

LAPORAN TUGAS BESAR II
MATA KULIAH IF2211 STRATEGI ALGORITMA
Aplikasi DFS dan BFS pada *Web Crawler* di dalam
Mesin Pencari (*Search Engine*)



Oleh :

Kelompok “WarTeg AERO”

Daniar Heri Kurniawan / 13512064

Edmund Ophie / 13512095

Eric / 13512021

Sekolah Teknik Elektro dan Informatika

Intitut Teknologi Bandung

2014

BAB 1

Deskripsi Masalah

1.1 Rumusan Masalah

1. Bagaimana cara membuat *web crawler* sederhana (tanpa perangkian laman *web*) yang diterapkan untuk menjelajah laman-laman *web* hanya di dalam situs-situs tertentu saja (tidak ke semua laman *web* di dunia maya) ?
2. Bagaimana cara membuat sistem database yang tepat untuk menyimpan data hasil crawling?
3. Bagaimana cara mengaplikasikan *web crawler* tersebut di dalam sebuah mesin pencari sederhana yang hanya melakukan pencarian pada situs *web* tertentu?

1.2 Batasan Masalah

1. Situs *web* yang dijelajahi sebagai uji coba dalam pembuatan web crawler ini adalah situs-situs kecil dan sedang seperti:
 - Situs rinaldimunir (<http://informatika.stei.itb.ac.id/~rinaldi.munir>)
 - Situs www.itb.ac.id
 - Situs Wikipedia (www.wikipedia.org)
 - Situs lainnya yang tidak terlalu besar
2. Pohon pencarian yang dibentuk oleh algoritma pencarian BFS dan DFS memiliki batas kedalaman tertentu.
3. Spesifikasi program yang harus kami buat yaitu:
 - a. Program *search engine* yang anda buat terdiri dari tiga bagian: Program *query*, *web crawler*, dan *explorer*. Program *query* adalah berupa antarmuka pengguna-komputer, program *query* dibuat berbasis *web* (*web-base*). Tampilan antarmuka pengguna-komputer kira-kira seperti Gambar 5 di bawah ini:

My Crawler and Search Engine

Cari

[Crawler](#) [Algoritma traversal](#) [Perihal](#)

Gambar 5. Antarmuka pengguna-komputer

Keterangan :

- **Crawler**: pengguna mengaktifkan menu ini untuk memulai melakukan penjelajahan laman-laman web dari situs crawler awal yang ditentukan.
- **Algoritma**: pengguna dapat memilih algoritma traversal (DFS atau BFS). Default: BFS
- **Perihal**: keterangan tentang program dan pembuatnya

Anda dapat menambahkan menu lainnya, gambar, logo, dan sebagainya. Susunan menu di dalam antarmuka tidak harus seperti di atas, silakan dikreasikan sendiri.

- Hasil penjelajahan *Crawler* disimpan di dalam sebuah file *index* dan dapat ditampilkan ke layar jika pengguna ingin melihatnya. Pengguna dapat mengklik URL pada setiap *record* di dalam file dan *web browser* mengakses laman tersebut dan menampilkannya ke layar. File *index* boleh menggunakan file database (dbf).
- Keyword* pada setiap laman web yang disimpan oleh *crawler* dapat berupa judul pada laman tersebut, kata-kata di dalam tag, dan lain-lain (silakan dipikirkan). Jumlah keyword tidak dibatasi.
- Modul *explorer* melakukan pencocokan *query* pada file *index* menggunakan fungsi *string matching* yang sudah tersedia di dalam bahasa pemrograman.
- Seluruh laman *web* yang mengandung *query* yang cocok ditampilkan daftarnya di halaman yang baru. Misalnya jika *query* yang dimasukkan pengguna adalah “merapi”, maka luaran yang dihasilkan adalah seluruh laman yang judulnya mengandung kata “merapi” seperti contoh berikut:

[Gunung Merapi Meletus pada Jumat Dinihari](#)

[Jalan-jalan ke Gunung Merapi](#)

[Mbah Maridjan, Juru Kunci Merapi](#)

[Antara Keraton, Merapi, dan Laut Kidul](#)

Pada setiap judul yang ditampilkan terdapat *hyperlink* (pranala) sehingga dengan mengklik pranala tersebut maka laman web tersebut diakses oleh *web browser* dan ditampilkan ke layar.

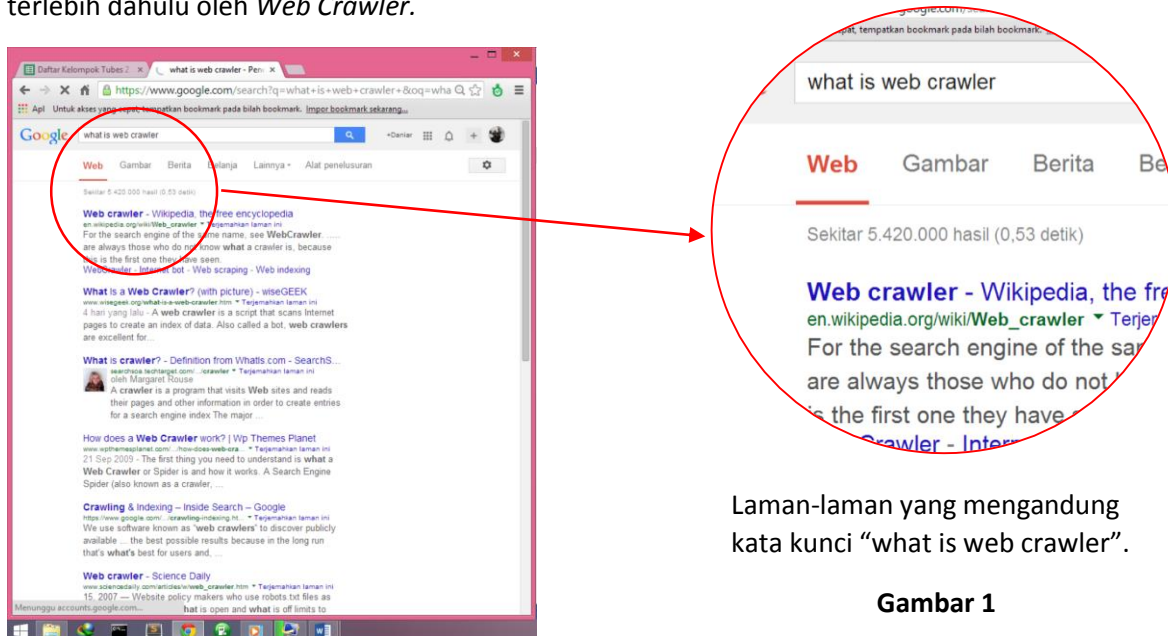
BAB 2

Dasar Teori

2.1 Mesin Pencari

Mesin pencari (*search engine*) adalah kakas yang sangat berguna untuk mencari informasi yang dibutuhkan di internet. Mesin pencari yang populer adalah *Google*, selain itu ada juga *Yahoo!*, *Altavista*, dan lain-lain meskipun tidak sepopuler *Google*.

