Aplikasi Teori Graf dalam Algoritma PageRank dan Optimasi SEO Website

I Kadek Yuda Budipratama Giri / 13516115
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
yudaikadek22@gmail.com

***Abstrak*—Dalam pembuatan *website*, selalu diinginkan trafik pengunjung yang ramai. Akan tetapi, pengguna tidak akan tahu tentang *website* yang kita buat apabila *website* kita tidak muncul pada *search engine* yang digunakan oleh pengguna. Oleh sebab itu, *website developer* harus mencari cara untuk mengoptimasi peringkat laman dari *website* yang dibuat dengan cara mengoptimasi teknik SEO (*Search Engine Optimization*) yang digunakan. Bagaimanakah algoritma PageRank yang digunakan untuk mengurutkan peringkat kemunculan suatu website yang dicari dengan *search engine*? Bagaimana informasi mengeai PageRank dapat membantu menaikkan peringkat laman suatu *website*? Dengan teori Graf, dapat digambarkan bagaimana algoritma PageRank.**

***Keywords*— *Website*, SEO, PageRank, Graf.**

I. PENDAHULUAN

Sejak Internet muncul pada tahun 1969, jutaan *website* sudah *online* di Internet dan mencapai puncaknya pada tahun 2014 di angka 968 juta website di dunia. Bagaimana caranya agar kita dapat memperoleh suatu halaman *website* sesuai keinginan kita? Jawabannya adalah *search engine*.

Search engine pertama bernama "Archie" diciptakan pada tahun 1990 oleh Alan Emtage, mahasiswa di McGill University Montreal. *Search engine* ini menggabungkan penggunaan *regular expression* dengan *web crawler* yang berupa *bot* (program yang mengotomatisasi suatu perintah berulang-ulang dalam waktu yang sangat cepat).

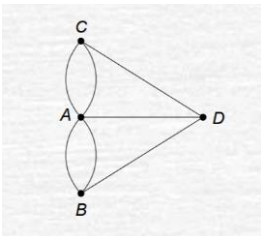
S*earch engine* bekerja dengan cara membaca masukkan dari pengguna, lalu memeriksa apakah ada kesalahan pengejaan atau ada ejaan yang lebih populer, lalu masukkan tersebut menjadi *query* untuk mencari konten yang diminta pada indeks yang berisi website yang telah dikunjungi *crawler* sebelumnya. Jika tidak ditemukan, maka pencarian ke Internet dilakukan oleh *crawler*. Indeks yang ada juga diperbaharui isinya oleh *crawler*. Jika ditemukan pranala berisi laman yang sesuai konten, maka akan dibuat sebuah list pranala yang sesuai. Hasilnya pun diurutkan sesuai relevansinya berdasarkan konten, pranala kutipan, dan ukuran penggunaan data untung memunculkan *website* tersebut.

Untuk mengurutkan website mana yang ditampilkan lebih dulu, Google menerapkan Algoritma bernama PageRank. PageRank adalah algoritma yang dikembangkan oleh Lawrence Page dan Sergey Brin pada tahun 1998. Algoritma PageRank ini berhasil membawa Google menjadi *search engine* yang paling banyak dipakai. Algoritma yang dipakai memanfaatkan teori Graf yang menggambarkan sebuah *website* sebagai simpul dan pranala yang merujuk ke *website* tersebut atau rujukan dari halaman tersebut sebagai sisi dari Graf.

Untuk meningkatkan urutan kemunculan suatu *website*, dibutuhkan suatu strategi untuk membuat algoritma *search* engine menempatkan kemunculan *website* pada posisi sebaik mungkin. Strategi yang digunakan ini bernama SEO (*Search Engine Optimization*). SEO merupakan proses untuk mendapatkan trafik pengunjung berdasarkan hasil pencarian *search engine*. Dengan kata lain, SEO membuat *search engine* memberikan *website* kita peringkat yang lebih baik dalam hasil pencariannya.

