebm, yang akan menaruh hasilnya dalam array prebm[]. Pattern diproses dengan mencari nilai kemunculan terakhir dari setiap karakter yang ada di dalam pattern. Perhitungan itu dilakukan dari karakter paling belakang.

Karakter	A	C	G	T
prebm	1	6	0	8

Tabel 1proses prebm pada pattern

Jika dilihat dari pattern, dari paling kanan ada karakter g yang pertama kali kita temukan, maka karena pattern memiliki panjang 8, kemunculan terakhir g kita isi dengan 0. Lalu setelah g, kita menemukan a, sehingga kemunculan terakhir a kita isi 1 dan C ada di bagian ke- 6. Sedangkan t yang tidak ada pada pattern namun ada pada teks kita isikan dengan 8 dimana 8 sudah diluar dari pattern. Ingat walaupun dinamakan kemunculan terakhir pada pattern, tetapi karena kita akan memproses dari sebelah kanan maka kita memberikan angka paling kecil dari paling kanan.

Setelah melakukan proses ini, kita akan memulai proses pencarian di dalam teks. Pertama-tama, kita menempatkan pattern yang memiliki panjang 8 pada 8 karakter pertama di teks, sehingga setiap karakter pada pattern memiliki pasanga karakter pada teks. Lalu kita memulai perbandingan kita.

teks	g c a t c g c a g a g a g t a t a c a t t a c g
Pattern	_ _ _ _ _ _ _ G

Tabel 2 perbandingan pertama Boyer-Moore

Pertama kita menemukan A pada teks dan G pada pattern maka karena terjad ketidaksamaan, kita akan melakukan lompatan karakter-buruk pada pattern dengan mencari karakter A yang ada pada teks di pattern. Karakter A ditemukan tepat setelah G, maka kita akan memindahkan posisi A pada pattern dengan posisi A pada teks.

teks	g c a t c g c a g a g a g t a t a c a t t a c g
Pattern	_ _ _ _ _ _ G A G

Tabel 3 pemrosesan dengan Boyer-Moore

Pada tabel 3 kita dapat melihat bahwa 2 karakter pertama pada pattern yang diperiksa dari kanan sama, maka perbandingan akan terus dilanjutkan. Tetapi saat menemukan karakter ketiga, karakter teks ditemukan C dan pada pattern ditemukan G, maka terjadi ketidaksamaan. Dengan adanya ketidaksamaan ini maka perlu dilakukan lompatan pattern yang diakibatkan oleh karakter-buruk. Sebelum melakukan lompatan, ingat bahwa kita sudah melakukan 4 kali perbandingan karakter.

Lompatan karakter-buruk ini dilakukan dengan cara mencari karakter teks yang tidak sama dengan pattern yaitu katrakter C, didalam pattern yang kita miliki. Ternyata karakter tersebut dtemukan di posisi kedua dari awal pattern. Oleh karena itu kita akan melakukan lompatan pattern dengan menyamakan posisi C pada pattern dengan posisi C pada teks.

teks	g c a t c g c a g a g a g t a t a c a t t a c g
Pattern	g c a g a g a g

Tabel 4 peletakan karakter yang sama dalam teks dan pattern

Setelah meletakan, maka kita akan melakukan perbandingan kembali dari karakter pattern yang paling kanan.

teks	g c a t c g c a g a g a g t a t a c a t t a c g
Pattern	G C A G A G A G

Tabel 5 perbandingan dari kanan setelah lompatan

Ternyata yang ditemukan pada perbandingan adalah semua karakter pada pattern sama dengan teks. Maka kita telah melakukan 12 perbandingan sampai saat ini. Sekarang kita akan meneruskan perbandingan dengan memindahkan posisi patern sebanyak 7 karakter karena kita telah berhasil menemukan kesamaan pada seluruh karakter pada pattern. Perpindahan ini yang kita sebut dengan perpindahan akhiran-baik.

teks	g c a t c G C A G A G A G t a t a c a t t a c g
Pattern	_ _ _ _ _ _ _ _ g

Tabel 6 perbandingan terakhir

Lalu, perbandingan akan dilakukan lagi. Pada posisi ini, terjadi ketidaksamaan pada perbandingan antara teks dan pattern. Maka kita akan melakukan pencarian karakter T pada pattern. Karena tidak ditemukan dan sebelumnya pada proses prebm kita telah menentukan

nilai T adalah 8, maka kita akan melakukan 8 perpindahan. Namun karena karakter pada teks telah habis, tidak sampai 8, maka pencarian berakhir.