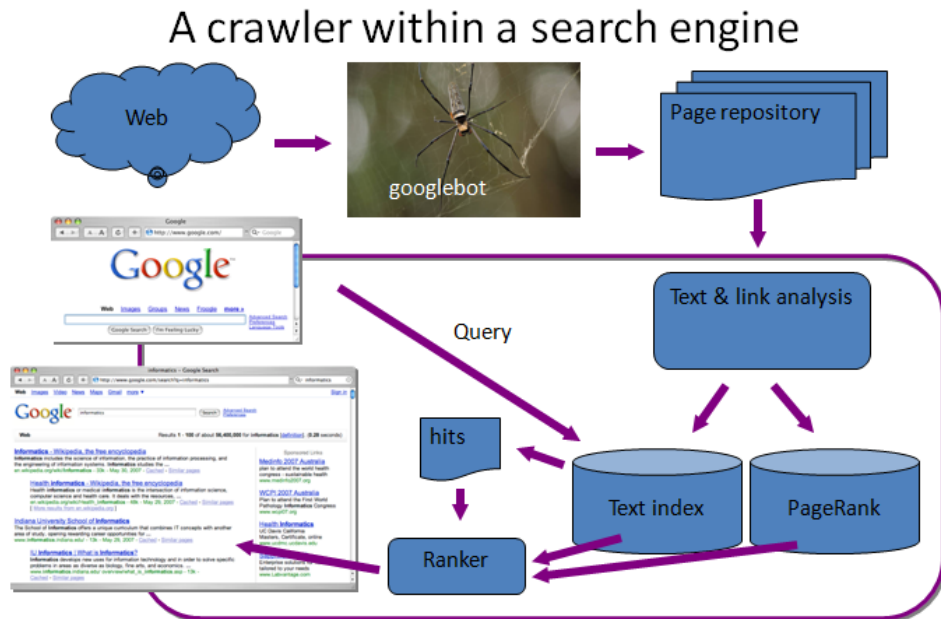
Tinjau mesin pencari yang bernama *Google*. Seringkali orang awam bertanya, mengapa *Google* begitu cepat menampilkan laman web (*web pages*) yang mengandung *query* yang dimasukkan pengguna (seperti gambar 1)? Jawabannya adalah karena laman-laman *web* tersebut sudah dijelajah terlebih dahulu oleh *Web Crawler*.



2.2 Web Crawler

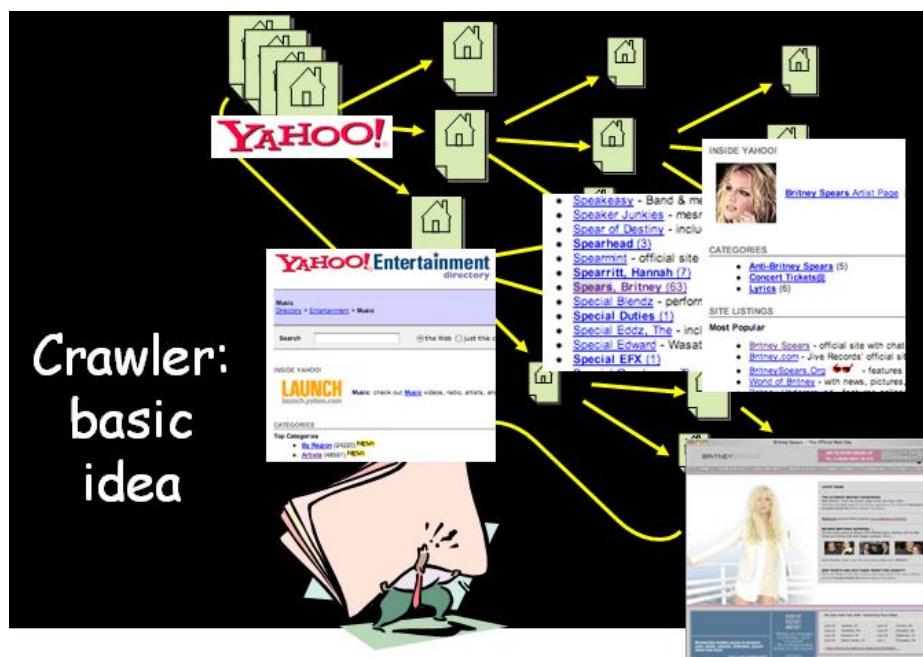
Web crawler adalah komponen mesin pencari yang terpenting. Tugasnya adalah menjelajah seluruh laman-laman *web* yang ada di dunia maya secara periodik (misalnya setiap 1 jam, setiap hari, dan sebagainya), karena laman-laman *web* selalu tumbuh dan berubah (bertambah, menyusut, atau sudah dimutakhirkan isinya). Nama lainnya adalah *Spider*, *Robot*, *Web Agent*, dan instans seperti *googlebot*, *msnbot*, dan lain-lain.

Setiap kali menjelajah internet, *web crawler* mencatat alamat laman tersebut (*URL*), beberapa kata penting (*keywords*) di dalam laman (nama laman *web*, judul di dalam laman, upa judul, dll), dan atribut lain seperti tanggal dan jam. Semua informasi tersebut disimpan di dalam sebuah file *index* (*text index*). Ketika pengguna (*user*) memasukkan *query* pada antarmuka mesin pencari, maka yang terjadi sesungguhnya adalah mesin melakukan pencarian *query* di dalam file *index* tersebut lalu menampilkan hasil pencarian sesuai dengan *ranking* laman. Oleh karena itu tidak heran jika pencarian berlangsung sangat cepat (Gambar 2).



