Implementasi BFS dan Analisis Page Rank pada Google *Search Engine*

Dery Rahman Ahaddienata, 13515097[1]
Program Studi Informatika
Sekolah Teknik Elekro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
[1]13515097@std.stei.itb.ac.id

***Abstraksi*—Perkembangan web saat ini sangatlah pesat. Hal ini disebabkan oleh penambahan informasi yang besar setiap harinya pada *World Wide Web*. Besarnya informasi yang bertebaran menyebabkan *Search Engine* memiliki peranan yang penting untuk menyaring informasi yang relevan. *Search Engine* akan mengambil data penting pada setiap link yang ditelusuri dengan *crawling*. Kemudian akan disimpan sebagai *cache* yang akan digunakan untuk mendukung *query* pencarian. Proses pengambilan ini dikenal sebagai *Information Retriveval* (IR) *System*. Ada banyak algoritma yang dapat digunakan untuk melakukan *crawling*, salah satunya adalah *Breadth First Crawling*. Penerapan *crawling* selanjutnya diikuti dengan *indexing* dan *ranking*. *Ranking* dilakukan untuk menentukan halaman mana yang lebih tinggi nilainya relatif terhadap halaman lain untuk kemudian dapat ditampilkan berdasarkan *ranking* tersebut. *Ranking* pada Google dilakukan dengan menggunakan algoritma *PageRank*.**

Kata Kunci—Breadth First Search, Information Retrieval, PageRank, Web Spider

I. PENDAHULUAN

Jumlah halaman pada *World Wide Web* berkembang sangat pesat mulai dari ribuan halaman pada 1993 hingga lebih dari 2 miliar halaman pada yang terindex saat ini [2]. Hal ini merupakan suatu tantangan sendiri bagi *Information Retrieval System* untuk dapat melakukan proses pengambilan data secara tepat dan akurat. Terlebih saat ini terdapat miliaran dokumen dengan jutaan update setiap harinya, sehingga keberadaan *Search Engine* sangatlah penting.

Search Engine akan mengambil semua kumpulan informasi dan dokumen dengan bantuan *Web Crawler*. Proses pengambilan ini dilakukan dengan cara menelusuri setiap link pada setiap halaman web yang dilewatinya. Informasi penting akan diambil dan disimpan kedalam repositori lokal sebagai *cache*. Dari repositori lokal inilah pencarian *query* dari user dilakukan. Sebelumnya, *Search Engine* akan melakukan *indexing* terlebih dahulu pada repositori lokal mereka secara rutin untuk mempercepat proses pencarian *query*.

II. *WEB CRAWLER* DAN PAGE RANK

Web Crawler atau *Web Spider*, *Crawler* dan *Spider* memiliki makna yang sama [4], merupakan teknologi untuk menelusuri semua halaman web dengan cara menelusuri setiap link yang terdapat pada halaman tersebut. *Crawler* merepresentasikan web sebagai bentuk graf berarah, dengan halaman direpresentasikan sebagai simpul dan link sebagai sisi. Halaman yang pertama kali dilakukan *crawling* disebut sebagai *seed URL*. *Seed URLs* merupakan kumpulan alamat yang akan direpresntasikan sebagai akar oleh *crawler*.

Terdapat 2 macam halaman yang ditelusuri, halaman baru yang belum pernah dilakukan *crawling*, dan halaman yang sudah pernah dilakukan *crawling*. Untuk halaman yang sudah pernah dilakukan *crawling*, *crawler* akan melihat, apakah terdapat perubahan atau tidak pada halaman tersebut. Jika halaman tersebut mengalami perubahan, *crawler* akan menandai halaman tersebut untuk kemudian akan ditelusuri kembali. Semakin sering suatu halaman diperbaharui, *crawler* akan lebih sering menelusuri halaman tersebut.

Crawler akan bekerja secara rutin, namun karena keterbatasan *resource* yang tidak sebanding dengan besarnya jumlah halaman yang terdapat pada *World Wide Web*, *Crawler* tidak mungkin untuk mengambil semua halaman yang ada. Oleh karena itu, web *crawler* mempunyai aturan sendiri untuk melakukan penelusuran. Ketentuan ini disebut sebagai *re-visiting policy*. Pada dasarnya, *re-visiting policy* bergantung kepada 2 parameter, yaitu *freshness* dan *age*. *Freshness* merupakan ukuran seberapa akurat halaman yang ditelusuri dengan halaman yang berada pada repositori lokal. Sedangkan *age* merupakan ukuran seberapa lama halaman tersebut dilakukan crawling terakhir kalinya. *Web Crawler* akan menandai halaman yang berubah sebagai halaman yang akan ditelusuri lagi dikemudian hari.

