PENCOCOKAN STRING MULTI PATTERN PADA PENGOLAH KATA MULTI PATTERN STRING MATCHING ON WORDPROCESSOR

Arliadinda Danusaputro[1], Rimba Whidiana[2], Retno Novi Dayawati[3]

[1]Teknik Informatika, Fakultas Teknik Informatika, Universitas Telkom

Abstrak

Aplikasi teks editor berkaitan erat dengan pencocokan string. String yang akan dicocokan (pattern) dibedakan menjadi dua, yaitu single-pattern dan multipattern. Saat ini pengolah kata menerapkan pencocokan string single-pattern karena setiap pattern yang dicocokan merupakan sebuah pattern. Dalam perkembangan pencarian informasi, diperlukan pencocokan string multi-pattern yang bertujuan membantu user dalam mencari informasi yang diinginkan.

Beragam ditemukan algoritma pencocokan string dan variantnya. Algoritma Knuth-morris-Pratt (KMP) merupakan salah satu algoritma pencocokan string single pattern yang tangguh di kelasnya. Akan tetapi algoritma ini harus dilakukan berulang kali sebanyak patternnya untuk pencocokan string multi pattern. Oleh karena itu algoritma ini dikembangkan menjadi algoritma Aho Corasick yang dapat melakukan pencocokan string multi pattern dalam sekali proses. Namun pada implementasi, run time KMP multi pattern lebih cepat daripada Aho Corasick. Hal ini dipengaruhi oleh jumlah pattern, panjang teks, jumlah pattern yang ditemukan pada teks

Kata Kunci : pencocokan string, single-pattern, multi-pattern, algoritma Knuth-

Abstract

Text editing aplication has a close relation with string matching. The string that will be matched is divided into single pattern and multi pattern. The existing word processors implement single pattern string matching because each pattern that is matched assumed as single pattern. In growth of information retrieval is needed multi-pattern string matching to assist user in searching wanted information.

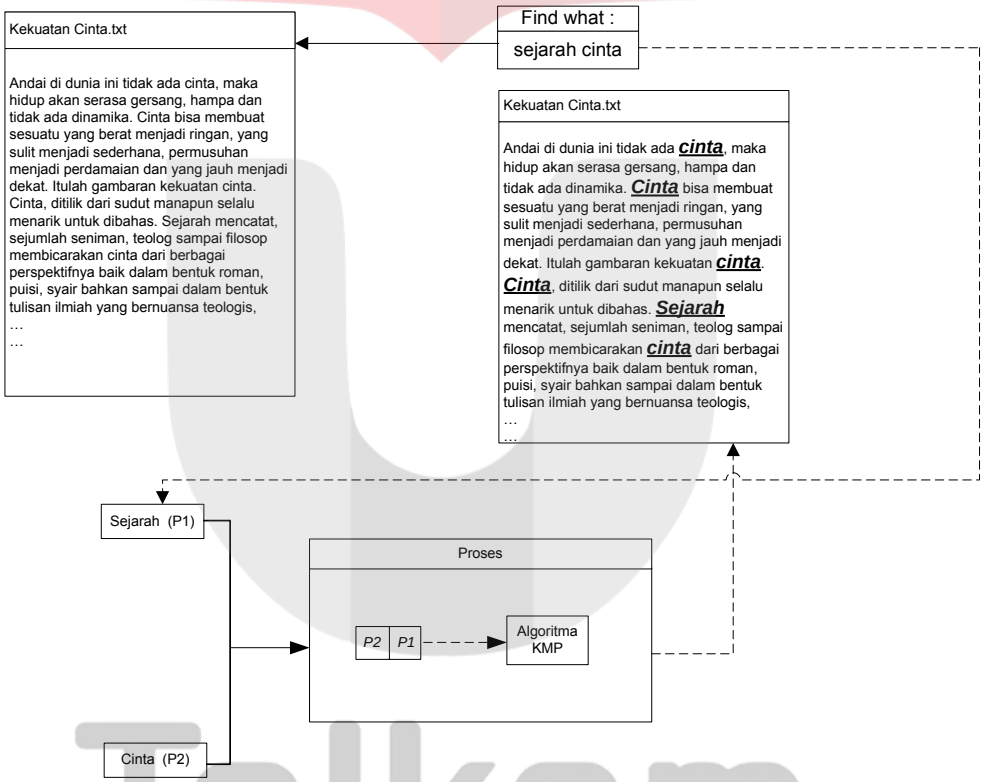
Kind of string matching algorithm is found with it's variant. Knuth- Morris-Pratt (KMP) algorithm is one of the single pattern string matching algorithm. However this algorithm must be done repeatedly counted its pattern to do multi pattern string mathcing. Therefore this algorithm is developed to become Aho Corasick algorithm able to multi pattern string matching in once process. But at implementation, quicker KMP pattern multi time run than Aho Corasick.This Matter is influenced by amount of pattern, length of text, amount of found pattern in text.