II. DASAR TEORI

A. Definisi Graf



Gambar 1. Contoh Graf
(Sumber: [3])

Graf adalah struktur diskrit yang digunakan untuk merepresentasikan hubungan antar objek-objek diskrit. Graf terdiri atas simpul (*vertices* atau *node*) dan sisi (*edges* atau *arches*) yang menjadi penghubung antarsimpul. Graf dapat direpresentasikan sebagai pasangan himpunan (V,E), di mana:

V = himpunan tidak-kosong dari simpul
$= \{ v_1, v_2, \ldots, v_n \}$

dan

E = himpunan sisi
$= \{ e_1, e_2, \ldots, e_n \}$

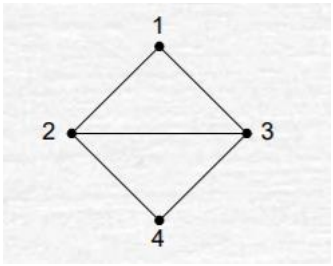
Sementara itu, sisi e yang menghubungkan simpul v_1 dan v_2 dapat dituliskan menjadi

$$e = (v_1, v_2)$$

Berdasarkan kompleksitasnya, graf tebagi menjadi dua jenis:

1. Graf sederhana (*Simple Graph*)

   Graf sederhana tidak mengandung sisi ganda maupun sisi kalang.



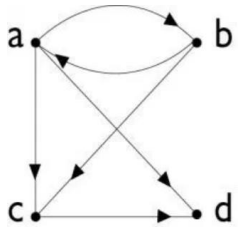
Gambar 2. Contoh Graf sederhana
(Sumber: [3])

2. Graf tidak-sederhana

   Graf tidak sederhana mengandung sisi ganda atau kalang pada grafnya. Sisi ganda adalah sisi-sisi yang menghubungkan pasangan simpul yang sama, sementara sisi kalang adalah sisi yang simpul awal dan tujuannya sama.

   a. Graf ganda (*Multigraph*)

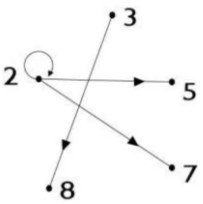
      Graf ganda adalah graf yang mengandung sisi ganda.



Gambar 3. Contoh Graf ganda
(Sumber: http://www.catatanrobert.com/relasi-dan-representasinya-dengan-tabel-matriks-dan-graf-berarah/)

   b. Graf Semu (*Pseudograph*)

      Graf semu adalah graf yang mengandung kalang.



Gambar 4. Contoh Graf semu
(Sumber: http://www.catatanrobert.com/relasi-dan-representasinya-dengan-tabel-matriks-dan-graf-berarah/)

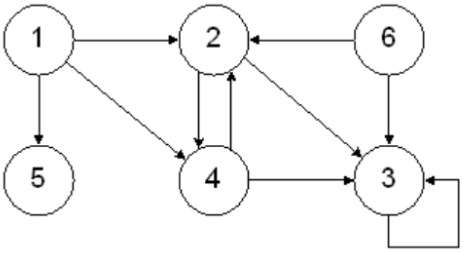
Selain itu, graf juga dapat dbedakan berdasarkan orientasi arah pada sisinya menjadi dua jenis:

1. Graf tak-berarah (*Undirected Graph*)

   Graf tak-berarah tidak mempunyai orientasi arah, sehingga urutan sisi yang dihubungkan dengan sisi tidak diperhatikan. Dengan kata lain, jika v_i dan v_j merupakan simpul dari sebuah graf, $e = (v_i, v_j) = (v_j, v_i)$. Graf dalam gambar 1, 2, 3, dan 4 merupakan contoh dari graf tak-berarah.

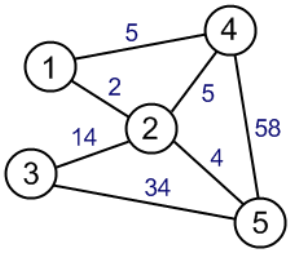
2. Graf berarah (*Directed Graph*)

   Graf berarah adalah graf yang diberikan orientasi arah pada sisinya, sehingga jika v_i dan v_j merupakan simpul dari sebuah graf $e = (v_i, v_j) \neq (v_j, v_i)$. Pada sisi e, Simpul v_i adalah simpul asal, sementara simpul v_j adalah simpul terminal.