Gambar 2. Web Crawler di dalam mesin pencari
(Sumber: Filippo Menczer @Indiana University School of Informatics in *Web Data Mining*)

Bagaimana *crawler* melakukan penjelajahan? Perhatikan bahwa setiap lama *web* umumnya memiliki pranala atau tautan (*link*) ke laman *web* lainnya. *Crawler* memulai penjelajahan dari sebuah laman *web* awal, lalu ia mengakses setiap pranala di dalam laman tersebut, mengunjungi laman *web* yang ditunjuk pranala tersebut, mencatat informasi di dalam laman tersebut, mengunjungi pranala berikutnya, dan seterusnya. Rangkaian pranala tersebut dapat dipandang sebagai sebuah struktur pohon, yang dalam hal ini simpul menyatakan laman *web* sedangkan busur menyatakan pranala ke laman tersebut (lihat Gambar 3).



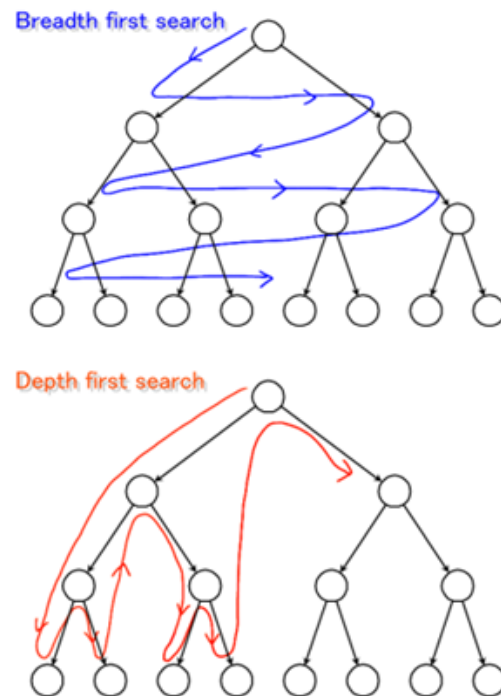
Gambar 3. Struktur laman-laman web membentuk pohon
(Sumber: Filippo Menczer @Indiana University School of Informatics in *Web Data Mining*)

2.3 Algoritma Pencarian Graf

Penjelahan laman-laman *web* tersebut dapat menggunakan salah satu dari dua algoritma pencarian di dalam graf, yaitu DFS dan BFS (lihat Gambar 4).

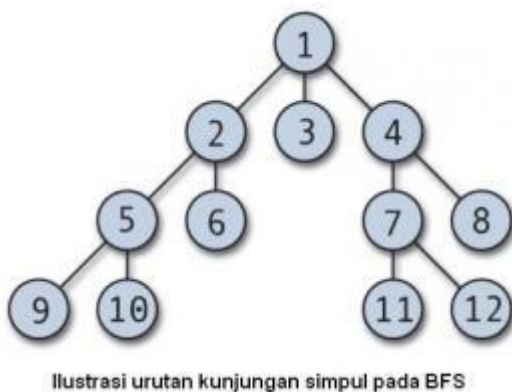
Graph traversal (BFS or DFS?)

- Breadth First Search
 - Implemented with QUEUE (FIFO)
 - Finds pages along shortest paths
 - If we start with “good” pages, this keeps us close; maybe other good stuff...
- Depth First Search
 - Implemented with STACK (LIFO)
 - Wander away (“lost in cyberspace”)



Gambar 4. Prinsip BFS dan DFS dalam penjelajahan laman laman web.

Algoritma yang pertama adalah algoritma BFS / Breadth First Search. Breadth-first search adalah algoritma yang melakukan pencarian secara melebar yang mengunjungi simpul secara preorder yaitu mengunjungi suatu simpul kemudian mengunjungi semua simpul yang bertetangga dengan simpul tersebut terlebih dahulu. Selanjutnya, simpul yang belum dikunjungi dan bertetangga dengan simpul-simpul yang tadi dikunjungi, demikian seterusnya. algoritma BFS menggunakan graf sebagai media representasi persoalan, tidak sulit untuk mengaplikasikan algoritma ini dalam persoalan-persoalan teori graf.



Ilustrasi urutan kunjungan simpul pada BFS

Gambar 5

Source: http://en.wikipedia.org/wiki/Breadth-first_search#How_it_works

Cara Kerja Algoritma BFS

Dalam algoritma BFS, simpul anak yang telah dikunjungi disimpan dalam suatu antrian. Antrian ini digunakan untuk mengacu simpul-simpul yang bertetangga dengannya yang akan dikunjungi kemudian sesuai urutan pengantrian. Untuk memperjelas cara kerja algoritma BFS beserta antrian yang digunakannya, berikut langkah-langkah algoritma BFS:

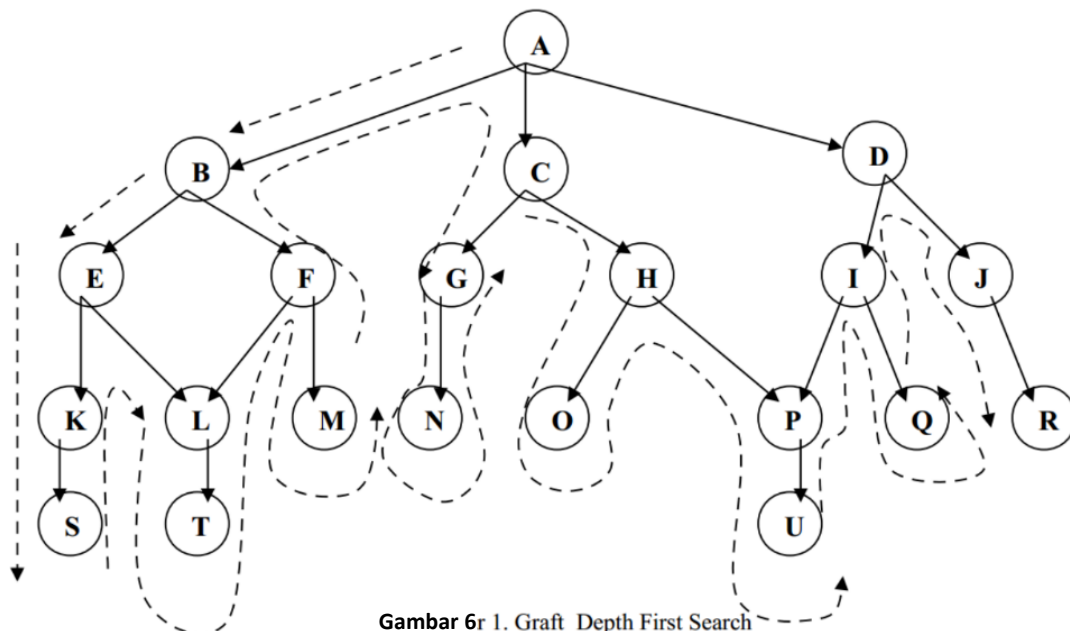
1. Masukkan simpul ujung (akar) ke dalam antrian
2. Ambil simpul dari awal antrian, lalu cek apakah simpul merupakan solusi