Keywords : string matching, single pattern, multi pattern, Knuth-Morris-Pratt

1. Pendahuluan

1.1 Latar Belakang

String matching merupakan pencocokan *string* yang terdapat dalam sebuah teks. Saat ini pengolah kata menerapkan pencocokan string *single-pattern* karena setiap *pattern* yang dicocokan merupakan sebuah *pattern*. Dalam perkembangan pencarian informasi, diperlukan pencocokan *string multi-pattern* yang bertujuan untuk mempermudah dalam mendapatkan *pattern* yang terkait. Pencocokan *string multi pattern* akan membantu user dalam mencari informasi yang diinginkan. Berikut ini adalah skema ilustrasi pencocokan *string multi pattern* :

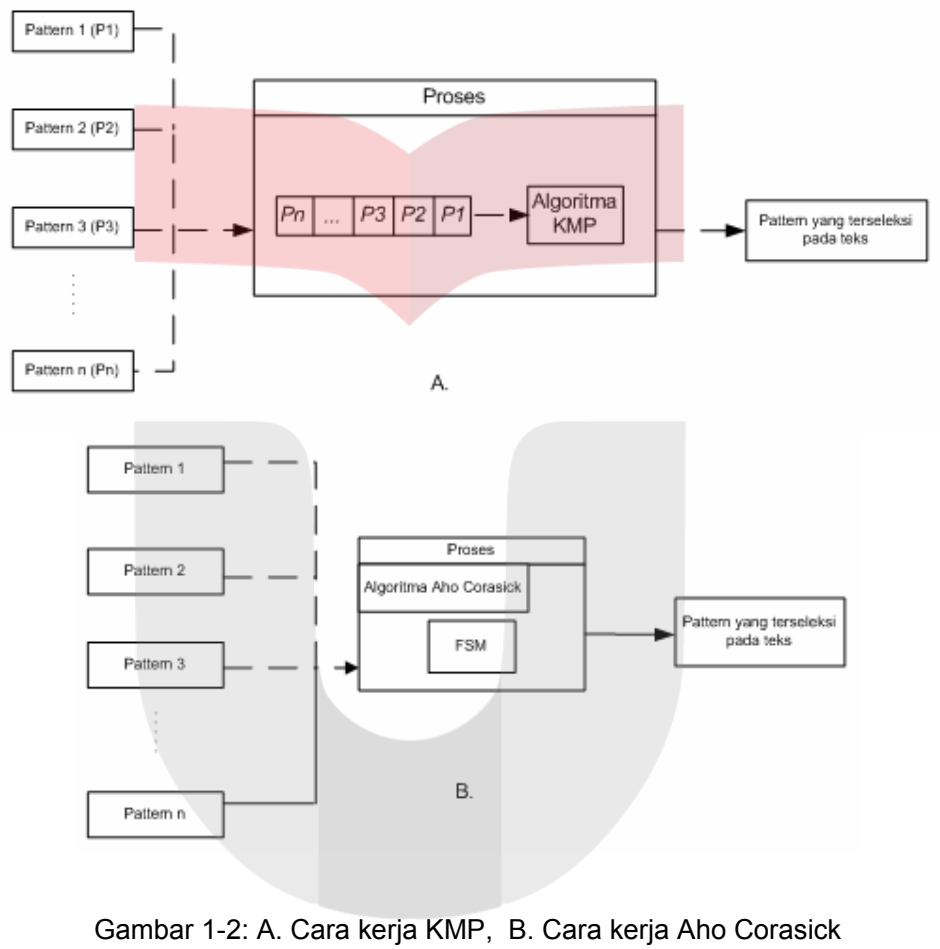


Gambar 1-1: Ilustrasi pencocokan *string multi pattern*

Banyak algoritma yang dapat digunakan untuk melakukan pencocokan string pada pengolah kata, salah satunya adalah algoritma Knuth-morris-pratt (KMP) yang mempunyai kompleksitas yang cukup baik karena menyimpan informasi untuk melakukan pergeseran pencocokan. Namun sayangnya algoritma KMP hanya berlaku untuk pencocokan string *single pattern*. Dalam kasus *multi pattern* KMP dapat dimodifikasi menjadi KMP *multi pattern,* tetapi algoritma ini perlu membaca teks sebanyak *pattern* yang akan dicocokkan. Sehingga perlu dikembangkan menjadi algoritma yang lebih efisien. Pengembangan dari algoritma KMP untuk *multi pattern* yaitu algoritma Aho Corasick (AC).