Dua tipe *re-visiting policy* yaitu *Uniform policy* dan *Proportional policy* atau bisa disebut *non-uniform policy* [4]. Pada *uniform policy*, *web crawler* akan melakukan penelusuran secara merata tanpa memandang apakah halaman tersebut mengalami perubahan atau tidak. Sedangkan pada *proportional policy*, *web crawler* akan melakukan penelusuran jika halaman tersebut mengalami frekuensi perubahan yang banyak.

Web crawler merupakan teknologi yang sangat rentan karena berinteraksi langsung dengan ribuan server web. Setiap halaman web mempunyai aturan sendiri. Ada bagian halaman web yang boleh ditelusuri *crawler* ada yang tidak. Dalam dunia *crawling*, ada kebijakan bersama yang harus disepakati bersama. Kebijakan ini bertujuan untuk mengontrol crawler, bagian mana dari suatu web yang boleh dikunjungi mana yang tidak.

Pada search engine, terdapat 2 isu penting mengenai web crawler. Yang pertama adalah strategi *crawling*, yang kedua adalah optimisasi *crawling* [2]. Pada strategi *crawling*, terdapat beberapa macam algoritma yang dapat digunakan, yaitu *breadth first search algorithm*, *depth first search algorithm*, *page rank algorithm*, *path-ascending crawling algorithm*, *focused crawling algorithm*, *genetic algorithm*, *naive bayes classification algorithm*, dan *HITS algorithm* [4]. Kebanyakan *search engine* mengadopsi *web crawler* yang menggunakan strategi *breadth first search* [5].

Setelah informasi penting disimpan kedalam repositori lokal, search engine akan melakukan *indexing*, yaitu melakukan *index* pada setiap halaman agar dapat mempercepat proses pencarian *query*. Meskipun begitu *indexing* tidak dapat memastikan tingkat keakuratan halaman web tersebut, sehingga pencarian query yang dilakukan oleh user tidak selalu menghasilkan halaman web yang relevan pada halaman pertama pencarian. Oleh karena itu, untuk menjamin keakuratan hasil pencarian, *search engine*, khususnya Google, menerapkan sistem page rank.

Makalah ini akan menjelaskan bagaimana BFS diterapkan pada *web crawler* dan juga bagaimana *sistem page rank* menghasilkan hasil pencarian yang berkualitas.

III. *Web Search Engine* : Google

Teknologi search engine telah berkembang pesat dalam 10 tahun terakhir. Pada tahun 1994, search engine pertama World Wide Web Worm (WWWW) dapat melakukan *indexing* sebanyak 110.000 halaman web. November 1997, *search engine* terbesar, Watch, berhasil melakukan *indexing* 2 juta hingga 100 juta halaman web. Google memperkirakan, bahwa halaman web pada tahun 2000 akan berkembang menjadi lebih dari jutaan halaman [1].

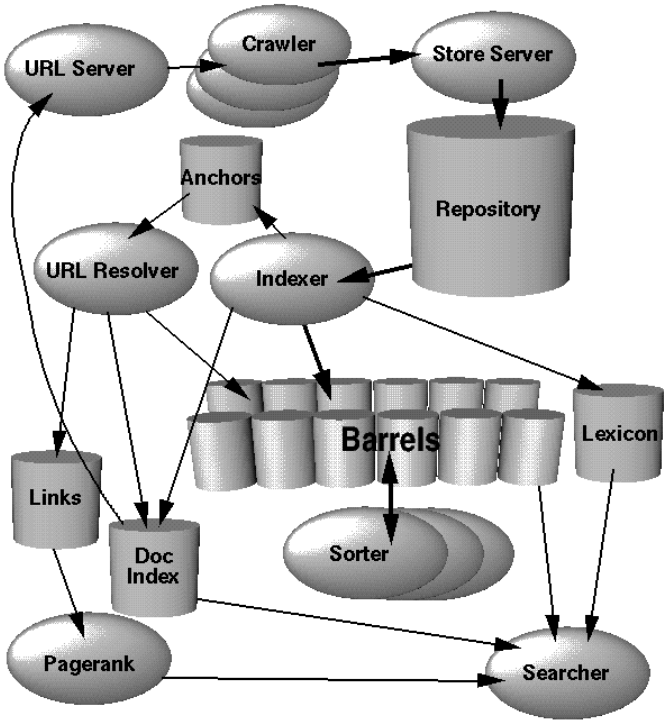
Hal ini merupakan tantangan tersendiri bagi teknologi *search engine* pada masa itu. Resource masih minim, sehingga perlu adanya pendekatan yang efektif untuk melakukan proses ribuan *gigabytes* data dalam repositori lokal *search engine*.

