Scientific Journal of Informatics
Vol. 3, No. 1, Mei 2016
p-ISSN 2407-7658 http://journal.unnes.ac.id/nju/index.php/sji e-ISSN 2460-0040

Pencarian *File* Teks Berbasis *Content* dengan Pencocokan *String* Menggunakan Algoritma *Brute force*

Danuri
Jurusan Teknik Informatika, Politeknik Negeri Bengkalis
Email: danuri@polbeng.ac.id

Abstrak

Keberadaan *file* menjadi penting saat dibutuhkan dan menjadi permasalahan apabila tidak ditemukan. Nama dari suatu *file* belum tentu memberikan gambaran isi yang terkandung pada *file*. Ini yang menjadi dasar dalam pencarian *file* berbasis konten. Terdapat beberapa algoritma untuk menyelesaikan permasalahan tersebut diantaranya algoritma *brute force*. Pengembangan algoritma pencarian dengan menciptakan pencarian lokal dan global memberikan kesempatan setiap kata pada *file* dan *file* pada lokasi yang dicari dapat diperiksa. Hasil pengujian menunjukkan rata-rata waktu proses 1 *file* sebesar 0.003847 detik dari 120 kali percobaan. Semakin banyak jumlah kata dalam suatu *file* dan jumlah *file* dalam satu tempat penyimpanan menyebabkan kebutuhan waktu semakin meningkat.

Kata Kunci: Pencarian lokal, pencarian global, algoritma *brute force*

1. PENDAHULUAN

File adalah berkas atau naskah yang disimpan pada komputer. Berkas ini dapat berupa tulisan atau teks, gambar, kumpulan instruksi program, musik video, dan sebagainya. Satu diantara aplikasi pengolah teks adalah *Microsoft Office Word* yang dapat menghasilkan dokumen laporan dan lain sebagainya [1]. *File* teks yang dihasilkan dari aplikasi tersebut diantaranya doc, docx, xls, xlsx, ppt, dan pptx.

Keberadaan suatu *file* sangat penting pada saat dibutuhkan. Teknik penyimpanan yang baik akan membantu dalam mempercepat proses penemuan suatu *file*. Namun, hal ini akan menjadi masalah apabila *file* yang dicari tidak disimpan secara baik seperti penamaan *file* yang tidak mencirikan isi dari *file* dan sudah lama tidak diakses sehingga daya ingat menurun terkait keberadaan *file* tersebut.

Dibutuhkan teknik tertentu agar *file* yang dicari dapat ditemukan dengan baik. Diantara teknik pencarian berupa pencocokan *string* antara *string* yang dicari dengan nama *file* yang tersedia di komputer. Pencarian *file* berdasarkan nama *file* belum mampu menjamin isi dari *file* sesuai dengan kebutuhan pengguna. Oleh karena itu, diperlukan algoritma pencarian *file* berdasarkan pencocokan *string* dengan *string* yang dicari dengan konten dari suatu *file*. Pencocokan *string* merupakan salah satu algoritma untuk mempercepat proses pencarian *file* berdasarkan *string* yang dimasukkan. Algoritma pencocokan *string* yang sering digunakan adalah algoritma *brute force*. Algoritma *brute force* adalah algoritma melakukan pencocokan *string* yang dinputkan dengan semua teks antara 0 dan n-m untuk menemukan keberadaan *string* yang diinputkan dalam teks [2].

2. METODE

2.1 Tinjauan Pustaka

Pencarian *file* banyak menarik minat untuk dikaji baik dengan menerapkan algoritma *brute force* untuk penyelesaian permasalahan tertentu atau dengan algoritma lainnya.