1.2 Perumusan Masalah

Permasalahan yang diangkat dalam tugas akhir ini adalah bagaimana cara pengembangan algoritma KMP menjadi Algoritma Aho Corasick. Berikut ini adalah skema proses pencocokan algoritma KMP untuk pencocokan *string multi pattern* :



Gambar 1-2: A. Cara kerja KMP, B. Cara kerja Aho Corasick

Rumusan masalah yang akan ditinjau adalah sebagai berikut :

1. Algoritma KMP hanya dapat melakukan pencocokan *string single pattern.* Untuk dapat menangani pencocokan *string multi pattern*, algoritma ini dilakukan sebanyak *pattern* yang akan dicocokkan. Pengembangan dari algoritma KMP untuk kasus tersebut adalah algoritma Aho Corasick dimana pencocokan *string multi pattern* dilakukan dalam sekali proses. Sehingga permasalahannya adalah bagaimana pengembangan algoritma yang dilakukan.
2. Bagaimana hasil implementasi pencocokan *string multi pattern* pada pengolah kata dengan menggunakan algoritma Aho Corasick dan algoritma KMP *multi pattern.*

Batasan masalah dalam tugas akhir ini adalah :

1. Teks yang digunakan berupa file bertype .txt, karena keterbatasan perangkat lunak pembangun yang membutuhkan komponen lain untuk dapat membaca file tertype selain .txt.
2. Inputan *pattern* berupa abjad [‘a’..’z’] kecuali spasi yang digunakan untuk membedakan *pattern* satu dengan yang lain, hal ini berkaitan dengan proses kerja algoritma Aho Corasick.

1.3 Tujuan Pembahasan

Tujuan dari penelitian Tugas Akhir ini adalah

1. Menganalisis pengembangan algoritma KMP menjadi algoritma Aho Corasick untuk pencocokan *string multi pattern* dari segi efisiensi algoritma.
2. Menganalisis performansi dan hasil pengujian dari implementasi algoritma KMP *multi pattern* dan Aho Corasick berdasarkan *run time* dan kompleksitas waktu asimptotik.

1.4 Metodologi Penyelesaian Masalah

Metodologi penyelesaian masalah yang digunakan dalam penelitian Tugas Akhir ini adalah

1. Studi literatur
   a. Pencarian referensi
      Mencari referensi yang berhubungan dengan algoritma KMP, algoritma Aho Corasick, metode pencocokan *string*, dan hal-hal lain yang berkaitan dengan judul pada Tugas Akhir ini.
   b. Pendalaman materi
      Mempelajari dan memahami algoritma Aho Corasick dengan menanyakan kepada Pembimbing Tugas Akhir maupun kepada teman-teman.
2. Analisis Algoritma
   Menganalisis algoritma yang akan diimplementasikan, dilihat dari cara kerja dan data yang digunakan.
3. Pembangunan perangkat lunak dengan metode terstruktur yang meliputi:
   a. Analisis sistem
      Tahap ini dilakukan pengumpulan kebutuhan elemen-elemen di tingkat perangkat lunak yang akan dibangun. Dengan analisis ini, harus dapat menentukan domain-domain data atau informasi, fungsi, proses, atau prosedur yang diperlukan beserta unjuk kerjanya.
   b. Desain sistem
      Proses desain mengubah kebutuhan-kebutuhan menjadi bentuk karakteristik yang dimengerti perangkat lunak sebelum dimulai penulisan program. Desain ini harus didokumentasikan dengan baik dan menjadi bagian konfigurasi perangkat lunak desain.

c. Pengkodean
Tahap ini dilakukan implementasi hasil rancangan ke dalam baris-baris kode program yang dapat dimengerti oleh mesin (komputer). Perangkat lunak akan diimplementasikan ke dalam bentuk program berdasarkan hasil analisa dan perancangan yang diperoleh dari tahap sebelumnya.
d. Pengujian perangkat lunak yang telah dibuat
Melakukan pengujian terhadap perangkat lunak yang telah dibuat

4. Pengevaluasian sistem dengan beberapa kasus uji.
5. Pembuatan dokumentasi perangkat lunak.

1.5 Sistematika Penulisan

Penulisan Tugas Akhir ini dibagi dalam lima bab, yang terdiri atas :

BAB I PENDAHULUAN
Berisi latar belakang, perumusan masalah, tujuan pembahasan, metodologi penyelesaian masalah dan sistematika penulisan.