3. Jika simpul merupakan solusi, pencarian selesai dan hasil dikembalikan.
4. Jika simpul bukan solusi, masukkan seluruh simpul yang bertetangga dengan simpul tersebut (simpul anak) ke dalam antrian
5. Jika antrian kosong dan setiap simpul sudah dicek, pencarian selesai dan mengembalikan hasil solusi tidak ditemukan
6. Ulangi pencarian dari langkah kedua

Algoritma yang kedua adalah Depth First Search , DFS adalah algoritma pencarian simpul dalam graf secara travelsal yang dimulai dari simpul akar dan mengecek simpul anaknya yang pertama, setelah itu, algoritma mengecek simpul anak dari simpul anak yang pertama tersebut, hingga mencapai simpul daun atau simpul tujuan. Jika solusi belum ditemukan, algoritma melakukan runut balik (backtracking) ke simpul orangtuanya yang paling baru diperiksa lalu mengecek simpul anaknya yang belum diperiksa, sedemikian seterusnya hingga simpul solusi ditemukan. Depth First Search jauh lebih efisien untuk ruang pencarian dengan banyak percabangan, karena tidak perlu mengevaluasi semua simpul pada tingkat tertentu pada saat dimulainya pencarian. (Wikipedia,(2007)), Depth-First Search. Depth First Search mempunyai keuntungan dan kelemahan :

- a. Keuntungan dari Depth First Search
 1. Membutuhkan memori yang relatif kecil, karena hanya node-node pada lintasan yang aktif yang di simpan.
 2. Secara kebetulan, metode Depth First Search akan menemukan solusi tanpa harus menguji lebih banyak lagi dalam ruang keadaan.
- b. Kelemahan dari Depth First Search.

Memungkinkan tidak ditemukannya tujuan yang diharapkan.



Gambar 6r 1. Graft Depth First Search
 Sumber: Elearning.uin-suka.ac.id/attachment/pencarian_heuristik_bjyr6.pdf

BAB 3

Analisis Pemecahan Masalah

1. Membuat *web crawler* sederhana (tanpa perangkian laman *web*)

Web crawler yang kami buat menggunakan bahasa C# sebagai bahasa utamanya dan html, php, serta sql untuk membuat beberapa fitur nya. Program utama yang melakukan crawling terdiri dari tiga kelas. Kelas yang pertama (Program.cs) adalah kelas yang berisi program main, algoritma BFS, dan algoritma DFS. Kelas yang kedua (General.cs), kelas ini berisi method-method yang digunakan untuk mengolah halaman html sehingga bisa dimanfaatkan untuk proses penyimpanan di sistem basisdata. Kemudian kelas yang ketiga (SQLFunction) adalah kelas yang khusus menangani proses koneksi antara sistem basisdata MySQL dengan program C# kami yang mengolah datanya. Selain itu kelas SQLFunction ini juga mengolah query-query dari user.

Dalam mengaplikasikan algoritma BFS dan DFS, kami memanfaatkan berbagai template class yang sudah disediakan oleh C#, misalnya List<>, Queue<>, Stack<>, dan SortedSet<>. Dalam algoritma BFS, kami menggunakan struktur data antrian atau queue untuk menyimpan URL – URL yang harus dikunjungi. Sedangkan pada algoritma DFS, kami menggunakan algoritma DFS yang berbasis stack. Pada algoritma DFS ini, kami tidak hanya mengekskan satu simpul disetiap level, tetapi kami mengekskan semua anak yang ada pada simpul target kemudian menyimpannya kedalam stack. Model algoritma DFS seperti itu sudah kami uraikan pada bagian Dasar Teori.

Algoritma BFS dan DFS masing –masing punya kelebihan dan kelemahan. Dalam kasus web crawler ini, keduanya sangat berpotensi untuk melakukan crawling tanpa heenti dan menyebabkan ia lost in space. Oleh karena itu kami membatasi kedalaman crawling yang bisa dilakukan pada algoritma DFS dan BFS tersebut. Batas kedalaman tersebut akan melakukan increment secara otomatis setiap kali pohon status yang dibentuk bertambah level kedalamannya. Sehingga dengan adanya batas kedalaman ini, proses crawling pada suatu simpul akan dihentikan jika memang batas kedalamannya sudah tercapai, kemudian prosesnya akan dilanjutkan dengan proses backtracking ke URL-URL lain yang sudah ada di dalam stack atau queue hingga stack atau queue nya kosong.