Dengan begitu dapat kita lihat kita telah melakukan 13 perbandingan untuk pemrosesan dengan Boyer-Moore ini. Telah dibuktikan juga bahwa pattern ditemukan pada teks.

Algoritma Boyer-Moore meberikan nilai kompleksitas waktu untuk proses prebm sebanyak O(m+σ) dan proses pencarian sebanyak O(mn), dimana m merupakan panjang dari pattern, σ merupakan jumlah dari kemunculan terkahir semua karakter dalam pattern, dan n adalah banyaknya karakter dalam teks.

III. VARIASI ALGORITMA BOYER-MOORE

A. Algoritma Turbo-BM

Algoritma ini menggunakan konsep Boyer-Moore sebagai ide awalanya. Pada algoritma ini tidak diperlukan proses tambahan lagi. Proses awalan pada pattern sama dengan proses awalan pada pattern yang dilakukan dengan Boyer –Moore. Namun proses pencarian yang dilakukan sedikit berbeda. Proses pencarian dilakukan dengan cara menyimpan sebuah variabel yang menjadi faktor pengingat dari teks yang sama dengan akhiran dari pattern dari proses perbandingan terakhir (dan jika dilakukan juga proses pergeseran oleh akhiran-baik).

Perpindahan yang akan dilakukan selama pencarian string juga akan berbeda dengan perpindahan dari Boyer-Moore. Perpindahan ini akan disebut dengan perpindahan turbo.

Perpindahan turbo ini hanya dapat dilakukan jika pada saat melakukan perbandingan pada saat tersebut, panjang dari karakter yang sama antara pattern dan teks lebih besar dari panjang dari karakter sama yang telah disimpan sebelumnya di variabel faktor pengingat.

Dengan begitu, proses perbandingan antara karakter akan jauh lebih sedikiti dibandingkan dengan perbandingan di Boyer Moore. Untuk lebih jelasny mari kita lakukan pengamatan dengan soal yang sebelumnya telah kita selesaikan dengan Boyer-Moore.

Karena Turbo-BM melakukan prebm yang sama dengan Boyer-Moore, maka hasil dari tabel prebm tidak akan berubah. Pertama-tama seperti pada Boyer-Moore, kita akan meletakan pattern bersesuaian dengan 8 karakter pertama didalam teks selanjutnya kita lakukan perbandingan dari kanan.

teks	g c a t c g c a g a g a g t a t a c a t t a c g
Pattern	_ _ _ _ _ _ _ G

Tabel 7 perbandingan pertama dengan TurboBM

Setelah itu maka akan dilakukan pergeseran dengan karakter-buruk seperti pada Boyer-Moore dan akan ditemukan perbandingan selanjutnya sampai ditemukan perbandingan karakter buruk lagi.

teks	g c a t c g c a g a g a g t a t a c a t t a c g
Pattern	_ _ _ _ _ _ G AG

Tabel 8 karakter-buruk kedua pada Turbo BM

Perlu diingat, kita akan memiliki fariabel faktor yang menyimpan karakter yang sama. Mari kita sebut variabel tersebut sebagai v, maka kita telah menemukan 2 karakter yang sama pada perbandingan kali ini. v akan menyimpan nilai 2. Sampai saat ini, perbandingan yang telah dilakukan sebanyak 4 kali.

Setelah melakukan peletakan dengan posisi yang sama antara karakter C si pattern dan di teks maka kita akan melakukan perbandingan lagi.

teks	g c a t c g c a g a g a g t a t a c a t t a c g
Pattern	GC _ _ AGAG

Tabel 9 perbandingan dengan menggunakan variabel v

Pertama kita akan menemukan bahwa karakte g dan a sama seperti pada perbandingan sebelumnya, lalu ditemukan karakter sama berikutnya yaitu G dan A. Maka karena sampai saat ini jumlah perbandingannya telah lebih banyak dari jumlah di v, yaitu sudah ada 4 maka akan dilakukan perbandingan akan dilompat ke tiga karakter berikutnya. Ditemukan karakter G dan C yang sama dengan teks. Oleh karena itu, akan langsung dilakukan lompatan Turbo. Sampai saat ini perbandingan yang dilakukan sudah 10 kali.

teks	g c a t c GCAGAGAGt a t a c a t t a c g
Pattern	_ _ _ _ _ _ _g

Tabel 10 setelah melakukan lompatan turbo

Lompatan turbo dilakukan dan ada perbandingan selanjutnya yang menemukan ketidakcocokan. Sama seperti pada Boyer-Moore akan dilakukan perpindahan 8 karakter sekaligus dan karena panjang teks telah habis, maka pencarian akan berakhir sampai disini.