Pada tahun 1997, pencarian *query* dilakukan dari jutaan halaman yang telah dilakukan indexing. Pencarian ini optimal, namun tidak menghasilkan hasil yang diharapkan. Kebanyakan dari hasil pencarian merupakan halaman yang tidak relevan dengan apa yang diinginkan oleh user. User biasanya akan melihat 10 halaman teratas hasil pencarian, namun, kebanyakan *search engine* pada masa tersebut menghasilkan hasil pencarian yang tidak diurutkan secara tepat. Oleh karena itu untuk menjamin hasil yang presisi, Google menerapkan algoritma Page Rank sebagai ide dari hasil tesis salah satu pendirinya, Larry Page.

Google lahir sebagai buah pemikiran Larry Page dan Sergey Brin ketika mengenyam pendidikan PHD di Stanford University pada tahun 1998 [3]. Ide utama dibalik *search engine* ini adalah halaman web yang saling berhubungan satu-sama lain melalui link. Keterhubungan antar halaman ini direpresentasikan sebagai graf kompleks yang masif. Secara periodik, *crawler* akan melakukan penelusuran melalui link tersebut.

A. Arsitektur Google *search engine*

Google merupakan *search engine* yang menarik untuk dipelajari. URLserver mengirimkan list berbagi macam *seed* URL kedalam *crawler*. Google menggunakan *multiple crawler* pada proses *crawling*nya. Sehingga *crawling* dapat dilakukan secara parallel. Baik URLserver maupun *crawler* diimplementasikan dengan bahasa Python [1].



Gambar 1. *High level* arsitektur Google

Setiap halaman web yang berhasil ditelusuri akan dimasukan kedalam storeserver. Storeserver akan mengkompres halaman tersebut, kemudian akan disimpan kedalam repositori. Setiap ID halaman baru yang telah ditelusuri disebut sebagai docID. Didalam repositori ini, indexer dan sorter akan melakukan tugasnya untuk melakukan *indexing*. Setelah itu Page Rank akan melakukan tugasnya memberi nilai pada setiap halaman tersebut dengan beberapa kali iterasi.

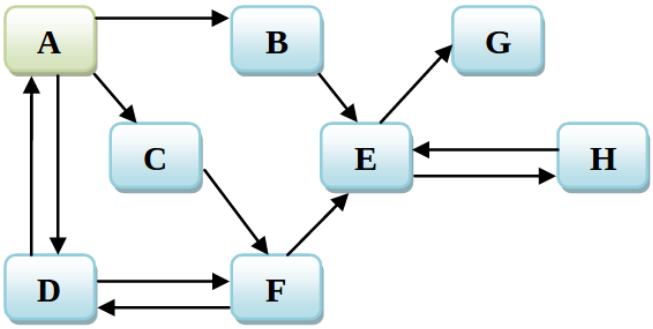
Pada arsitektur standar, *web search engine* memiliki 3 komponen penting, yaitu *crawling*, *indexing*, dan *query* [5]. Pada *search engine* Google, page rank akan dijalankan sebelum memulai pencarian *query*, sehingga dihasilkan hasil pencarian dengan urutan yang bergantung pada nilai kualitas relatif terhadap halaman lain.

IV. Dasar Teori

Makalah ini akan difokuskan untuk membahas strategi *web crawler* yang paling umum diaplikasikan pada *search engine*, yaitu BFS. Selain itu, makalah ini juga difokuskan untuk membahas bagaimana Page Rank bekerja untuk mendapatkan hasil pencarian yang berkualitas.

A. Web

Web merupakan halaman web dengan representasi graf berarah. Halaman web sebagai simpul dan *link* sebagai sisinya. Sisi yang mengarah kedalam simpul merepresentasikan bahwa halaman tersebut mempunyai *link* masuk / *inlink*. Sedangkan sisi yang mengarah keluar simpul merepresentasikan bahwa halaman tersebut mempunyai *link* keluar / *outlink*. Penggunaan *inlink* dan *outlink* akan berpengaruh pada nilai page rank suatu halaman web.

