IMPLEMENTASI DIRECTED ACYCLIC WORD GRAPH DENGAN MENGGUNAKAN ALGORITMA BLOW THE BRIDGE PADA WEB CRAWLER UNTUK INDEXING WEB

Santosa Raharjanto[1], Budi Susanto[2], R. Gunawan Santosa[3]

Abstrak:

Dengan kemungkinan begitu banyaknya kata yang kembar atau sama dalam sebuah halaman web, pemeriksaan setiap kata yang kembar dengan memanfaatkan pemeriksaan dalam *database* secara teori akan membuat kinerja *web crawling* menjadi kurang efektif. Oleh karenanya, kata-kata yang terdapat dalam sebuah halaman web perlu untuk diperiksa dan dipilah dalam memori utama dengan memanfaatkan *Directed Acyclic Word Graph* sebelum masuk pada database sebagai daftar kata. Analisis dilakukan pada data-data *indexing* web dan kecepatan *searching* untuk melihat potensi *Directed Acyclic Word Graph* pada 500 dokumen web yang ada di internet. Kesimpulan yang diperoleh antara lain, penggunaan *Directed Acyclic Word Graph* pada 500 dokumen yang diujicobakan dapat menghemat jumlah kata hingga hampir 96% dari jumlah data semula, sedangkan pencarian kata pada *Directed Acyclic Word Graph* dipengaruhi oleh faktor-faktor seperti kecepatan perangkat keras, jumlah URL yang ditemukan, panjang kata yang dicari, dan sering atau tidaknya kata tersebut muncul pada satu kedalaman tertentu.

Kata Kunci: *web crawling, Directed Acyclic Word Graph, indexing web.*

1. Pendahuluan

Search Engine atau mesin pencari merupakan sebuah fasilitas yang berguna karena memudahkan pengguna internet untuk mencari informasi yang dibutuhkan. *Search Engine* memudahkan pencarian suatu situs karena pencarian dilakukan berdasarkan kata kunci yang diinputkan, kemudian sebuah *Search Engine* akan menampilkan semua URL (*Uniform Resource Locator*) dari situs-situs yang berhubungan dengan kata kunci tersebut. Dengan kemungkinan begitu banyaknya kata yang kembar atau sama dalam sebuah halaman web, pemeriksaan setiap kata yang kembar dengan memanfaatkan pemeriksaan dalam *database* secara teori akan membuat kinerja *web crawling* menjadi kurang efektif. Oleh karenanya, kata-kata yang terdapat dalam sebuah halaman web perlu untuk diperiksa dan dipilah dalam memori utama sebelum masuk pada database sebagai daftar kata.

Metode yang akan diimplementasikan adalah metode dengan algoritma *Directed Acyclic Word Graph* (DAWG) yang akan diimplementasikan pada *web crawler*. Metode ini memerlukan sebuah URL utama (*portal*) sebagai inputan, kemudian *meta tags* beserta URL akan disimpan ke dalam sebuah *search engine data file*. Setiap *entry point* yang direpresentasikan dalam graf melambangkan huruf pertama di dalam proses pencarian. Tiap node merepresentasikan sebuah huruf, dan satu node dapat terhubung oleh lebih dari satu node berikutnya, tergantung apakah huruf tersebut merupakan huruf yang dicari. Struktur data DAWG mirip dengan struktur data *trie*, namun jauh lebih

[1] *Santosa Raharjanto, Mahasiswa Teknik Informatika, Fakultas Teknik, Universitas Kristen Duta Wacana*
[2] *Budi Susanto, S.Kom., M.T., Dosen Teknik Informatika, Fakultas Teknik, Universitas Kristen Duta Wacana*
[3] *Drs. R. Gunawan Santosa, M. Si., Dosen Teknik Informatika, Fakultas Teknik, Universitas Kristen Duta Wacana*