Dengan melakukan algoritma TurboBM ini, ditemukan bahwa perbandingan yang dihasilkan lebih sedikit dibaningkan dengan Boyer-Moore, yaitu sebanyak 11 kali.

Hasil kompleksitas waktu yang diperlukan juga lebih sedikit yaitu untuk melakukan proses pertama pada pattern adalah O(m+σ) dan proses pada pencarian sebanyak O(n).

B. Algortima Apostolico-Giancarlo

Algoritma ini juga menggunakan ide dasar algoritma Boyer-Moore. Setelah mengetahui cara kerja dari algoritma TurboBM maka saat mempelajari algorima Apostolico-Giancarlo ini, kita akan menemukan kesamaan pada keduanya.

Sama seperti TurboBM, algoritma ini juga memiliki sebuah variabel yang menyimpan akhiran yang sama. Akhiran yang sama ini disimpan dalam sebuah array yang

menyimpan semua karakter sama yang telah ditemukan. Pada prosesnya nanti, tidak semua karakter yang akan dilakukan perbandingan. Karakter yang sama yang pernah dibandingkan dan telah disimpan tidak akan diulang lagi perbandingannya.

Pada dasarnya TurboBM menghitung panjang karakter sama yang pernah dibandingkan sedangkan Apostolico-Giancarlo menyimpan pattern dari perbandingan yang sama tersebut agar perbandingan tersebut tidak terjadi lagi.

Pertama-tama akan disamakan posisi pattern dengan 8 karakter pertama pada teks.

teks	g c a t c g c a g a g a g t a t a c a t t a c g
Pattern	_ _ _ _ _ _ _ G

Tabel 11 perbandingan pertama dengan Apostolico-Giancarlo

Setelah itu maka akan dilakukan pergeseran dengan karakter-buruk seperti pada Boyer-Moore dan akan ditemukan perbandingan selanjutnya sampai ditemukan perbandingan karakter buruk lagi.

teks	g c a t c g c a g a g a g t a t a c a t t a c g
Pattern	_ _ _ _ _ _ _ G A G

Tabel 12 karakter-buruk kedua pada Apostolico-Giancarlo

Perlu diingat, kita akan memiliki tabel yang menyimpan karakter yang sama. Mari kita sebut tabel tersebut sebagai skip, maka kita telah menemukan 2 karakter yang sama pada perbandingan kali ini. tabel akan menyimpan 2 karakter tersebut. Sampai saat ini, perbandingan yang telah dilakukan sebanyak 4 kali.

Setelah melakukan peletakan dengan posisi yang sama antara karakter C si pattern dan di teks maka kita akan melakukan perbandingan lagi.

teks	g c a t c g c a g a g a g t a t a c a t t a c g
Pattern	G C _ _ A G A G

Tabel 13 perbandingan dengan menggunakan tabel skip

Pertama kita akan menemukan bahwa karakte g dan a sama seperti pada perbandingan sebelumnya, lalu ditemukan satu karakter berikutnya lagi yang sama yaitu g. Maka karena sampai saat ini terdapat pattern baru di tabel skip yaitu AGAG, yaitu sudah ada 4 dan karakter selanjutrnya sama dengan maka akan dilakukan perbandingan ke karakter selanjutnya yang tidak sama. Ditemukan karakter G dan C yang sama dengan teks. Oleh karena itu, akan langsung dilakukan lompatan 7 karakter seperti pada lompatan akhiran-baik pada Boyer-Moore. Sampai saat ini perbandingan yang dilakukan sudah 10 kali.

teks	g c a t c G C A G A G A G t a t a c a t t a c g
Pattern	_ _ _ _ _ _ _ _ g

Tabel 14 setelah melakukan lompatan akhiran-baik

Setelah itu proses akan sama dilakuakan seperti proses akhir TurboBM. Sehingga didapatkan banyak perbandingan yang sama yaitu 11 kali perbandingan.