Kajian tentang pencarian data katalog buku pada perpustakaan menerapkan algoritma *brute force*. Perpustakaan merupakan salah satu tempat yang banyak digunakan mahasiswa untuk mencari informasi di dalam memecahkan suatu masalah yang ditemui pada proses pembelajaran. Jumlah buku yang sangat banyak akan memberikan masalah dalam hal pencarian data katalog buku yang terdapat pada perpustakaan. Dalam melakukan pencarian data katalog buku perpustakaan, pencocokan *string* merupakan suatu bagian dalam proses pencarian *string* (*string searching*). Hasil yang dicapai dari penerapan algoritma *brute force* dapat melakukan pencocokan *string* dan memberikan hasil yang diinginkan dsesuai dengan data pencarian pada katalog buku perpustakaan dan pengunjung perpustakaan dalam melakukan pencarian terhadap katalog buku perpustakaan dapat diselesaikan dengan waktu yang singkat [2].

Algoritma *brute force* untuk menyelesaian permasalahan *font italic* untuk kata asing secara otomatis di dalam *Microsoft Office Word*. Algoritma *brute force* akan menempatkan dan mencari semua kemungkinan kata asing dengan masukan karakter dan panjang kata asing tertentu, tentunya dengan banyak sekali kombinasi dalam kalimat-kalimat. Hasilnya yaitu algoritma *brute force* mencoba setiap posisi *pattern* terhadap teks, kemudian dilakukan proses pencocokan setiap karakter dan teks pada posisi tersebut [1].

Pencarian *string* merupakan bagian yang sangat penting dalam beberapa masalah termasuk dalam masalah edit teks, pencarian teks dan manipulasi simbol. Pengembangan pada algoritma *brute force* yang diberi nama algoritma *Start-End-Mid*. Algoritma yang diusulkan tidak melakukan pencocokan terhadap semua karakter dari *string* yang dicari melainkan melakukan pengecekan karakter pertama, akhir, dan tengah. Hasil pengembangan algoritma ini mempercepat proses pencarian [3].

2.2 Landasan Teori

2.2.1 Dokumen digital

Dokumen digital merupakan setiap informasi elektronik yang dibuat, diteruskan, dikirimkan, diterima, atau disimpan dalam bentuk *analog*, *digital*, elektromagnetik, optikal, atau sejenisnya, yang dapat dilihat, ditampilkan atau didengar melalui komputer atau sistem elektronik. Untuk dapat mendistribusikan melalui media web ada beberapa format standar yang digunakan, beberapa diantaranya yang sering digunakan adalah *Portable Document Format* (PDF), *Hypertext Markup Language* (HTML), *eXtensible Markup Language* (XML), *eXtensible Hypertext Markup Language* (XHTML) dan lain sebagainya [4]. Format standar hasil olahan dari *software* pengolah kata yaitu txt, doc, docx, xls, xlsx, ppt, dan pptx.

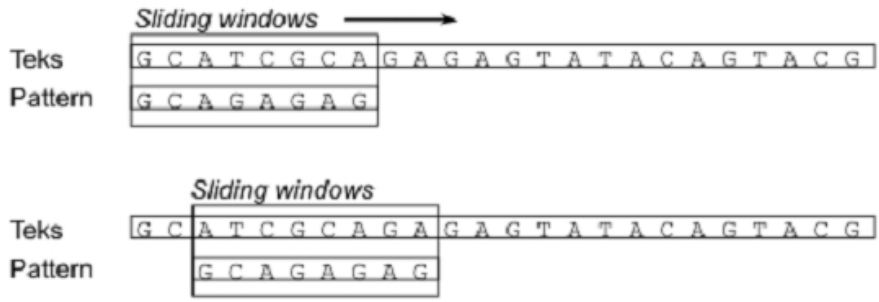
2.2.2 Pencocokan *String*

Pencocokan *string* atau *string matching* adalah proses pencarian semua kemunculan *string* pendek *P[0..n-1]* yang disebut *pattern* di *string* yang lebih panjang *T[0..m-1]* yang disebut teks. Pencocokan *string* merupakan permasalahan paling sederhana dari semua permasalahan *string* lainnya, dan merupakan bagian dari pemrosesan data, pengkompresian data, *lexical analysis*, dan temu balik informasi. Teknik untuk menyelesaikan permasalahan pencocokan *string* biasanya akan menghasilkan implikasi langsung ke aplikasi *string* lainnya [5].