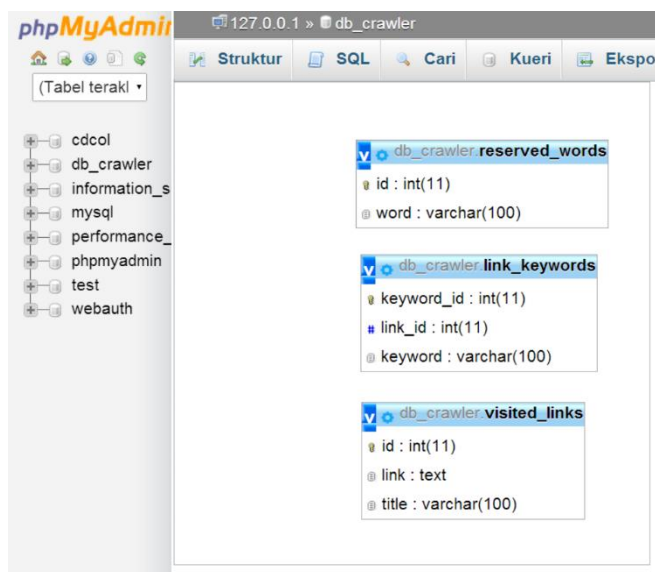
Pada algoritma DFS, mengambil elemen pada stack kami memberi istilah pop sedangkan pada BFS dilakukan dengan depart. Setiap kali terjadi proses pop atau depart, kami mengolah halaman html yang didownload dari URL tersebut kemudian melakukan update keyword. Keyword disini maksudnya adalah daftar kata-kata yang terdapat dalam laman yang akan kita simpan pada sistem basisdata. Setelah mendapatkan keywordsnya, kami melakukan update juga ke sistem basisdata untuk menambahkan URL yang barusan telah dikunjungi agar nantinya tidak dikunjungi lagi. Pada sistem basisdata, kami melakukan pengindekan pada masing-masing tabelnya untuk mempermudah pemrosesannya dan untuk memberikan primary key pada setiap tabelnya.

Sistem basisdata yang kami gunakan adalah MySQL. Kami memiliki 3 tabel yang secara garis besar menyimpan keywords, link, judul link, dan reserved words. Keywords yang kita simpan memiliki sebuah indeks unik yang akan bersesuaian dengan indeks pada link. Sehingga ketika terjadi query untuk menampilkan halaman web berdasarkan keyword dari user, kami akan menampilkan link-linknya sesuai dengan link yang indeksnya bersesuaian dengan indeks pada keywords. Keywords yang kami simpan hanya kata-kata tertentu yang memang mengandung makna, oleh sebab itu kami mempunyai daftar pengecualian kata yang tersimpan dalam reserved words. Setelah semua keywords terkumpul dan link-linknya telah kami kunjungi menggunakan

algoritma DFS dan BFS, user bisa melakukan query agar dia tahu link-link atau URL-URL mana saja yang mengandung keyword yang dia ketikkan. Link yang kami tampilkan juga disertai dengan judul atau title yang ada pada laman tersebut layaknya search engine pada umumnya.

2. Membuat sistem database yang tepat untuk menyimpan data hasil crawling

Dalam membuat sistem basisdata pada web crawler ini, kami memutuskan untuk menggunakan MySQL karena lebih mudah mengolahnya berdasarkan query-query yang akan diberikan user. Selain itu, MySQL juga mengatasi keterbatasan memori penyimpanan pada variabel. Dengan melakukan update pada MySQL, kami tidak perlu mengkhawatirkan terjadinya segmentation fault yang disebabkan kesalahan akses memory pada variabel C# karena variabel tersebut menyimpan data yang sangat besar. Sistem basisdata pada MySQL memungkinkan web crawler yang kami buat untuk menyimpan keywords hingga puluhan ribu kata. Data yang sebesar itu tentu tidak akan efisien jika kami hanya menyimpannya pada variabel ataupun pada teks kosong.



Gambar 7. Sistem Basisdata

MySQL merupakan Data Base Management System yang telah dibekali dengan bermacam-macam kemampuan dalam mengolah query. Oleh karena itu web crawler yang kami buat juga terintegrasi dengan bahasa SQL untuk mengolah query-querynya. Untuk menyimpan data- data penting pada web crawler ini, kami membuat 3 tabel relasi seperti terlihat pada Gambar 7 disamping. Kelemahan dari penggunaan MySQL ini adalah kami harus menjadi server dengan menginstall berbagai aplikasi tambahan sebelum dapat menjalankan program web crawler.

3. Mengaplikasikan *web crawler* tersebut di dalam sebuah mesin pencari sederhana

Web crawler yang kami buat tidak hanya menerima query dari Command Line Interpreter, tetapi juga harus bisa memberikan tampilan layaknya search engine pada umumnya, yaitu berbasis html. Dengan melakukan integrasi antara executable program dengan php script pada web html, kami melakukan trigger untuk menjalankan program C# melalui tampilan web html. Mesin pencari yang melakukan proses crawling berdasarkan algoritma BFS dan DFS ini memiliki batas kedalaman crawling yang bisa diatur oleh user dari User Interface berbasis web tersebut. Selain itu user juga bisa memasukkan website apa saja yang ingin dijalankan sebagai web awal pada web crawler.

BAB 4

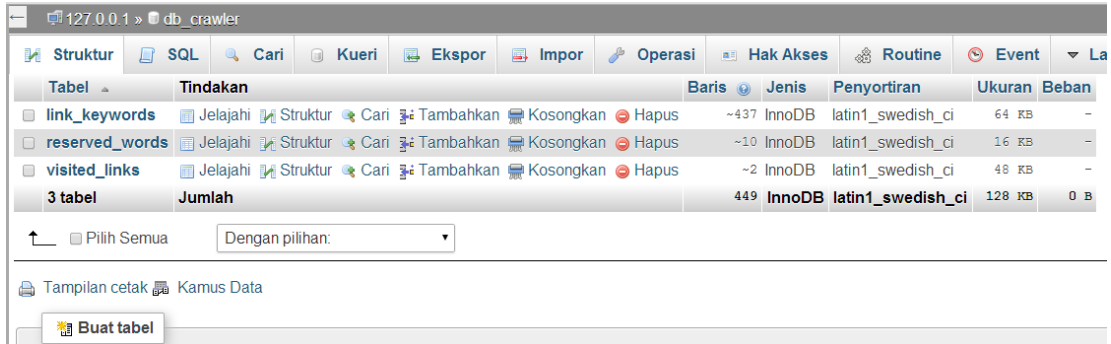
Implementasi dan Pengujian

1. Implementasi

Secara umum web crawler ini dibagi menjadi tiga bagian besar, yaitu : crawler, search-engine (GUI), dan database.

a. Crawler

Pada web crawler kami menggunakan 2 buah algoritma, yaitu BFS dan DFS. Program crawler ini dibuat dalam bahasa C# dan program jadinya berbentuk sebuah executeable bernama crawler.exe. Crawler ini akan melakukan penjelajahan sesuai algoritma yang dipilih oleh user. Crawler akan mendapatkan semua link-link serta keywords yang ada pada website dan menyimpannya kedalam sebuah database. Untuk menjalankan crawler ini dapat dilakukan melalui GUI yang telah dibuat. Gambar 8 menunjukkan hasil crawling yang telah terupdate pada file basisdata MySQL.