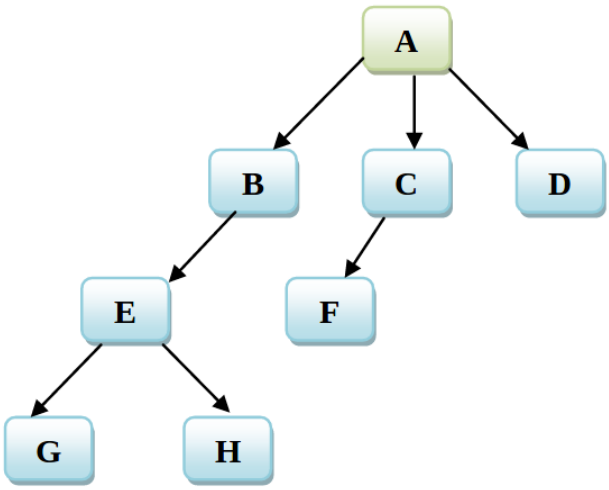


Gambar 2. Contoh graf pada web

B. BFS

Strategi yang paling sering digunakan untuk melakukan *crawling* adalah dengan *Breadth First Search*. BFS merupakan algoritma pencarian merentang dari suatu graf. Pertama-tama simpul akar akan diekspansi kemudian menelusuri semua simpul tetangga yang satu level dengan tetangga lainnya. Selanjutnya suatu level telah ditelusuri akan diekspansi lagi dan lakukan hal yang serupa kepada tetangga-tetangga selanjutnya. Hal ini dilakukan terus menerus hingga suatu kedalaman tertentu.

Penerapan BFS menggunakan *Queue* FIFO / *First In First Out*. Simpul tetangga yang diekspansi dimasukan kedalam *queue*. Simpul yang akan dikunjungi merupakan simpul yang diambil dari *queue*. Simpul yang telah dikunjungi ditandai sebagai simpul yang telah ditelusuri.



Gambar 3. Contoh hasil penerapan BFS

Dari gambar 2, yang menjadi simpul akar adalah simpul A. Selanjutnya ekspansi simpul A. Ekspansi ini menghasilkan simpul tetangganya yaitu simpul B, C, dan D. Masukan simpul-simpul tersebut kedalam *queue*.

B	C	D								

Tandai simpul A sebagai simpul yang telah dikunjungi. Kemudian keluarkan simpul B dari queue dan telusuri. Ekspansi lagi simpul B, sehingga menghasilkan simpul tetangganya yaitu simpul E. Masukan kedalam queue, dan tandai bahwa simpul B telah dikunjungi. Lakukan secara berulang kali hingga tidak ada lagi simpul yang dapat dikunjungi.

C	D	E								

D	E	F								

E	F									

F	G	H								

G	H									

H										

pseudo code BFS

```
BFS(Graf, akar):

    buat set kosong S
    buat queue kosong Q

    S.insert(akar)
    Q.enqueue(akar)

    while Q tidak kosong:
        curr = Q.dequeue()
        for setiap simpul n tetangga curr:
            if n tidak dalam S:
                S.insert(n)
                Q.enqueue(n)
```

Listing 1. Pseudo code BFS

C. Page Rank

Suatu halaman web dikatakan berkualitas jika mempunyai page rank yang tinggi. Page rank yang tinggi didasarkan pada berapa banyak link yang masuk/keluar pada halaman web tersebut. Semakin banyak link yang masuk (*link* dari halaman web lain yang mengarah ke halaman tersebut) semakin penting pula halaman tersebut. Semakin tinggi page rank suatu halaman web, semakin besar kemungkinan halaman web tersebut ditampilkan dihalaman awal pencarian.