String matching adalah pencarian sebuah *pattern* pada sebuah teks [6]. Prinsip kerja algoritma *string matching* adalah sebagai berikut [7]:

1. Memindai teks dengan bantuan sebuah *window* yang ukurannya sama dengan panjang *pattern*.
2. Menempatkan *window* pada awal teks.
3. Membandingkan karakter pada *window* dengan karakter dari *pattern*. Setelah pencocokan (baik hasilnya cocok atau tidak cocok), dilakukan *shift* ke kanan pada *window*. Prosedur ini dilakukan berulang-ulang sampai *window* berada pada akhir teks. Mekanisme ini disebut mekanisme *sliding-window*.

Contoh *sliding windows* yang sedang menggeser *pattern* di teks dapat dilihat di Gambar 1.



Gambar 1. Ilustrasi pergerseran pada proses pencocokan *string*

2.2.3 Algoritma *Brute Force*

Algoritma *brute force* merupakan algoritma sederhana yang dapat digunakan dalam pencarian pola. Ide dasar dari algoritma *brute force* adalah membandingkan karakter demi karakter antar *string* yang dicari dengan *string* sumber. Apabila tidak sesuai maka akan dilakukan penggeseran posisi dari kiri ke kanan. Demikian seterusnya sampai ditemukan *string* yang dicari [3].

Beberapa karakteristik dari algoritma *brute force* dapat dijelaskan sebagai berikut [8].

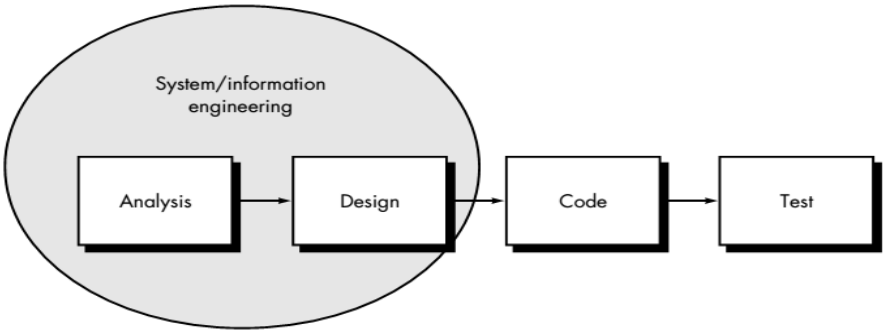
1. Membutuhkan jumlah langkah yang banyak dalam menyelesaikan suatu permasalahan sehingga jika diterapkan menjadi suatu algoritma program aplikasi akan membutuhkan banyak memori.
2. Digunakan sebagai dasar dalam menemukan suatu solusi yang lebih efektif.
3. Banyak dipilih dalam penyelesaian sebuah permasalahan yang sederhana karena kemudahan cara berpikirnya.

4. Pada banyak kasus, algoritma ini banyak dipilih karena hampir dapat dipastikan dapat menyelesaikan banyak persoalan yang ada.
5. Digunakan sebagai dasar bagi perbandingan keefektifan sebuah algoritma.

2.3 Metode

Metode *waterfall* merupakan metode yang digunakan terdiri atas analisa permasalahan, perancangan, penulisan kode program, dan pengujian [9]. Diagram dari metode *waterfall* dapat dilihat pada Gambar 2.