Tabel	Tindakan	Baris	Jenis	Penyortiran	Ukuran	Beban
link_keywords	Jelajahi Struktur Cari Tambahkan Kosongkan Hapus	~437	InnoDB	latin1_swedish_ci	64 KB	-
reserved_words	Jelajahi Struktur Cari Tambahkan Kosongkan Hapus	~10	InnoDB	latin1_swedish_ci	16 KB	-
visited_links	Jelajahi Struktur Cari Tambahkan Kosongkan Hapus	~2	InnoDB	latin1_swedish_ci	48 KB	-
3 tabel	Jumlah	449	InnoDB	latin1_swedish_ci	128 KB	0 B

Gambar 8. Hasil crawler yang menghasilkan data sebesar 128 kb

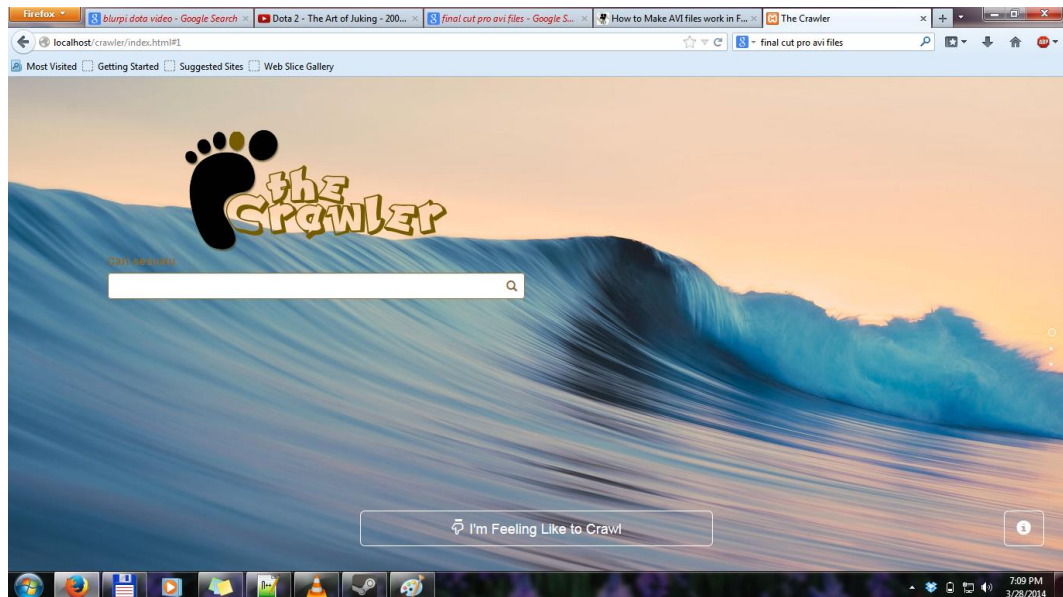
b. Search-engine (GUI)

Untuk memvisualisasikan dan memproses search-engine ini kami menggunakan kombinasi dari pemrograman multi-bahasa, yaitu: HTML dan Javascript untuk GUI nya, serta PHP untuk pemrosesan “query”-nya. Kami juga membuat program dalam bahasa C# yang bernama QueryMaker.exe. Program ini yang sebenarnya terkoneksi langsung ke database. Program ini memproses query dari user di dalam database, dan melakukan “*page ranking*” sederhana. Program ini dipanggil secara diam-diam ketika user melakukan pencarian.

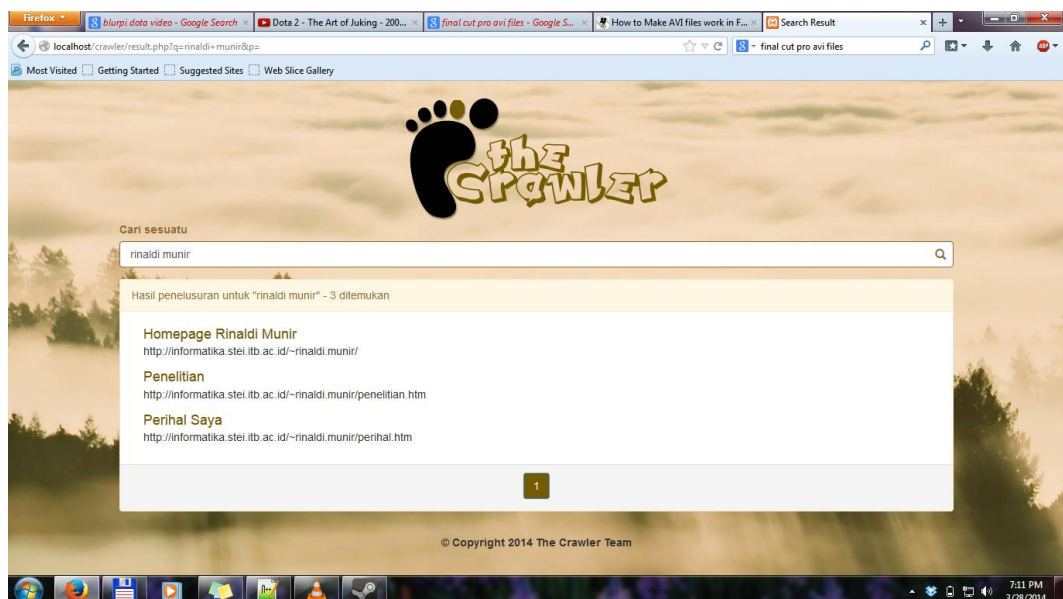
Pada tampilan awal, terdapat halaman/menu, yaitu: Search-Engine menu, Crawler menu, dan About menu. Pada search-engine menu, user dapat memasukkan “keywords” yang ingin dicari pada “textbox” yang disediakan. PHP akan memproses “query” dari user dan mencari link yang berkorelasi dengan “keywords” pada database. Data yang didapat kemudian ditampilkan sebagai kombinasi judul dan link-nya. Pada crawler-menu, terdapat konfigurasi yang perlu diisi sebelumnya, yaitu :

- URL-URL yang akan dicrawl.
- Algoritma crawl (BFS/DFS).
- Kedalaman-nya.