Gambar 5. Contoh Graf berarah
(Sumber: http://faculty.cs.niu.edu/~freedman/340/340notes/340graph.htm)

Graf sendiri dapat mempunyai bobot tiap sisi. Bobot yang dimiliki oleh tiap sisi graf dapat menyatakan hubungan antar simpul dalam bentuk objektif, seperti jumlah pranala yang merujuk ke suatu laman ke laman lain, jarak antar kota, ongkos produksi, dan sebagainya.



Gambar 6. Contoh Graf berbobot
(https://math.stackexchange.com/questions/1136972/how-to-normalize-edges-weight-between-0-and-1)

B. Representasi Graf

Agar dapat diolah, graf harus direpresentasikan di dalam memori komputer. Beberapa alternative yamg dapat digunakan untuk merepresentasikan graf:

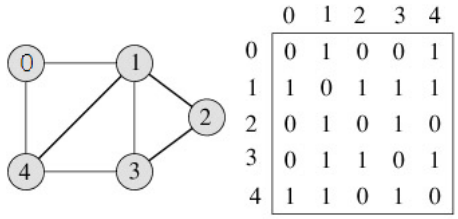
1. *Adjacency Matrix* (Matrix Ketetanggaan)

   Dalam graf, terdapat istilah *adjacent* yang berarti bertetangga. Dua buah simpul dikatakan *adjacent* atau bertetangga jika keduanya terhubung langsung dengan sebuah sisi. Jadi, simpul v_i bertetangga dengan simpul v_j jika (v_i, v_j) adalah sebuah sisi pada graf G.

   Adjacency Matrix adalah representasi yang paling umum dalam merepresentasikan graf. Misal $G = (V, E)$ adalah graf yang mempunyai n buah simpul ($n \geq 1$). *Adjacency Matrix* G merupakan matriks yang berukuran n x n dengan elemen $G = [g_{ij}]$ dengan nilai

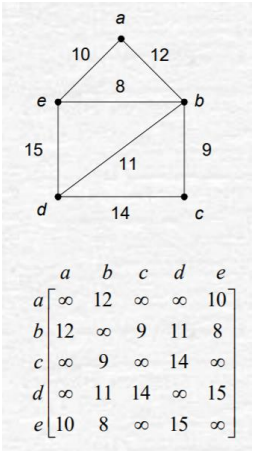
$$g_{ij} = \begin{cases} 0, jika\ simpul\ i\ dan\ j\ tidak\ bertetangga \\ 1, jika\ simpul\ i\ dan\ j\ bertetangga \end{cases}$$

Selain dengan angka 0 dan 1, elemen g_{ij} juga dapat diisi dengan jumlah sisi yang menghubungkan (v_i, v_j) jika graf yang direpresentasikan adalah graf ganda. Untuk merepresentasikan graf semu, pada simpul v_i diberikan nilai 1 untuk posisi (i, i) pada matriks ketetanggaannya.



Gambar 7. Representasi Graf dengan *Adjacency Matrix*
(Sumber: http://www.geeksforgeeks.org/graph-and-its-representations/)

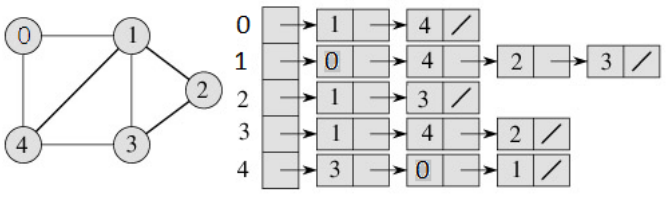
Untuk graf berbobot, tiap elemen g_{ij} merepresentasikan bobot tiap sisi yang menghubungkan simpul i dengan simpul j.