Kompleksitas waktu yang digunakan untuk melakukan prose prebm adalah $O(m+\sigma)$ dan untuk melakukan pencarian pada teks adalah $O(m)$. Selain itu dapat dilihat juga bahwa denagn algoritnma ini ditemukan string yang ingin dicari.

C. Algoritma Horspool

Algoritma ini hanya mempergunakan perpindahan karakter-buruk yang terjadi pada Boyer-Moore. Dengan bigut, algoritma ini jauh lebih sederhana dibandingkan dengan Boyer-Moore.

Pada awalnya, pattern juga diproses dahulu dengan proses preBM, hanya saja ada sedikit perbedaan, perhitungan 0 tidak dimasukan sehingga nilai dari g adalah 2 bukannya 0.

Karakter	A	C	G	T
prebm	1	6	2	8

Tabel 15 prebm dari Horspool

Selain itu, perpindahan akhiran-baik tidak akan digunakan pada algoritma ini.

Perbedaan lain yang terdapat pada algoritma ini adalah, perbandingan tidak hanya dilakukan dari kanan pattern namun juga dilakukan dari kiri pattern. Saat ditemukan kesamaan pada karakter ertama dari kanan yang diperiksa pada pattern, maka akan diperiksa lagi karakter pertama dari sebelah kiri pattern. Maka, algoritma ini seperti menggabungkan algoritma Boyer-Moore dan algoritma KMP.

Pada proses pertama akan dilakukan penyamaan posisi pattern dengan 8 karakter pertama pada teks.

teks	g c a t c g c a g a g a g t a t a c a t t a c g
Pattern	_ _ _ _ _ _ _ G

Tabel 16perbandingan pertama dengan Horspool

Melalui perbandingan ini maka akan dilakukan perpindahan karakter buruk seperti pada Boyer-Moore. Lalu akan ada perbandingan per karakter lagi.

teks	g c a t c g c a g a g a g t a t a c a t t a c g
Pattern	g _ _ _ _ _ _ G

Tabel 17 perbandingan dengan karakter pertama yang sama

Karena didapatkan karakter yang sama dengan yang pertama pada teks, maka akan diperiksa karakter paling terakhir dari kanan pada pattern. Karena tidak ditemukan kesamaan aka akan dicari karakter G pada teks, yang

sama dengan karakter G paling awal di dalam pattern. Ditemukan dengan memindahkan 2 karakter ke depan sehingga menjadi:

teks	g c a t c g c a g a g a g t a t a c a t t a c g
Pattern	G C A G A G A G

Tabel 18 perbandingan lengkap dengan Horspool

Melalui perpindahan ini, maka akan dilakukan perbandingan lagi dari sebelah kanan pattern yang hasilnya sama, dan loncat ke perbandingan dari kiri. Karena hasil sama, maka perbandingan dari kiri ke kanan pattern akan terus dilakukan selama hasilnya sama. Setelah itu ditemukan bahwa terdapat pattern tersebut di dalam teks. Perlu diingat bahwa sampai saat ini kita telah melakukan 11 perbandingan.

Setelah itu pattern akan pindah 2, karena teks pertama yang diperiksa dengan pattern adalag G dan bernilai 2 pada tabel preBM.

teks	g c a t c g c a g a g a g t a t a c a t t a c g
Pattern	_ _ _ _ _ _ _ g

Tabel 19 berpindah dua karakter

Karena tidak ditemukan perbandingan yang hasilnya sama, maka akan dilakukan perpindahan karakter-buruk lagi, dimana karakter A yang tidak sama pada teks bernilai 1, maka dilakukan perpindahan satu ke kanan. Perpindahan ini juga menghasilkan perbandingan yang hasilnyatidak sama tetapi dengan karakter T pada teks. Karena T bernilai 8 pada tabel prebm maka pattern akan dilompat sebanyak 8 karakter pada teks.

teks	g c a t c g c a g a g a g t a t a c a t t a c g
Pattern	g __ __ __G

Tabel 20 pelompatan 8 karakter

Setelah pelompatan terjadi, ditemukan perbandingan pertama antara teks dan pattern sama. Maka akan diperiksa karakter pertama dari kiri pattern. Karena tidak ditemukan kesamaan maka yang dilakukan seharusnya melakukan perpindahan. Tetapi karena teks telah habis maka pencarian berakhir. Sampai saat ini jumlah perbandingan yang terjadi sudah sampai 17 kali perbandingan.