1. Analisa permasalahan, meliputi format *file* yang menjadi inputan, teknik pencarian dokumen teks dengan pencocokan pola, proses *indexing*.
2. Perancangan perangkat lunak, meliputi perancangan perangkat lunak, perancangan antarmuka dan perancangan *database*.
3. Penulisan kode program, penulisan program menggunakan bahasa pemrograman PHP dan untuk pengelolaan data menggunakan DBMS MySQL.
4. Pengujian, pengujian dilakukan menggunakan metode *blackbox* untuk beberapa model pengujian meliputi pengujian fungsionalitas perangkat lunak, pengujian lingkungan perangkat lunak dan pengujian waktu pencarian



Gambar 2. Model waterfall

3. HASIL DAN PEMBAHASAN

3.1 Proses Pencarian *File*

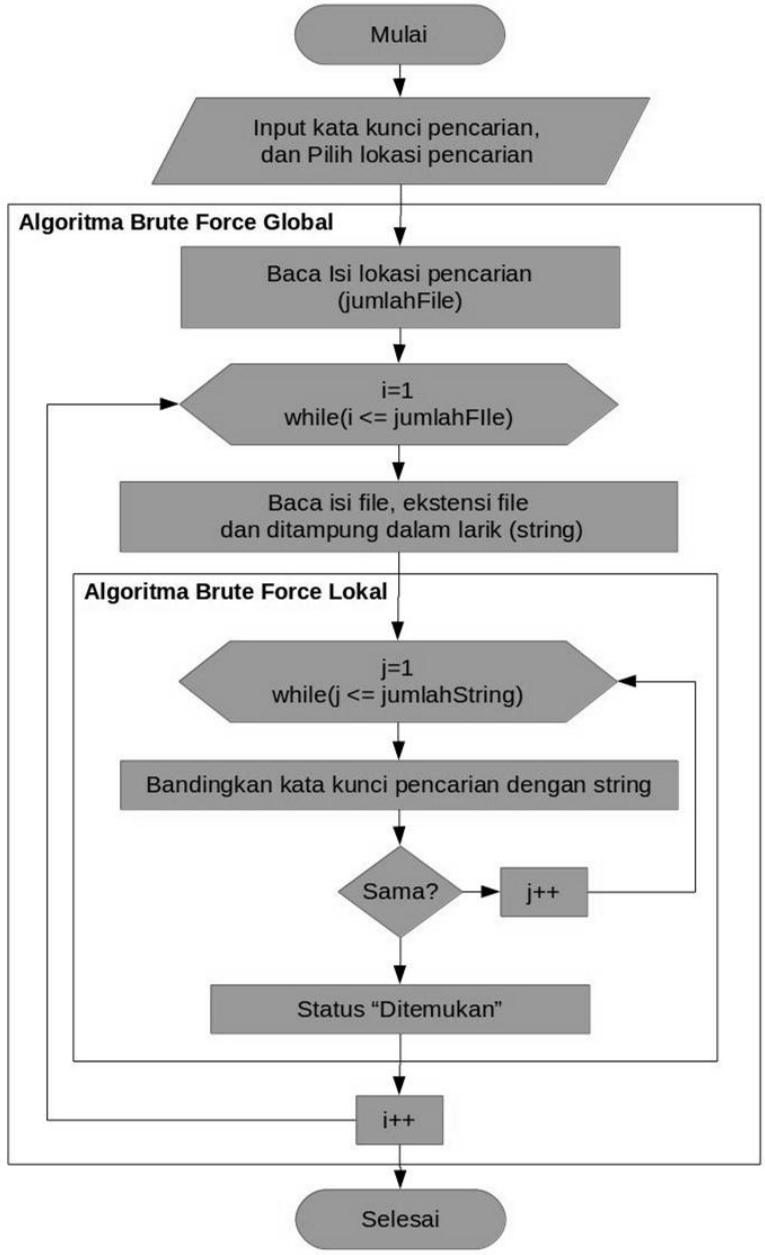
Penerapan algoritma *brute force* untuk proses pencarian *file* dengan pencocokan *string* dapat dilihat pada Gambar 3. Proses pencarian dimulai dengan menginputkan kata kunci yang ingin dicari dan penentuan lokasi yang akan dicari. Aplikasi akan menerima inputan kemudian melakukan proses pencarian lokasi dan membaca kata kunci yang akan dicari. Pada saat lokasi ditemukan, aplikasi akan membaca isi semua *file* yang ada di lokasi tersebut. Terdapat batasan disini bahwa *file* yang dibaca merupakan *file* teks berekstensi txt, doc, docx, xls, xlsx, ppt, pptx dan pdf. Hasil dari pembacaan tersebut akan diketahui jumlah *file* yang tersedia.