Gambar 8. Representasi Graf Berbobot dengan *Adjacency Matrix*
(Sumber: [3])

2. *Adjacency List*

Untuk menghemat memori pada komputer, list dapat menjadi alternatif representasi karena dalam list, kita tidak perlu menyimpan nilai yang tidak ada (seperti 0 atau ∞). *Adjacency List* menyimpan nama/identitas simpul ke dalam array yang tiap elemen arraynya berupa list yang berisi simpul-simpul yang berhubungan dengan simpul asalnya.



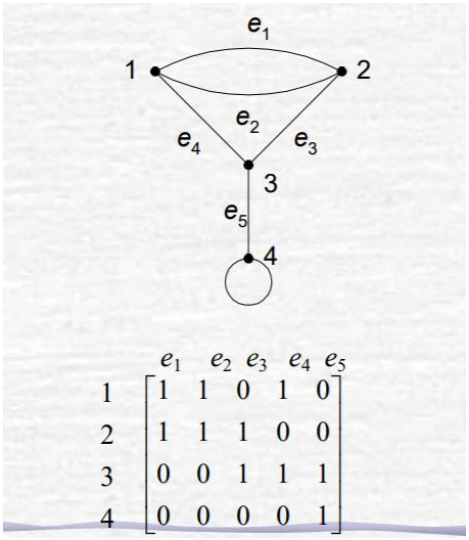
Gambar 9. Representasi Graf dengan *Adjacency List*
(Sumber: http://www.geeksforgeeks.org/graph-and-its-representations/)

3. *Incidency Matrix*

Dalam graf, istilah *Incident* berarti untuk sembarang sisi $e = (v_i, v_j)$, sisi e dikatakan bersisian dengan simpul v_i dan v_j.

Incidency Matrix merepresentasikan kebersisian suatu simpul dengan suatu sisi. Dengan kata lain, *Incidency Matrix* dapat menunjukkan apakah suatu simpul menjadi bagian dari suatu sisi. Misal $G = (V, E)$ adalah graf yang mempunyai n buah simpul $(n \geq 1)$ dan m buah sisi. *Adjacency Matrix* G merupakan matriks yang berukuran n x m dengan elemen $G = [g_{ij}]$ dengan nilai

$$g_{ij} \begin{cases} 0, & \textit{jika simpul i tidak bersisian dengan sisi j} \\ 1, & \textit{jika simpul i bersisian dengan sisi j} \end{cases}$$



Gambar 10. Representasi Graf dengan *Incidency Matrix*
(Sumber: [3])

4. *Incidency List*

Seperti penggunaan list pada *Adjacency List*, list digunakan untuk menghemat memori komputer yang tidak perlu digunakan karena representasi 0 dapat dihilangkan dari list. *Incidency List* menyimpan nama/identitas simpul ke dalam array yang tiap elemennya berisi list yang berisi simpul-simpul yang bersisian dengan sisinya. Selain itu, *Incidency List* juga menyimpan nama/identitas sisi ke dalam array yang menyimpan list simpul yang terhubung dengan dirinya.

C. Search Engine Optimization (SEO)

Search Engine Optimization (SEO) adalah teknik yang digunakan untuk mengoptimalkan urutan kemunculan website dalam pencarian oleh *search engine*. Usaha yang dilakukan untuk meningkatkan jumlah pengunjung ke halaman yang dimiliki ada dua cara, yaitu:

1. On-The-Page SEO

Usaha dan strategi dilakukan pada kode dalam halaman *website*. Usaha yang dilakukan berkaitan dengan HTML (judul, deskripsi, struktur, *header*, warna, pemilihan kata pada website), arsitektur *website* (apakah dapat di-'crawl' dengan mudah oleh *search engine*, apakah dapat terlihat jelas dalam berbagai *platform*, apakah URL mengandung kata yang dicari, apakah koneksi ke website yang disediakan aman), dan konten (kualitas konten, konten terbaru, apakah konten dapat menjawab

pertanyaan dan *query* dari pengguna, riset atas konten, dsb).