1. Pendahuluan

Search Engine atau mesin pencari merupakan sebuah fasilitas yang berguna karena memudahkan pengguna internet untuk mencari informasi yang dibutuhkan. *Search Engine* memudahkan pencarian suatu situs karena pencarian dilakukan berdasarkan kata kunci yang diinputkan, kemudian sebuah *Search Engine* akan menampilkan semua URL (*Uniform Resource Locator*) dari situs-situs yang berhubungan dengan kata kunci tersebut. Dengan kemungkinan begitu banyaknya kata yang kembar atau sama dalam sebuah halaman web, pemeriksaan setiap kata yang kembar dengan memanfaatkan pemeriksaan dalam *database* secara teori akan membuat kinerja *web crawling* menjadi kurang efektif. Oleh karenanya, kata-kata yang terdapat dalam sebuah halaman web perlu untuk diperiksa dan dipilah dalam memori utama sebelum masuk pada database sebagai daftar kata.

Metode yang akan diimplementasikan adalah metode dengan algoritma *Directed Acyclic Word Graph* (DAWG) yang akan diimplementasikan pada *web crawler*. Metode ini memerlukan sebuah URL utama (*portal*) sebagai inputan, kemudian *meta tags* beserta URL akan disimpan ke dalam sebuah *search engine data file*. Setiap *entry point* yang direpresentasikan dalam graf melambangkan huruf pertama di dalam proses pencarian. Tiap node merepresentasikan sebuah huruf, dan satu node dapat terhubung oleh lebih dari satu node berikutnya, tergantung apakah huruf tersebut merupakan huruf yang dicari. Struktur data DAWG mirip dengan struktur data *trie*, namun jauh lebih effisien untuk ukuran datanya. DAWG menggabungkan huruf-huruf yang sama pada node yang sama jika ada huruf yang identik. Disebut identik bila mempunyai huruf dan kedalaman yang sama, node tersebut juga mempunyai tanda stop yang sama, serta mempunyai anak yang juga identik. Hal ini membuat ukuran DAWG menjadi lebih kecil.

Untuk mengetahui seberapa efektif *web crawler* yang memanfaatkan metode DAWG ini dalam mengambil dan memilah kata dengan kondisi software, hardware dan koneksi internet yang telah ditentukan, diperlukan setidaknya 500 dokumen web yang diambil langsung dari internet. Analisis dilakukan pada data-data *indexing* web dan kecepatan *searching* untuk melihat potensi *Directed Acyclic Word Graph* pada 500 dokumen web yang ada di internet.

2. Cara Kerja

Terdapat 3 hal utama yang menjadi pilar utama dari program yang hendak dibuat. Penjelasan ketiga pilar utama itu adalah sebagai berikut :

1) Input

Pada tahap ini, *user* diminta untuk memasukkan URL portal yang akan dicari, misalkan dalam kasus ini digunakan URL portal : http://www.ukdw.ac.id . Program akan melakukan pengecekan validitas URL yang diinputkan kemudian mulai melakukan proses *spidering*.

2) Proses

Proses yang dilakukan melibatkan beberapa proses penting, antara lain :

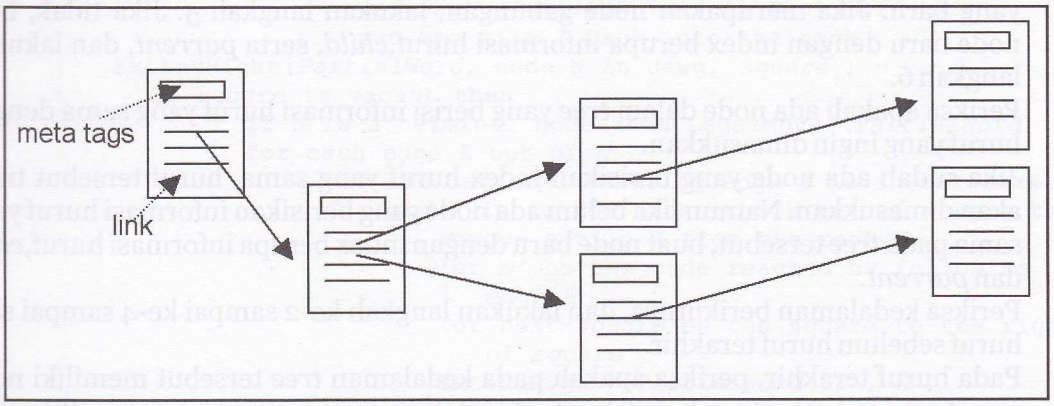
a) koneksi ke halaman web tersebut
b) pembuatan indeks URL
c) pengekstrakan *link* file
d) pembacaan *header* file
e) pembacaan *body* file
f) pengubahan setiap kata menjadi DAWG tree.
g) Peyimpanan data ke dalam media penyimpanan.