Sistem ini dapat dianalogikan seperti makalah. Makalah yang banyak di*recite* oleh makalah lain, mempunyai nilai yang

relatif tinggi. Hal ini dikarenakan makalah tersebut penting sebagai acuan dari makalah yang lain. Makalah yang di*recite* oleh makalah yang mempunyai nilai yang tinggi, akan mempunyai nilai yang tinggi juga relatif terhadap makalah lain. Hal ini berlaku juga dengan halaman web.

$$PR(A) = (1-d) + d\,(PR(T1)/C(T1) + ... + PR(Tn)/C(Tn))$$

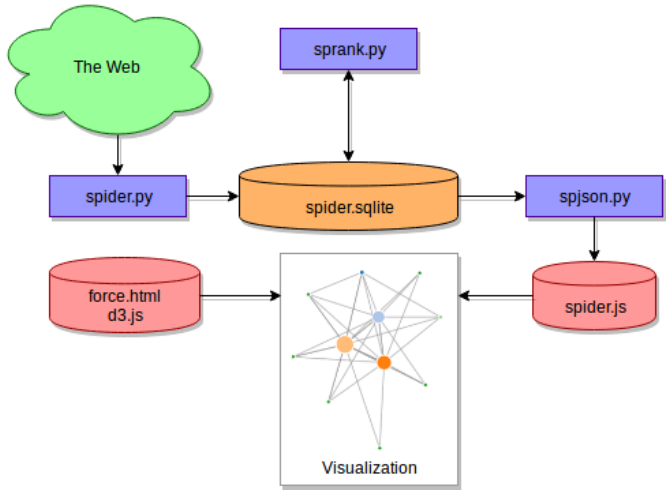
Asumsi bahwa terdapat halaman web *A* berhubungan dengan *T1* ... *Tn* halaman web lain. *PR(A)* didefinisikan sebagai nilai page rank pada halaman *A*. Parameter *d* merupakan faktor dumping yang dapat diset antara 0 dan 1. Google menggunakan nilai 0.85 sebagai faktor dumpingnya [1].

PR(A) dapat dihitung dengan menggunakan algoritma iteratif hingga didapatkan nilai page rank yang cukup stabil. Penentuan nilai page rank awal pada suatu halaman dibebaskan. Darimana saja nilai page rank awal dipilih, hasil akhir dari proses iteratif akan selalu menuju ke distribusi normal. Dalam artian rata-rata dari seluruh pagerank tersebut adalah 1.0.

V. *TOOLS* DAN METODE

Crawling dilakukan dengan menggunakan program spider[2] yang ditulis menggunakan python. Percobaan dilakukan dengan 2 situs, yang pertama menggunakan situs *dummy* deryrahman.github.io/dummy_web, dan yang kedua adalah itb.ac.id. Pada percobaan situs *dummy* dilakukan *crawling* sebanyak 11 halaman web dummy sedangkan pada situs itb dilakukan *crawling* sebanyak 50 dan 500 halaman. Hasil *crawling* disimpan dalam *database* SQLite. Selanjutnya dilakukan *ranking* pada semua halaman web dengan menggunakan program pagerank[2]. Penentuan nilai page rank dilakukan secara iteratif hingga mencapai distribusi normal. Kurang lebih arsitektur mesin yang digunakan terdiri dari spider (spider.py), page rank generator (sprank.py), database (spider.sqlite), dan visualizer[2].

[2]Program web spider, page rank, dan visualizer menggunakan program open source yang dikembangkan oleh Charles Severance Ph.D. yang telah dimodifikasi menjadi model BFS oleh penulis. Referensi program [6]

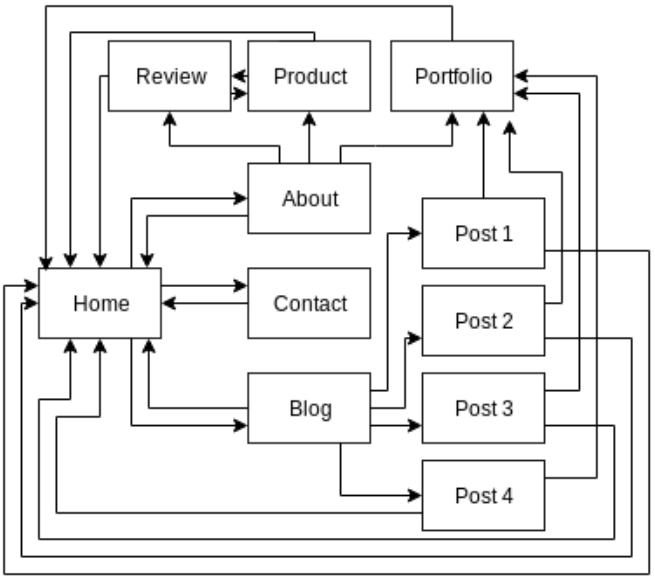


Gambar 4. Arsitektur percobaan *crawling*

Pada situs *dummy*, halaman web dibuat sedemikian rupa sehingga menyerupai situs asli dengan total terdapat 11 halaman web. Dengan rincian 1 halaman home, 3 halaman page, 4 halaman post, dan 3 sembarang halaman.