2. Off-The-Page SEO
   Usaha yang dilakukan pada bagian ini di luar konteks teknis yang ada pada kode program yang memunculkan website. Faktor tersebut adalah *Trust* (Kepercayaan pengguna terhadap website, apakah link yang ada pada halaman membuat website terpercaya), *Link* (apakah pranala yang ada pada website berasal dari website yang terpercaya, berapa banyak website yang merujuk pada website Anda), *Personal* (letak geografis pengunjung, apakah pengunjung merekomendasikan website Anda), dan *Social* (apakah orang-orang membagikan pranala ke website Anda).

IV. ALGORITMA PAGERANK

PageRank adalah algoritma yang dikembangkan oleh Lawrence Page dan Sergey Brin pada tahun 1998. Algoritma ini digunakan untuk menentukan tingkat kepentingan suatu halaman *website* terhadap halaman *website* yang lain. Apbila kita ingin melihat nilai PR (PageRank) dari suatu website, kita dapat memeriksanya dengan Google Toolbar (http://toolbar.google.com/). Skala yang ditunjukkan oleh toolbar hanya bernilai mulai dari 0 hingga 10, dengan angka tertinggi menunjukkan tingkat kepentingan tertinggi, meskipun angka sebenarnya bisa menjadi jutaan. Nilai yang muncul di toolbar mengikuti suatu skala sebagai berikut

Toolbar Page Rank (Hasil log berbasis 10)	Real Page Rank
0	0 – 10
1	10 – 100
2	100 – 1000
3	1000 – 10000
4	10000 – 100000
dst	dst

Tabel 1. Penskalaan nilai pada PR Toolbar dengan aslinya.
(Sumber: [5])

Nilai dari PR ini akan mempengaruhi urutan dari kemuculan suatu *website* dalam suatu pencarian menggunakan *search engine*.

Sistem yang digunakan oleh PageRank bersifat seperti "voting", di mana suatu *website* akan mendapat tambahan poin jika ada *website* lain yang memberikan link ke *website* tersebut. Jika tidak ada website lain yang merujuk kepada halaman tersebut, tidak terjadi pengurangan poin. Perlu diperhatikan juga perhitungan dilakukan per halaman, bukan per *domain*.

Untuk melakukan perhitungan dengan algoritma PageRank, terlebih dahulu kita harus merepresentasikan hubungan antar halaman *website* dengan graf. Representasi yang cocok adalah menggunakan representasi graf berarah dengan *adjacency matrix* sebagai representasi pada memori. Graf berarah dipilih sebagai representasi *website* karena pada nyatanya, sebuah link pada satu halaman hanya memindahkan pengguna ke satu arah yang berbeda dengan link yang ada pada halaman lain, sehingga representasi graf yang cocok adalah graf berarah. A. Representasi *adjacency matrix* dipilih karena hubungan yang kita lihat adalah hubungan antara simpul dan simpul yang lain, sehingga representasi *adjacency* lebih baik digunakan. Untuk alternatif, *adjacency list* dpaat juga digunakan.

A. Rumusan Algoritma PageRank

Misal halaman *website* A mempunyai halaman T1 … Tn yang merujuk ke halaman A, maka nilai PageRank untuk halaman A adalah

$$PR(A) = (1-d) + d\left(\frac{PR(T1)}{C(T1)} + \cdots + \frac{PR(T1)}{C(T1)}\right) \qquad (3.1)$$

PR(Tn) = Nilai PageRank dari halaman Tn.
C(Tn) = Jumlah link yang keluar dari halaman Tn.
d = *Damping Factor*.

B. Penghitungan Nilai Pagerank

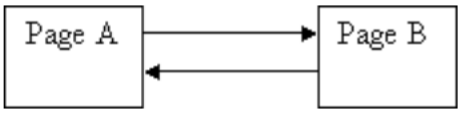
Hasil perhitungan dari PageRank ini akan membentuk distribusi probabilitas, sehingga rata-rata nilai PageRank untuk seluruh halaman *website* di seluruh dunia adalah 1.