---

[1] *Building Web Spiders: Web-Based Information Architectures, Jaime Carbonell, Carnegie Mellon University, http://www.andrew.cmu.edu/course/20-760/notes/lecture5.ppt, 27 Agustus 2007*

```
PROCEDURE SPIDER4(G, {SEEDS})
|       Initialize COLLECTION <big file of URL-page pairs>
|       Initialize VISITED <big hash-table>
|
|       For every ROOT in SEEDS
|               Initialize STACK <stack data structure>
|               Let STACK := push(ROOT, STACK)
|
|               While STACK is not empty,
|                       Do URLcurr := pop(STACK)
|                               Until URLcurr is not in VISITED
|                       insert-hash(URLcurr, VISITED)
|                       PAGE := look-up(URLcurr)
|                       STORE(<URLcurr, PAGE>, COLLECTION)
|                       For every URLi in PAGE,
|                               push(URLi, STACK)
Return COLLECTION
```

Berikut ini merupakan bagan umum proses kerja *crawlers* :



Gambar 1. Proses kerja *crawlers*

2.1. Directed Acyclic Word Graph

Directed Acyclic Word Graph (DAWG) adalah sebuah struktur data dengan pencarian kata yang sangat cepat. Setiap *entry point* yang direpresentasikan dalam graf melambangkan huruf pertama di dalam proses pencarian. Tiap node merepresentasikan sebuah huruf, dan satu node dapat terhubung oleh lebih dari satu node berikutnya, tergantung apakah huruf tersebut merupakan huruf yang dicari. Struktur data DAWG mirip dengan struktur data trie, namun jauh lebih effisien untuk ukuran datanya. Disebut *Directed graph* karena graf ini hanya dapat bergerak ke arah yang telah ditentukan diantara dua node. Dengan kata lain, kita hanya dapat bergerak dari node A ke node B apabila ada arah yang telah ditentukan sebelumnya. Disebut *Acyclic* karena node-node tersebut tidak berputar kembali. Dengan kata lain, node terakhir tidak menunjuk kembali ke node pertama. Kita tidak dapat membuat jalur dari A ke B ke C kemudian kembali lagi ke A. Hal ini akan membentuk putaran yang dapat mengkibatkan *looping* tanpa batas di dalam pencariannya.

Algoritma *tree*. Akan tetapi, algoritma *Directed Acyclic Word Graph* tersebut yang banyak dipakai untuk membentuk DAWG mempunyai beberapa kelemahan yaitu penambahan kata yang baru pada *tree* yang sudah diubah menjadi bentuk DAWG amat

sulit, bahkan nyaris mustahil, untuk dilakukan . Hal ini dikarenakan DAWG *tree* harus diubah dulu menjadi bentuk *trie tree* kembali, sebelum dapat dilakukan penambahan kata yang baru. Hal ini tentu akan sangat menyulitkan pembaharuan data yang mungkin akan dilakukan di masa mendatang. Padahal jumlah halaman *web* di internet tumbuh dengan cepat. Selain itu, pembentukan *index* pada DAWG hanya mengandung informasi berupa huruf, *parrent, child,* dan tanda stop. DAWG *tree* harus ditambahkan informasi baru berupa alamat web sehingga dapat dimanfaatkan *seach engine* untuk mendapatkan kata yang ada pada web tertentu.