Halaman	**Outlink**	**Inlink**
Home	About, Contact, Home	About, Contact, Blog, Post 1, Post 2, Post 3, Post 4, Review, Product, Portfolio
About	Home, Review, Product, Portfolio	Home
Contact	Home	Home
Blog	Home, Post 1, Post 2, Post 3, Post 4	Home
Post 1 … Post 4	Home	Blog
Review	Home, Product	About, Product
Product	Home, Review	About, Review
Portfolio	Home	About, Post 1, Post 2, Post 3, Post 4

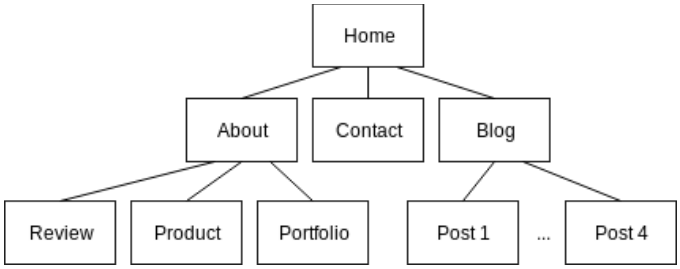
Tabel 1. Struktur pada web dummy



Gambar 5. Graf pada web *dummy*

VI. HASIL DAN ANALISIS

Hasil pencarian BFS dari web *dummy* terbagi menjadi 3 level. Dihasilkan representasi pohon melebarnya



Gambar 6. Representasi pohon hasil BFS

Home merupakan halaman yang pertamakali dikunjungi. *Crawler* akan memasukan *link* yang ditemukan pada halaman tersebut kedalam *queue*. Selanjutnya semua halaman pada level 2 ditelusuri terlebih dahulu, dan dilakukan ekspansi lagi jika menemukan *link* yang terdapat pada halaman tersebut. Jumlah *link* yang mengarah menuju halaman lain (*outlink*) akan disimpan kedalam *database*.

Ketika *crawler* mencoba mengekspansi halaman About, terdapat 4 kandidat *outlink*, yaitu Home, Review, Product, dan Portfolio. Karena link Home sudah dikunjungi, maka halaman ini tidak diekspansi. Jumlah inlink untuk halaman home akan bertambah.

```
Enter web url or enter: http://deryrahman.github.io/dummy_web
['http://deryrahman.github.io/dummy_web']
How many pages:11
1 http://deryrahman.github.io/dummy_web (280)
2 http://deryrahman.github.io/dummy_web/about.html (378)
3 http://deryrahman.github.io/dummy_web/contact.html (148)
4 http://deryrahman.github.io/dummy_web/blog.html (523)
5 http://deryrahman.github.io/dummy_web/review.html (230)
6 http://deryrahman.github.io/dummy_web/product.html (230)
7 http://deryrahman.github.io/dummy_web/portfolio.html (149)
8 http://deryrahman.github.io/dummy_web/post/post1.html (237)
9 http://deryrahman.github.io/dummy_web/post/post2.html (237)
10 http://deryrahman.github.io/dummy_web/post/post3.html (237)
11 http://deryrahman.github.io/dummy_web/post/post4.html (237)
--- 0 hours 0 minutes 7 seconds ---
```

Gambar 7. Hasil penelusuran dengan BFS

Inlink dan *outlink* akan digunakan sebagai metode perankingan oleh algoritma Page Rank. Sebelum melakukan iterasi, semua halaman mempunyai nilai Page Rank sebesar 1.0. Iterasi dilakukan hingga mencapai distribusi normal, yaitu, total page rank dibagi jumlah semua halaman adalah 1.0. Setelah dilakukan iterasi sebanyak 100 kali, hasil nilai page rank dengan sempurna mencapai titik distribusi normal pada iterasi ke 2.