Algoritma *brute force* diterapkan pada 2 model pencarian yaitu pencarian global disebut dengan *algoritma brute force global* dan pencarian lokal disebut dengan *algoritma brute force* lokal.

Inti dari pencarian *file* terletak pada algoritma *brute force* lokal. Disini proses pencocokan *string* dilakukan antara *string* yang dicari dengan *string* yang di dalam *file*. Proses dimulai dengan membaca semua *string* didalam suatu *file* kemudian ditampung kedalam suatu variabel larik. Proses perbandingan dilakukan dengan membandingkan setiap karakter pada *string file* dengan *string* yang dicari. Apabila terdapat 1 karakter saja yang tidak sesuai akan dilakukan perpindahan dengan menggeser ke kanan sejauh n karakter sama seperti yang diilustrasikan pada Gambar 1. Hal ini dilakukan terus menerus sampai semua *string* pada *file* yang bersangkutan dibandingkan dengan *string* yang dicari. Apabila ditemukan *string* yang cocok, aplikasi akan memberikan informasi bahwa *file* yang bersangkutan terdapat *string* yang dicari dengan pesan ditemukan. Apabila tidak ditemukan, proses pencarian akan dilanjutkan ke *file* lainnya.

Pencarian pada *file* berikutnya diawali dengan pembacaan seluruh isi *file* yang dikonversi menjadi *string* dan dilanjutkan proses lainnya. Proses yang dilalui sama halnya pada *file* sebelumnya. Proses pencarian pada semua *file* yang ada dilokasi yang telah ditetapkan sebelumnya digolongkan sebagai algoritma *brute force* global. Hal ini didasarkan pada proses pembacaan semua *file* yang memiliki ekstensi sesuai dengan batasan yang ditetapkan dan proses pembacaan *file* dilakukan satu persatu sampai semua *file* diproses. *File* yang ditemukan berkemungkinan lebih dari satu karena proses pencarian dilakukan terhadap semua *file* yang ada.

Hasil pencarian akan menampilkan status dari masing-masing *file* yang terbaca. Apabila ditemukan kata kunci pada *file* yang bersangkutan akan ditampilkan status ‘ditemukan’ dan apabila tidak ditemukan akan ditampilkan status ‘tidak ditemukan’. Untuk kebutuhan perhitungan waktu ditempatkan script untuk membaca waktu diawal dan akhir dari eksekusi. Waktu proses ditampilkan ke antarmuka pada bagian bawah hasil dari pencarian.



Gambar 3. *Flowchart* pencarian *file*

3.2 Pengujian Fungsionalitas Aplikasi

Pengujian untuk mengetahui fungsionalitas dari setiap elemen yang ada di aplikasi. Ringkasannya dapat dilihat pada Tabel 1.

Tabel 1. Pengujian fungsionalitas aplikasi

No	Pengujian	Hasil Pengujian	Keterangan
1	Pengujian elemen inputan kata kunci (*textfield)*	Berhasil	Pengujian dilakukan dengan kondisi *textfield* kosong dan berisi
2	Pengujian elemen *div result* (hasil pencarian)	Berhasil	Pengujian dilakukan didahului dengan penekanan tombol cari dan diuji coba dengan perintah *echo*
3	Pengujian elemen button cari	Berhasil	Pengujian dilakukan dengan proses pemberian aksi

Dari pengujian diperoleh semua elemen yang ada di aplikasi dapat difungsikan sesuai dengan peruntukannya.

3.3 Pengujian Lingkungan Perangkat Lunak

Ditujukan untuk mengetahui apakah hasil rancangan kompatibel terhadap *web browser*. Pengujian dilakukan dengan menjalankan aplikasi di beberapa *web browser*. Ringkasannya dapat dilihat pada Tabel 2.