Untuk melakukan dapat melakukan *indexing* web, penulis mengembangkan metode yang biasa digunakan untuk membentuk *Directed Acyclic Word Graph*. Metode yang baru adalah metode *blow the bridge*.

Adapun algoritma *Blow The Bridge* mengambil langkah sebagai berikut:

1. Pada setiap kata yang dimasukkan, pecah kata tersebut menjadi huruf.
2. Mulai dari huruf pertama hingga satu huruf sebelum huruf terakhir, lakukan langkah 3 sampai langkah 14
3. Memeriksa apakah huruf tersebut huruf pertama atau bukan. Jika huruf tersebut huruf pertama, lakukan langkah 5.
4. Memeriksa apakah node sebelumnya merupakan node gabungan ataukah node yang baru. Jika merupakan node gabungan, lakukan langkah 5. Jika tidak, buat node baru dengan index berupa informasi huruf,*child*, serta *parrent*, dan lakukan langkah 6.
5. Periksa apakah ada node dalam *tree* yang berisi informasi huruf yang sama dengan huruf yang ingin dimasukkan.
6. Jika sudah ada node yang berisikan index huruf yang sama, huruf tersebut tidak akan dimasukkan. Namun jika belum ada node yang berisikan informasi huruf yang sama pada *tree* tersebut, buat node baru dengan index berupa informasi huruf,*child* dan *parrent*.
7. Periksa kedalaman berikutnya, dan lakukan langkah ke-2 sampai ke-4 sampai satu huruf sebelum huruf terakhir.
8. Pada huruf terakhir, periksa apakah pada kedalaman *tree* tersebut memiliki node yang berisi index tentang huruf, tanda stop dan asal web yang sama persis. Bila node tersebut sama persis, node tidak perlu ditambahkan, hanya perlu menambahkan informasi tentang *parrent*-nya. Namun bila node tersebut berbeda, maka buatlah node yang baru dengan informasi berupa huruf, *parrent,* tanda stop, dan informasi asal web.
9. Dari node terakhir yang terbentuk, mulai dari node pada kedalaman tersebut hingga node pada kedalaman kedua, periksa apakah *parrent* dari node *tersebut mempunyai informasi* huruf yang sama. Jika ya, maka gabungkan node tersebut, jika tidak, lakukan langkah 14.
10. Lakukan langkah 9 sampai kedalaman kedua atau sampai tidak ada lagi node yang berisi informasi huruf yang sama.
11. Periksa node tersebut, apakah dia mempunyai nomor *parrent* yang sama atau tidak. Jika ya, gabungkan node tersebut,.Jika tidak, lakukan langkah 14.
12. Mulai dari node tersebut, hingga satu node sebelum node terakhir, periksa *child* dari node tersebut. Jika *child* dari node tersebut memiliki informasi huruf dan tanda stop yang sama, maka gabungkan node tersebut beserta update informasi tentang *parrent, child* dan tanda stopnya. Jika tidak, lakukan langkah 14.
13. Gabungkan node terakhir dan update informasi asal webnya bila memiliki tanda

stop dan huruf yang sama.
14. Ulangi langkah ke 2 pada kata berikutnya.
15. Selesai

2.2. Metode Pencarian pada Directed Acyclic Word Graph

Metode pencarian dalam Algoritma Directed Acyclic Word Graph (DAWG) dapat menggunakan *simple two-part strategy* ataupun dengan Depth-first Search.

Simple Two-part Strategy menggunakan dua langkah dasar, yaitu:
1. Cari semua komponen kiri yang mungkin.
2. Pada komponen kiri yang tepat, cari komponen kanan yang tepat.