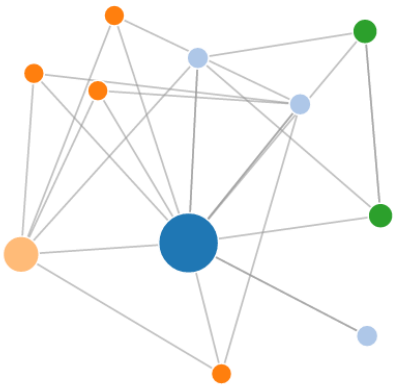
```
(10, 1.0, 5.45, 1, u'http://deryrahman.github.io/dummy_web')
(5, 1.0, 2.25, 9, u'http://deryrahman.github.io/dummy_web/portfolio.html')
(2, 1.0, 0.75, 10, u'http://deryrahman.github.io/dummy_web/review.html')
(2, 1.0, 0.75, 11, u'http://deryrahman.github.io/dummy_web/product.html')
(1, 1.0, 0.3333333333333333, 2, u'http://deryrahman.github.io/dummy_web/about.html')
(1, 1.0, 0.3333333333333333, 3, u'http://deryrahman.github.io/dummy_web/contact.html')
(1, 1.0, 0.3333333333333333, 4, u'http://deryrahman.github.io/dummy_web/blog.html')
(1, 1.0, 0.2, 5, u'http://deryrahman.github.io/dummy_web/post/post1.html')
(1, 1.0, 0.2, 6, u'http://deryrahman.github.io/dummy_web/post/post2.html')
(1, 1.0, 0.2, 7, u'http://deryrahman.github.io/dummy_web/post/post3.html')
(1, 1.0, 0.2, 8, u'http://deryrahman.github.io/dummy_web/post/post4.html')
11 rows.
```

Gambar 8. Hasil dari page rank seluruh halaman

Url	Page Rank
http://deryrahman.github.io/dummy_web	5.45
http://deryrahman.github.io/dummy_web/portfolio.html	2.25
http://deryrahman.github.io/dummy_web/review.html	0.75
http://deryrahman.github.io/dummy_web/product.html	0.75
http://deryrahman.github.io/dummy_web/about.html	0.33
http://deryrahman.github.io/dummy_web/contact.html	0.33
http://deryrahman.github.io/dummy_web/blog.html	0.33
https://deryrahman.github.io/dummy_web/post/post1.html	0.2
https://deryrahman.github.io/dummy_web/post/post2.html	0.2
https://deryrahman.github.io/dummy_web/post/post3.html	0.2
https://deryrahman.github.io/dummy_web/post/post4.html	0.2

Tabel 2. Hasil perhitungan page rank

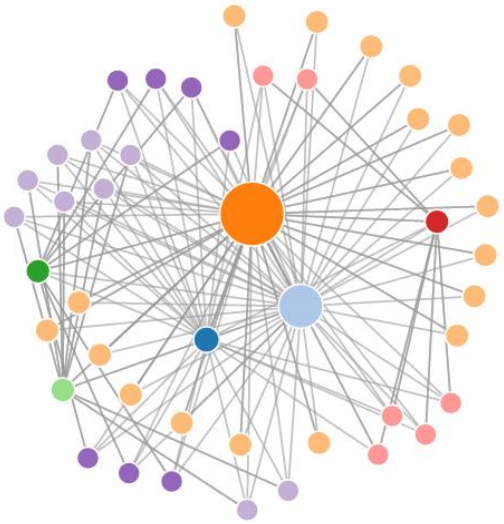
Nilai page rank tersebut dapat dijadikan sebagai acuan tingkat kepentingan suatu halaman web. Halaman home, merupakan halaman yang paling banyak diacu oleh halaman lain, sehingga nilai kepentingan relatif lebih tinggi dibanding dengan halaman lainnya. Berbeda dengan halaman post yang hanya diacu oleh satu halaman saja. Nilai page rank dapat dijadikan salah satu parameter dalam penccarian oleh *search engine*.



Gambar 9. Representasi page rank secara visual pada web dummy

Halaman	Simbol
Home	
Portfolio	
Review dan Product	
About, Contact, dan Blog	
Post 1 … Post 4	

Percobaan kedua dilakukan pada website itb.ac.id, dengan 2 kali percobaan. Percobaan pertama *crawler* menelusuri 50 halaman, sedangkan pada percobaan berikutnya, *crawler* menelusuri 500 halaman. Pada percobaan pertama, untuk melakukan penelurusan 50 halaman, diperlukan waktu 4 menit 28 detik. Kemudian dilakukan iterasi hingga mencapai distribusi normal. Pada percobaan 500 halaman memerlukan waktu kurang lebih 3 jam. Kemudian dilakukan iterasi hingga mencapai distribusi normal.