Tabel 2. Ringkasan pengujian eksperimen kedua

No	*Web browser*	Hasil	Keterangan
1	Mozilla firefox	Bisa dijalankan	Aplikasi dapat dijalankan dan tidak memerlukan *plugin* tambahan
2	IE	Bisa dijalankan	Aplikasi dapat dijalankan dan tidak memerlukan *plugin* tambahan
3	Chrome	Bisa dijalankan	Aplikasi dapat dijalankan dan tidak memerlukan *plugin* tambahan

Dari pengujian dapat diperoleh aplikasi mampu berjalan di 3 *web browser* yaitu Mozilla firefox, IE, dan chrome.

3.4 Pengujian Waktu Pencarian

Pengujian dilakukan untuk mengetahui performa algoritma dalam melakukan proses pencarian *file*. Ringkasannya dapat dilihat pada Tabel 3.

Tabel 3. Ringkasan pengujian waktu pencarian

No	Jumlah *File*	Rata-rata Waktu Proses (detik)
1	1	0.005222
2	5	0.017725
3	10	0.021865
4	15	0.063194
5	20	0.081426

Pengujian dilakukan terhadap *file* dengan format doc, docx, xls, xlsx, ppt, pptx, pdf, dan txt. Pengujian dengan melakukan percobaan 3 kali untuk setiap jenis format *file*. Hasil yang diperoleh menunjukan performa algoritma yang diterapkan dalam aplikasi rata-rata waktu proses 1 *file* sebesar 0.005222 detik sehingga apabila dirasiokan dari 120 kali percobaan untuk eksekusi 1 *file* membutuhkan waktu 0.003847 detik.

4. SIMPULAN

Pencarian berbasis konten pada *file* teks dapat dilakukan dengan menggunakan algoritma *brute force*. Pengembangan algoritma pencarian dengan menciptakan pencarian local dan global memberikan kesempatan semua *file* dapat dibaca dan diperiksa memenuhi kriteria pencarian. Semakin banyak *file* didalam suatu *folder* atau tempat pencarian semakin besar waktu yang dibutuhkan.

5. REFERENSI

[1] Saragih, M.A. 2013. Implementasi Algoritma *Brute Force* dalam Pencocokan Teks Font Italic untuk Kata Berbahasa Inggris pada Dokumen Microsoft Office Word. *Pelita Informatika Budi Darma*. Vol. 4(3): 84-87

[2] Mesran. 2014. Implementasi Algoritma *Brute Force* dalam Pencarian Data Katalog Buku Perpustakaan. *Informasi dan Teknologi Ilmiah (INTI)*. Vol. 3(1): 100-104

[3] Abdeen, R.A. 2011. An Algorithm for *String* Searching Based on Brute-Force Algorithm. *International Journal of Computer Science and Network Security (IJCSNS)*. Vol. 11(7): 24-27

[4] Haryanto, B. 2009. *Sistem Operasi.* Informatika : Bandung

[5] Breslauer, D. 1992. Efficient *String* Algorithmics. *PhD Thesis*. Computer Science Department. Columbia University

[6] Rasool, A., Tiwari, A., Khare, G.S.N. 2012. *String* Matching Methodologies: A Comparative Analysis. *International Journal of Computer Science and Information Technologies ((IJCSIT)*. Vol.3(2): 3394 - 3397

[7] Efendi,D., Hartono.T., Kurnaedi,A. 2013. Penerapan *String* Matching menggunakan Algoritma Boyer-Moore pada Translator Bahasa Pascal ke C. *Majalah Ilmiah UNIKOM*. Vol.11(2):262-275

[8] Munir,R. 2005. *Algoritma dan Pemrograman dalam Bahasa Pascal dan C*. Edisi 3. Informatika. Bandung

[9] Pressman, R.S. 2011. *Software Engineering a Practitioner's Approach*. 5th Edition. Mc Graw Hill. New York – USA