Adapun algoritma yang digunakan adalah sebagai berikut :

```
LeftPart(PartialWord, node N in dawg, limit) =
   ExtendRight (PartialWord, N, AnchorSquare)
   if limit > 0 then
         for each edge E out of N
               if the letter I labeling edge E is
                     in our rack then
                  remove a tile labeled 1 from the rack
                  let N' be the node reached by following edge E
                  LeftPart (PartialWord . 1, N',limit - 1 )
                  put the tile I back into the rack
ExtendRight(PartialWord, node N in dawg, square ) =
   if square is vacant then
         If N is a terminal node then LegalMove (PartialWord)
         for each edge E out of N
               if the letter I labeling edge E is in our rack
                  and 1 is in the cross-check set of square then
                     remove a tile I from the rack
                     let N' be the node reached by following
                        edge E
                     let next-square be the square to the right
                        of square
                     ExtendRight (PartialWord .I , N',
                        next-square )
                     put the tile I back into the rack
   else
         let I be the letter occupying square
         If N has an edge labeled by I that
           leads to some node N' then
             let next-square be the square to the right of square
             ExtendRight (PartialWord . I, N', next-square )
```

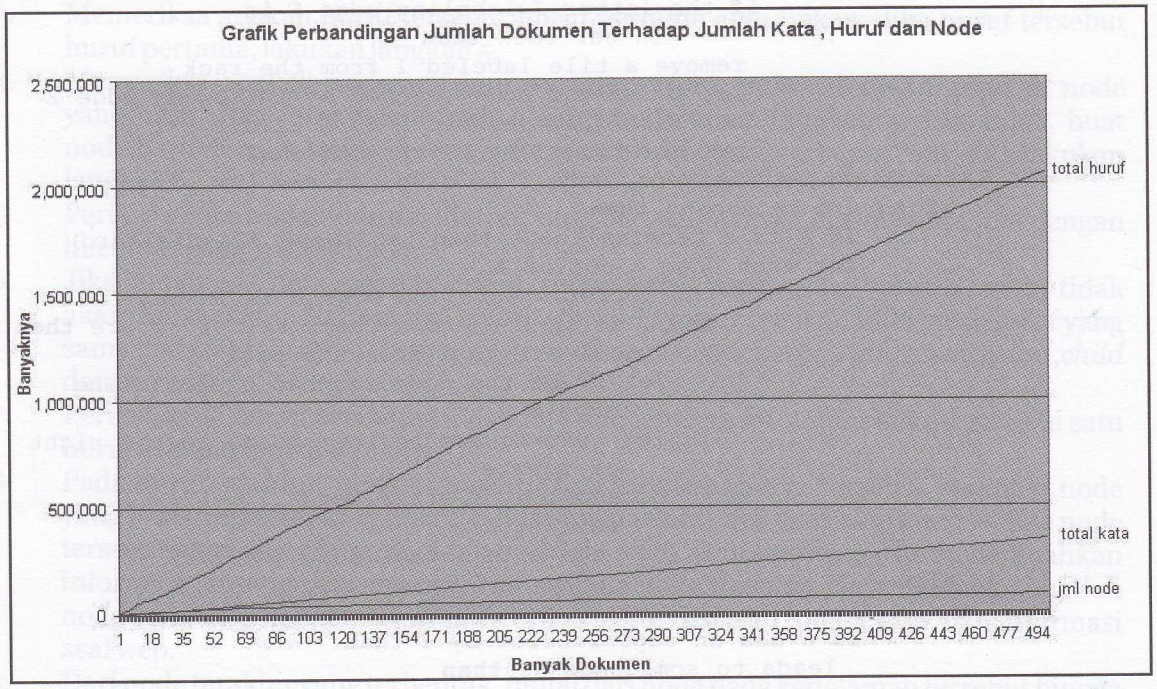
3. Hasil dan Pembahasan

Pada gambar 1 terlihat dengan jelas bahwa perkembangan node pada tiap dokumen tidak berbanding lurus dengan perkembangan jumlah kata maupun huruf. Meskipun data huruf maupun kata terus bertambah pada tiap dokumen, penambahan node DAWG sendiri sangat kecil. Hal ini dikarenakan huruf-huruf dan kata-kata tersebut banyak yang sama pada tiap dokumennya, sehingga semakin banyak kata ataupun huruf yang dijumpai, semakin besar penghematan yang dilakukan oleh DAWG.