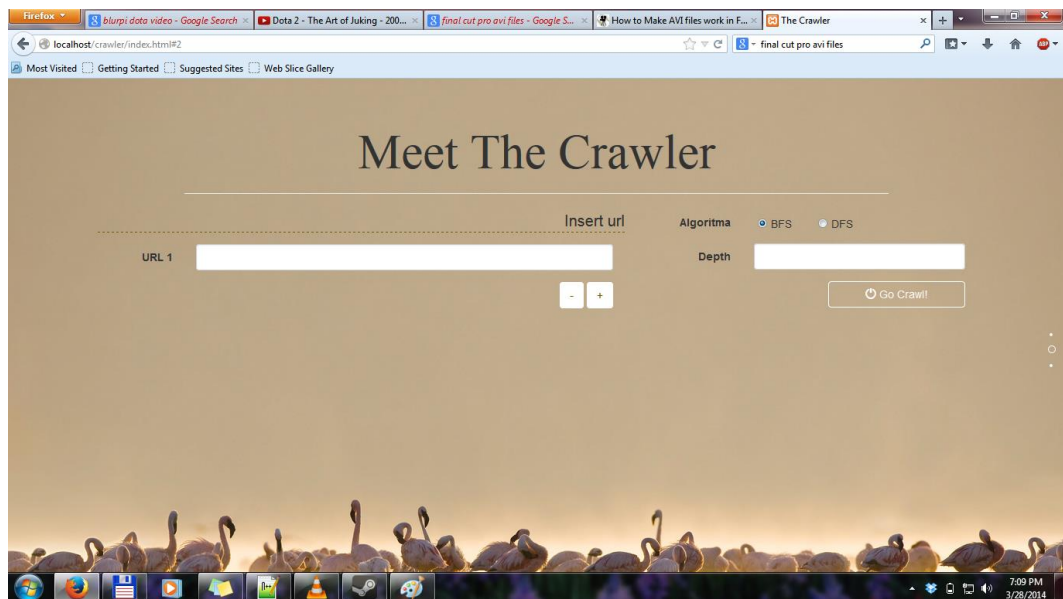
Seperti tampilan GUI pada Gambar 9, 10, 11, 12, ada beberapa tombol yang bisa dimanfaatkan oleh user untuk mempermudah penggunaan aplikasi web crawler ini. Ketika tombol “Go-Crawl” ditekan, maka PHP akan menjalankan program “Crawler.exe”. Pada about menu, terdapat deskripsi singkat tentang program dan tentang tim kami. GUI yang telah kami buat memiliki 4 buah sesi (page) yang berbeda. Page pertama berupa Halaman awal ketika aplikasi web crawler baru dibuka. Yang kedua adalah tampilan ketika hasil pemrosesan keywords dari user ditampilkan, seperti pada gambar 10. Kemudian gambar 11 adalah tampilan yang bisa digunakan untuk mengatur spesifikasi proses crawling sesuai yang kehendak user. Yang terakhir adalah tampilan ABOUT yang berisi data kami dan penjelasan sedikit tentang program.



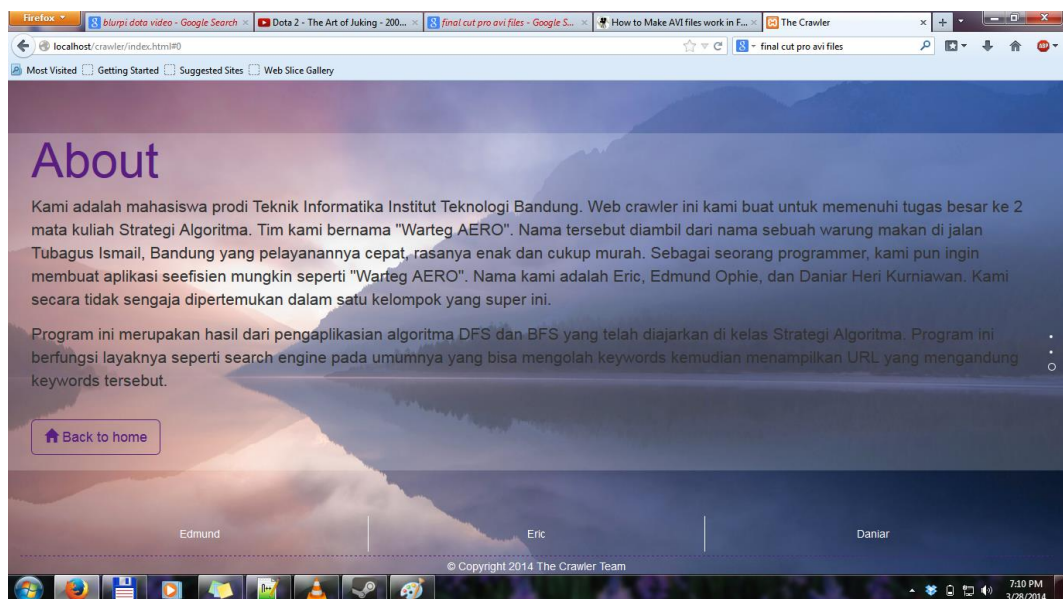
Gambar 9. Tampilan awal web crawler yang telah kita implementasikan pada halaman HTML



Gambar 10. Hasil pemrosesan keywords dari user berdasarkan database yang telah dibentuk dari proses crawling



Gambar 11. Tampilan untuk mengatur spesifikasi crawling seperti yang dikehendaki user. User bisa mengatur URL awal dan kedalaman algoritma DFS dan BFS.