Gambar 10. Representasi page rank secara visual pada web itb.ac.id sebanyak 50 halaman

VII. *KESIMPULAN*

Implementasi BFS pada *web crawler* merupakan cara yang efektif untuk dapat menjangkau seluruh halaman web. Crawler menggunakan strategi BFS akan membuat representasi pohon melebar dari suatu graf. Pada *World Wide Web*, keterhubungan antar halaman direpresentasikan dengan baik dengan graf berarah. Strategi ini merupakan strategi yang paling umum digunakan pada *search engine*. Dengan menggunakan BFS, *search engine* dapat melakukan pengambilan data pada halaman, kemudian memasukannya kedalam repositori. Didalam repositori, semua halamn web tersebut akan dilakukan *indexing*, sehingga dapat mempercepat pencarian *query*.

Search engine yang baik akan menampilkan hasil pencarian terurut sesuai dengan tingkat relevansi dokumen terhadap keyword pencarian. Salah satu parameter yang dapat dijadikan sebagai acuan adalah dengan menggunakan Page Rank. Page Rank merupakan algoritma yang dapat menentukan tingkat kepentingan suatu halaman web dibanding halaman web yang lain. Page Rank merupakan buah pikir pendiri Larry Page selaku pendiri Google. Dengan bantuan page rank, hasil pencarian pada Google jauh lebih baik dibanding dengan search engine lainnya.

ACKNOWLEDGMENT

Saya mengucapkan terima kasih kepada Allah SWT atas segala yang diberikan-Nya sehingga makal ini dapat selesai. Ucapan terima kasih juga ditujukan kepada kedua orang tua saya yang selalu mencupport sehingga dapat menjalankan perkuliahan di Institut Teknologi Bandung dengan baik. Tak lupa saya ingin mengucapkan terima kasih kepada dosen-dosen mata kuliah strategi algoritma, yaitu Dr. Ir. Rinaldi Munir M.T., Dr. Nur Ulfa Maulidevi S.T., dan Dr. Masayu Leylia Khodra S.T. atas segala ilmu yang telah dibagikan.

REFERENSI

[1] S. Brin and L. Page, "The Anatomy of a Large-Scale Hypertextual Web Search Engine," *Proc. Seventh World Wide Web Conf. (WWW7), International World Wide Web Conference Committee (IW3C2)*, 1998.

[2] V. Shkapenyuk, T. Suel, "Design and Implementation of a High-Performance Distributed Web Crawler," New York : CIS Department Polytechnic University Brooklyn, NY 11201

[3] K. K. Lavania, S. Jain, M. K. Gupta, and N. Sharma, "Google: A Case Study (Web Searching and Crawling)," *International Journal of Computer Theory and Engineering, Vol. 5, No. 2*, April 2013

[4] A. V. Singh , Vikas , and A. Mishra, "A Review of Web Crawler Algorithms," *(IJCSIT) International Journal of Computer Science and Information Technologies, Vol. 5 (5)* , 2014, 6689-6691

[5] A. G. K. Leng, Ravi Kumar P, A. Kumar Singh, and R. Kumar Dash, "PyBot: An Algorithm for Web Crawling," *International Conference on Nanoscience, Technology and Societal Implications (NSTSI)*, 2011

[6] C. Severance, "Page Rank and Web Searching", 18 May 2017, http://pythonlearn.com/code/pagerank/

[7] Chain Singh, Kuldeep Singh, and Hansraj, "A Survey on Web Crawling Algorithms Strategies," *International Journal of Research in Advent Technology (E-ISSN: 2321-9637) Special Issue National Conference "IAEISDISE 2014"*, 12-13 September 2014

[8] P. Kumari, G. Kakhani, "Comparative Analysis of Web PageRank Algorithm using DFS and BFS Crawling," *International Journal of Science and Research (IJSR) ISSN (Online): 2319-7064*

[9] I. Rogers, "The Google Page Rank Algorithm and How It Works," IPR Computing Ltd.

[10] A. Tobert, "The Google spider & you: What you need to know to get your site indexed", 18 May 2017, https://www.wordtracker.com/academy/learn-seo/technical-guides/google-spider-crawling

[11] Yen-Yu Chen Qingqing and Gan Torsten Suel, "Local Methods for Estimating PageRank Values," *New York : CIS Department Polytechnic University Brooklyn, NY 11201*

[12] C. Olston and M. Najork, "Web Crawling," *Foundations and Trends in Information Retrieval Vol. 4, No. 3 (2010) 175–246 2010 DOI: 10.1561/1500000017*

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, 19 Mei 2017

Dery Rahman Ahaddienata, 13515097