BAB II LANDASAN TEORI
Bab ini membahas berbagai teori dasar pendukung implementasi Tugas Akhir ini, antara lain mengenai pencocokan string pada pengolah kata dan algoritma KMP

BAB III PEMBAHASAN MASALAH
Membahas tentang algoritma Aho Corasick dan proses pengembangan algoritma KMP menjadi Aho Corasick

BAB IV PEMBANGUNAN PERANGKAT LUNAK
Bab ini membahas kebutuhan perangkat lunak dan perangkat keras yang digunakan untuk merealisasikan sistem. Selain itu, pada bab ini akan dibahas pengujian perangkat lunak, hasil uji coba dan analisis pengujian.

BAB V KESIMPULAN DAN SARAN
Berisi kesimpulan akhir dan saran terhadap pengembangan dari penelitian Tugas Akhir ini selanjutnya.

5. Penutup

5.1 Kesimpulan

Kesimpulan yang dapat diambil dari tugas akhir pencocokan *string multi pattern* pada pengolah kata ini adalah:

1. Pengembangan algoritma KMP menjadi algoritma Aho Corasick adalah dengan mengkombinasikan *finite state machine* dan pergerakan state yang mengadopsi dari fungsi pinggiran KMP.
2. Berdasarkan analisis algoritma, kompleksitas algoritma KMP *multi pattern* adalah *O(k(n+m))* sedangkan kompleksitas algoritma Aho Corasick adalah *O*(n+m+z) dimana k adalah jumlah *pattern* yang akan dicocokan dalam teks, n adalah panjang *pattern,* m adalah panjang teks, dan z adalah jumlah *pattern* yang muncul dalam teks.
3. Berdasarkan hasil pengujian berdasarkan waktu algoritma KMP *multi pattern* lebih cepat dibandingkan dengan algoritma Aho Corasick, hal ini dipengaruhi oleh panjang teks, jumlah *pattern* yang dimasukkan dan jumlah *pattern* yang terseleksi.

5.2 Saran

Saran yang dapat diajukan untuk pengembangan dan perbaikan pencocokan *string multi pattern* pada pengolah kata ini adalah pengembangan untuk optimasi algoritma Aho Corasick yang ada saat ini.

Daftar Pustaka

[1] *Aho-Corasick algorithm*. Wikipedia-the free encyclopedia. 28 November 2006. http://en.wikipedia.org/wiki/Aho-Corasick_algorithm

[2] A. V. Aho and M. J. Corasick. *Efficient string matching: An aid to bibliographic search*. Communications of the ACM, 18(6):333–340, 1975.

[3] Haryanto Bambang. 2004. *Teori Bahasa, Otomata, dan Komputasi serta terapannya.* Bandung: Informatika.

[4] Haryanto Bambang. 2000. *Struktur Data.* Bandung: Informatika.

[5] Kilpelainen, Pekka. 2005. "Biosequence Algorithms, lecture 4: Set Matching and Aho-Corasick Algorithm". University of Kuopio Department of Computer Science

[6] Lecroq, Thierry. 1995. *Experimental Results on String Matching Algorithms*. France, Laboratoire d'Informatique de Rouen, Universite´ de Rouen, Faculte´s des Sciences

[7] *Mengenal Pengolah Kata*, e-smartschool www. i-smartschool.com/PNK/003/PNK0030004.asp

[8] Munir, Rinaldi. *Algoritma Pencarian String (String Matching).* Departemen Teknik Informatika, Institut Teknologi Bandung.

[9] Page Lawrence and Sergey Brin. *The Anatomy of a Large-Scale Hypertextual Web Search Engine.* Computer Science Department, Stanford University, tanford, CA 94305, USA sergey@cs.stanford.edu and page@cs.stanford.edu

[10] *String searching algorithm.* Wikipedia - the free encyclopedia. 8 December 2006. http://en.wikipedia.org/wiki/String_searching_algorithm

[11] Yulianto, Fazmah Arif. *A First Step Toward Algorithm Complexity Analysis*. Departemen Teknik Informatika, Sekolah Tinggi Teknologi Telkom.