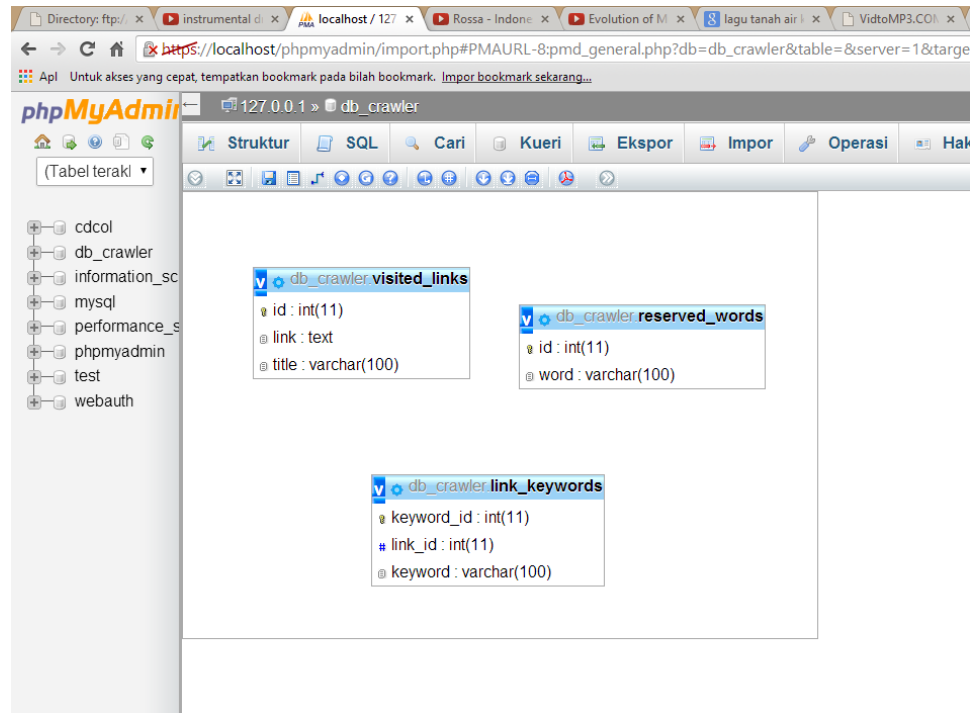


Gambar 12. Tampilan yang berisi deskripsi pembuat dan deskripsi ringkas program.

c. Database

Untuk database, kami menggunakan MySQL sebagai DBMS nya. Penggunaan database ini cukup mudah dan sudah cukup ramah untuk dikombinasikan dengan bahasa C#. Pada database ini, kami membuat 3 buah tabel relasi, yaitu: tabel visited_links, link_keywords, reserved_words. Tabel visited_links berisi link yang sudah pernah di-crawl beserta judul-nya. Tabel link_keywords berisi daftar keyword dari link-link yang sudah di-crawl. Tabel reserved_words berisi daftar keywords yang menjadi pengecualian. Jika kemudian keywords

ini ditemukan saat proses crawl maka akan diabaikan dan tidak akan dimasukkan ke database.



Gambar 13. Database yang menggunakan MySQL

2. Pengujian

Kami melakukan pengujian terhadap program melalui testing program secara bertahap. Kami membuat crawler terlebih dahulu dan melakukan pengujian crawler dengan cara menjalankan program tersebut. Kami kemudian membuat search engine dan melakukan pengujian pada search engine. Setelah itu kami membuat user interface melalui pemrograman web dan mengintegrasikan program yang sudah ada dengan php. User interface juga dicoba dengan cara mencoba semua interaksi yang mungkin dilakukan.

Catatan: Hasil pengujian bisa langsung dilihat melalui video yang telah kami unggah di youtube atau bisa melihat video yang terlampir pada CD.

BAB 5

Kesimpulan dan Saran

Pada tugas besar ini kami membuat sebuah web crawler sederhana menggunakan bahasa C# dan berintegrasi juga dengan beberapa bahasa lain untuk melengkapi fitur-fiturnya. Web crawler ini bekerja berdasarkan algoritma BFS dan DFS yang diimplementasikan menggunakan template class yang sudah disediakan bahasa C# yaitu Queue<> dan Stack<>. Pada algoritma DFS dan BFS ini, kami memberikan batas kedalaman maksimal proses penelusuran agar proses crawlingnya lebih terkendali. Proses crawling akan menyimpan keywords atau kata kunci –kata kunci yang nantinya bisa digunakan untuk membantu user menemukan website yang tepat seperti kata kunci yang dia mau. Untuk memperoleh URL mana saja yang cocok dengan kata kunci dari user, kami melakukan proses query pada sistem basisdata yang telah menyimpan keywords untuk menampilkan URL yang bersesuaian.

Untuk menyimpan daftar keyword, link, dan data lainnya, kami memanfaatkan sistem basisdata yang disediakan oleh MySQL. MySQL menyediakan berbagai fitur-fitur yang dapat mempermudah kami dalam memproses query, sehingga web crawler kami menjadi lebih efektif pemrosesannya daripada harus menyimpan data pada file txt atau pun pada variabel. Untuk memudahkan penggunaan user, kami memberikan User Interface yang berbasis web layaknya search engine pada umumnya. Dengan memakai web crawler yang telah kami buat, user bisa mencari laman-laman pada web berdasarkan keywords yang dia berikan. Selain itu, user juga bisa menyeting laman atau URL awal sebagai seed untuk proses crawling menggunakan algoritma BFS dan DFS yang telah diimplementasikan pada bahasa C#.

Kami yakin program yang kami buat masih bisa di kembangkan lebih jauh lagi dan masih memungkinkan untuk mengalami berbagai perbaikan. Namun sejauh ini kami cukup puas dengan program ini dan kami memiliki beberapa saran bagi developer-developer yang juga mengembangkan program sejenis, diantaranya yaitu : Sebaiknya proses crawling yang dilakukan dengan algoritma BFS dan DFS memiliki batas kedalaman agar prosesnya terkendali. Untuk manage data hasil crawling, sebaiknya menggunakan perangkat lunak lain yang khusus menangani masalah basis data, misalnya MySQL. Dalam menggunakan tipe data pada C#, sebaiknya memanfaatkan secara maksimal template – template yang sudah disediakan oleh C#, karena pepatah mengatakan “don’t reinvent the wheel!”.

Tidak lupa kami mengucapkan syukur kepada Tuhan dan semua pihak yang telah membantu kelompok kami dalam menyelesaikan tugas ini. Tentu saja web crawler yang kami buat ini belum lah sempurna, sehingga kami terbuka dengan segala kritik dan saran yang membangun. Demikian yang bisa kami jelaskan dalam laporan ini, jika ada kata-kata yang kurang berkenan, kami minta maaf.