Berdasarkan gambar 1 , tidak didapat adanya titik balik pada jumlah node DAWG. Titik balik adalah titik puncak dari suatu grafik yang kecenderungannya akan menurun

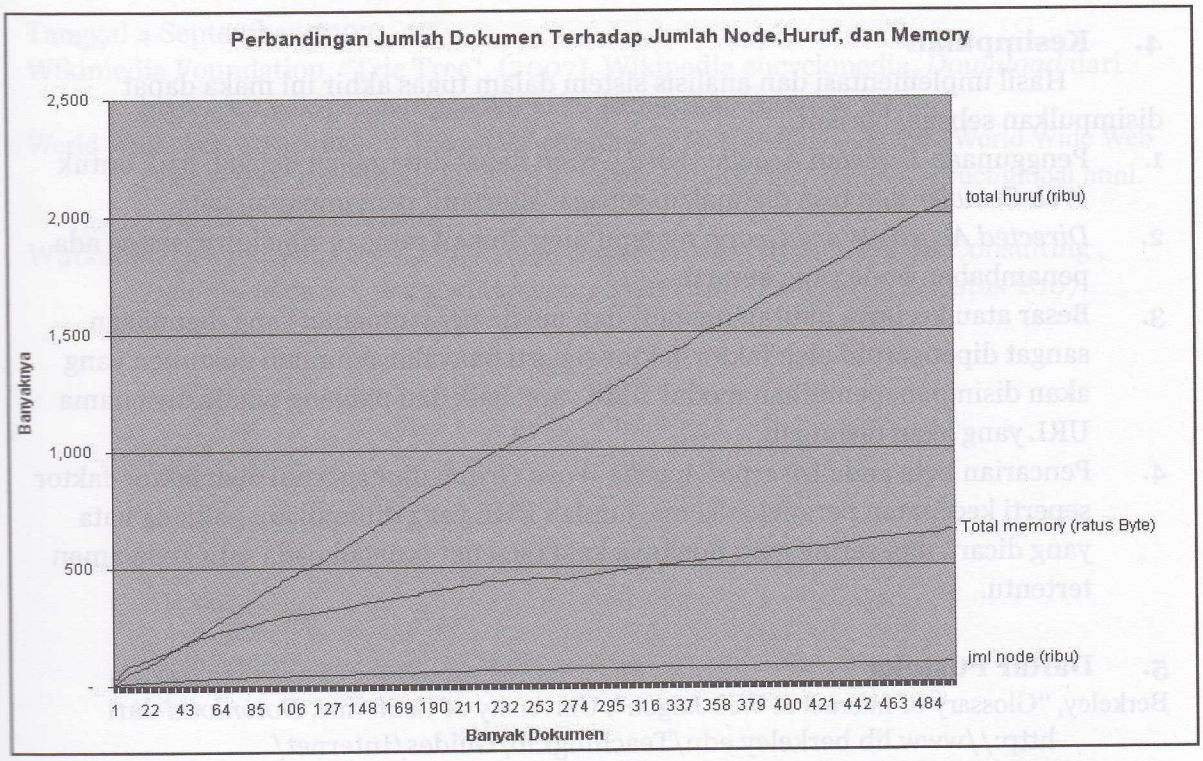
setelah titik balik terpenuhi. Hasil percobaan yang ada menunjukkan bahwa node DAWG tidak membuat grafik menjadi berbentuk parabola, akan tetapi grafik dengan bentuk lurus yang cenderung mendatar (linier) pada 500 dokumen web yang diujicobakan. Dengan grafik yang telah terbentuk, kita dapat memprediksikan dua kemungkinan yang terjadi bila dokumen web kita tambah di masa depan:

1. Jumlah node DAWG akan terus menunjukkan grafik yang lurus atau cenderung mendatar tanpa adanya penurunan yang berarti, berapapun besarnya jumlah dokumen yang dimasukkan.
2. Jumlah node DAWG akan menunjukkan grafik berbentuk parabola jika diisi dengan data yang mempunyai variasi yang banyak (misalnya diisi dengan bahasa yang berbeda-beda) dan dengan jumlah kata hingga lebih dari puluhan atau bahkan ratusan juta kata.



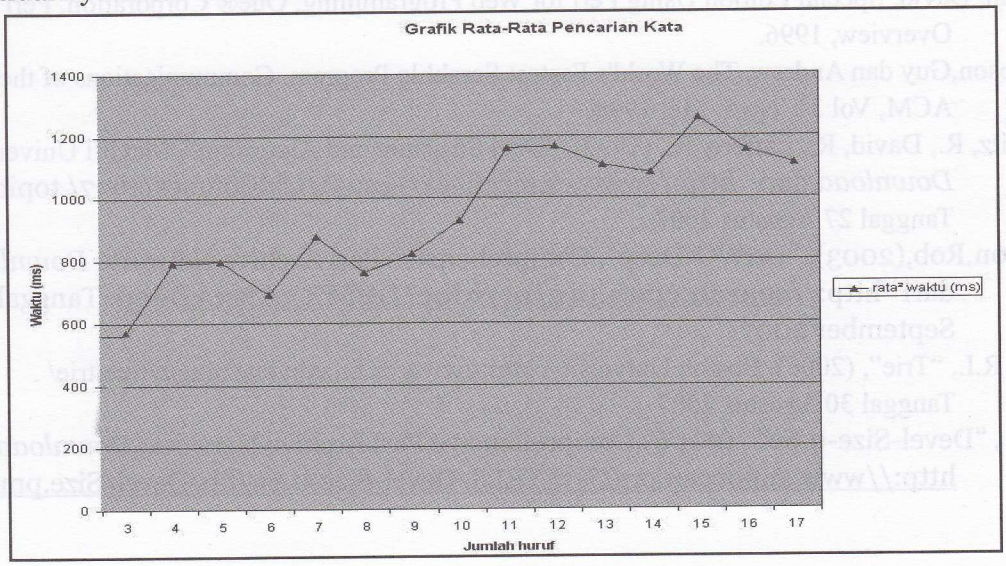
Gambar 2. Grafik Perbandingan Jumlah Dokumen terhadap Jumlah Kata, Huruf dan Node DAWG

Sedangkan pada gambar 2 terlihat dengan jelas bahwa perkembangan node pada tiap dokumen berbanding lurus dengan perkembangan jumlah ukuran memori, namun tidak berbanding lurus dengan jumlah huruf yang dimasukkan. Meskipun data huruf terus bertambah pada tiap dokumen, besarnya memori yang digunakan hanya akan bertambah seiring dengan penambahan node DAWG. Dengan kata lain, penambahan memori tidak dipengaruhi oleh seberapa banyak kata atau huruf yang dimasukkan, akan tetapi akan dipengaruhi oleh seberapa banyak node DAWG yang terbentuk.



Gambar 3. Grafik Perbandingan Jumlah Dokumen terhadap Jumlah Node DAWG, Huruf dan Total Memori

Pada Gambar 3, penggunaan *searching* hanyalah gambaran kinerja sistem. Kata-kata yang digunakan untuk *searching* merupakan kata-kata yang dipilih secara acak sehingga grafik pada gambar 3 bukanlah hasil yang dapat dianalisis secara mendalam, Akan tetapi, dapat diperkirakan bahwa selain faktor perangkat keras, jumlah URL yang ditemukan juga turut mempengaruhi waktu pencarian. Selain itu, faktor panjangnya kata juga turut mempengaruhi pencarian, meskipun hal ini tidak menciptakan perbedaan waktu yang signifikan. Faktor yang lain adalah faktor sering atau tidaknya huruf tersebut muncul.