Waktu kompleksitas yang dibutuhkan pada algoritma ini adalah O(m+σ) dan O (σ) untuk menghitung prebm dan memerlukan O(mn) untuk melakukan pencarian.

IV. PERBANDINGAN

Dari proses diatas dapat kita dapatkan perbandingan antara proses yang ada dalam mengejakan satu pencarian string yang sama. Perbandingan tersebut dapat dilihat pada tabel dibawah ini:

Algoritma	Total Kompleksitas	Total perbandingan
Boyer-Moore	O(m+σ) + O(mn)	13
TurboBM	O(m+σ) + O(n)	11
Apostolico-Giancarlo	O(m+σ) + O(n)	11
Horspool	O(m+σ)+O(σ)+O(mn)	17

Dari tabel diatas, dapat dilihat bahwa Boyer-Moore masih memiliki banyak kekurangan jika dibandingkan dengan TurboBM dan Apostolico-Giancarlo. Dimana Boyer-Moore melakuakn perbandingan yang sama berkali-kali tanpa menyimpan nilai karakter yang pernah dibandingkan sebelumnya, sehingga perbandingan sebelumnya menjadi kurang dimanfaatkan.

Dengan algoritma TurboBM dan Apostolico-Giancarlo, perbandingan sebelumnya yang telah dipakai menjadi bermanfaat dalam penambahan tingak keefisienan algoritma. Sehingga perbandingan yang didapatkan juga menjadi lebih sedikit dibandingkan Boyer-Moore. Namun, algoritma pencariannya pun menjadi lebih sulit dibandingkan dengan algoritm Boyer-Moore biasa.

Algoritma Horspoole yang tida menggunakan perpindahan akhiran-baik tidak memiliki keefisieansian yang lebih baik dari Boyer-Moore. Dengan algoritma ini perbandingan malah terjadi lebih banyak dibandingkan dengan Boyer-Moore, apalagi dibandingkan dengan dua lainnya. hal ini sepertinya disebabkan oleh waktu komputasi pada pencarian proses prebm yang memerlukan lebih banyak yaitu ditambah O(σ). Namun, algoritma pencariannya tentu menjadi lebih sederhana dibandingkan oleh algoritma Boyer-Moore.

Dengan begitu dapat dilihat bahwa variasi dari Boyer-Moore bisa membuat algoritma menjadi lebih mangkus ataupun sebaliknya.

V. KESIMPULAN

Setelah melakukan perbandingan diatas maka dapat kita lihat bahwa algoritma Boyer-Moore yang dianggap baik masih kalah mangkus dibandingkan dengan beberapa variasi turunannya yaitu algoritma TurboBM dan algoritma Apostolico-Giancarlo. Sedangakan Algoritma Horspool menghasilkan perbandingan yang lebih buruk dari Boyer-Moore.

Masih banyak lagi varias-variasi dan penyederhanaan lain dari algoritma Boyer-Moore ini yang mungkin saja juga menyediakan proses yang lebih mangkus dibandingkan tiga algoritma yang di bahas pada makalah ini.

REFERENSI

[1] Stephen, G.A., 1994, *String Searching Algorithms*, World Scientific

[2] Levitin, Anany, 2012, *Introduction to: The Design and Anaysis Algorithm*, 3rd Edition, USA: Pearson Education, Inc.

[3] http://en.wikipedia.org/wiki/Boyer%E2%80%93Moore_string_search_algorithmB. Smith, "An approach to graphs of linear forms (Unpublished work style)," unpublished. 20 Desember 2012. 13:10

[4] http://www-igm.univ-mlv.fr/~lecroq/string/node15.html#SECTION00150 .20 Desember 2012. 13:12
[5] http://www-igm.univ-mlv.fr/~lecroq/string/node16.html#SECTION00160. .20 Desember 2012. 10:10
[6] http://www-igm.univ-mlv.fr/~lecroq/string/node18.html#SECTION00180. 19 Desember 2012. 10:10

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, 21 Desember 2012

Flora Monica Mirabella (13510094)