Gambar 4. ***Grafik Pencarian Kata***

4. Kesimpulan

Hasil implementasi dan analisis sistem dalam tugas akhir ini maka dapat disimpulkan sebagai berikut:

1. Penggunaan *Directed Acyclic Word Graph* dapat diterapkan dengan baik untuk *Web Crawler* dan bisa digunakan sebagai alternatif penyimpanan kata.
2. *Directed Acyclic Word Graph* mampu menyimpan ratusan ribu huruf tanpa ada penambahan node yang berarti.
3. Besar atau kecilnya ukuran memori dan media penyimpanan yang digunakan sangat dipengaruhi oleh faktor-faktor seperti banyaknya jumlah dokumen yang akan disimpan, pemilihan modul DBM yang baik dan panjang pendeknya nama URL yang akan disimpan.
4. Pencarian kata pada *Directed Acyclic Word Graph* dipengaruhi oleh faktor-faktor seperti kecepatan perangkat keras, jumlah URL yang ditemukan, panjang kata yang dicari, dan sering atau tidaknya kata tersebut muncul pada satu kedalaman tertentu.

5. Daftar Pustaka

Berkeley, "Glossary of Internet & Web Jargon", University of California, *Download* dari http://www.lib.berkeley.edu/TeachingLib/Guides/Internet/Glossary.html.Tanggal 29 Agustus 2007

Burger,Sonya, "Effective and Intelligent Web Searching" , Sonja, Independent Schools Association of Southern Africa , *Download* dari http://196.46.113.21/ISASA/Friday%20Presentations/Sonja%20Burger%20Web%20Tips%20.ppt. Tanggal 29 Agustus 2007

Carbonell, Jaime, (2000), "Building Web Spiders: Web-Based Information Architectures",Carnegie Mellon University, *Download* dari http://www.andrew.cmu.edu/course/20-760/notes/lecture5.ppt. Tanggal 27 Agustus 2007

Cook, David dan Deborah Sellers , Launching Business on the Web, Que® Corporation, 1995.

Harlan, David, Special Edition Using Perl for Web Programming, Que® Corporation, Perl Overview, 1996.

Jacobson,Guy dan Andrew. The World's Fastest Scrabble Program. Communications of the ACM, Vol 31. No.5. Mei 1998.

Kravitz, R., David, R., Lafferty,R., (1997), "Data Structure and Algorithm", McGill University, *Download* dari http://www.cs.mcgill.ca/~cs251/OldCourses/1997/ topic26/. Tanggal 27 Agustus 2007

Kinyon,Rob,(2003), "DAWG-Deep", Comprehensive Perl Archive Network, *Download* dari http://**search.cpan.org/perldoc?DBM%3A%3ADeep**. Tanggal 4 September 2007

Pitts, R.I., "Trie", (2006), Boston University, http://www.cs.bu.edu/teaching/c/tree/trie/ . Tanggal 30 Agustus 2007

Tels , "Devel-Size-0.69" , (2006),Comprehensive Perl Archive Network, *Download* dari http://www.annocpan.org/%7ETELS/Devel-Size-0.69/lib/Devel/Size.pm .

Tanggal 4 September 2007

Wikimedia Foundation, "Trie Tree", (2007), Wikipedia encyclopedia, *Download* dari http://en.wikipedia.org/wiki/Trie. Tanggal 30 Agustus 2007

World Wide Web Consortium, "The global structure of an HTML document", World Wide Web Consortium, *Download* dari http://www.w3.org/TR/ html401/struct/global.html. Tanggal 27 Agustus 2007

Wutka, Mark and Ceal, (2004), "Directed Acyclic Word Graph", Wutka Consulting , *Download* dari www.wutka.com/dawg.html. Tanggal 27 Agustus 2007