Untuk menghitung nilai PageRank, kita tidak perlu mengetahui nilai PageRank awalnya. Menurut Paper yang dikeluarkan oleh Google,

> *"PageRank or PR(A) can be calculated using a simple iterative algorithm, and corresponds to the principal eigenvector of the normalized link matrix of the web."*

Yang berarti kita hanya perlu melakukan pengulangan sampai kita menemukan nilai rata-rata (*normalized sum*) semua nilai websitenya menjadi 1.

Untuk membuktikannya, kita dapat melakukan perhitungan dengan memodelkan suatu hubungan halaman *website* yang saling menrujuk ke halaman selain dirinya dengan graf seperti gambar berikut



Gambar 11. Contoh Graf yang Memodelkan Hubungan Halaman Website
(Sumber: [5])

Misalkan kedua halaman tersebut mempunyai PageRank PR(A) dan PR(B). Kita akan menghitung nilai PageRank kedua website dengan nilai awal PR(A) dan PR(B) yang berbeda-beda hingga kita mencapai nilai rata-rata PR menjadi 1. Dalam kata lain,

$$\overline{PR} = \frac{PR(A) + PR(B)}{2} = 1 \qquad (3.2)$$

Atau persamaan yang lebih umum

$$\overline{PR} = \frac{PR(T1) + \cdots + PR(Tn)}{n} = 1 \quad (3.3)$$

n = Jumlah halaman yang ada
$\overline{PR}$ = Rata-rata nilai PR untuk semua website
PR = Nilai PageRank untuk suatu *website*
T1 = *Website* pertama yang dihitung

Perhitungan pertama kita lakukan dengan nilai awal

$$PR(A) = 40$$
$$PR(B) = 40$$
$$d = 0.85$$

Perhitungan Pertama dengan rumus (3.1)

$$PR(A) = (1 - 0.85) + 0.85\left(\frac{PR(B)}{C(B)}\right)$$

Dari gambar 12, kita dapat mengetahui bahwa $C(A) = 1$ dan $C(B) = 1$, sehingga

$$PR(A) = (0.15) + 0.85\left(\frac{40}{1}\right)$$
$$PR(A) = 34.25$$

$$PR(B) = (0.15) + 0.85\left(\frac{PR(A)}{C(A)}\right)$$
$$PR(B) = (0.15) + 0.85\left(\frac{34.25}{C(A)}\right)$$
$$PR(B) = 29.1775$$

Perhitungan Kedua

$$PR(A) = 0.15 + 0.85\left(\frac{29.1775}{1}\right)$$
$$PR(A) = 24.950875$$

$$PR(B) = 0.15 + 0.85\left(\frac{24.950875}{1}\right)$$
$$PR(B) = 21.35824375$$

Dapat dilihat dari perhitungan yang ada, nilai PageRank setiap pengulangan akan menjadi lebih kecil dari sebelumnya. Hal ini menandakan bahwa nilai-nilai ini akan terus mendekati nilai 1 dan nantinya kita akan mendapatkan nilai 1 sebagai rata-rata.

Untuk membuktikannya, akan dibuat perhitungan yang sama dari awal dengan nilai awal

$$PR(A) = 40$$
$$PR(B) = 40$$
$$d = 0.85$$

Perlu diperhatikan nilai di atas memenuhi persamaan (3.2) dan (3.3).

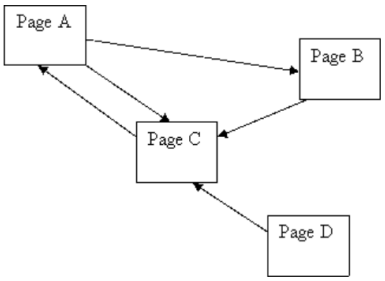
Perhitungan pertama

$$PR(A) = 0.15 + 0.85\left(\frac{1}{1}\right)$$
$$PR(A) = 1$$

$$PR(B) = 0.15 + 0.85\left(\frac{1}{1}\right)$$
$$PR(B) = 1$$

Nilai tersebut tidak berubah dari nilai 1, menandakan pengurangan atau penambahan nilai tidak akan berhenti sampai nilai rata-rata PR sama dengan 1.

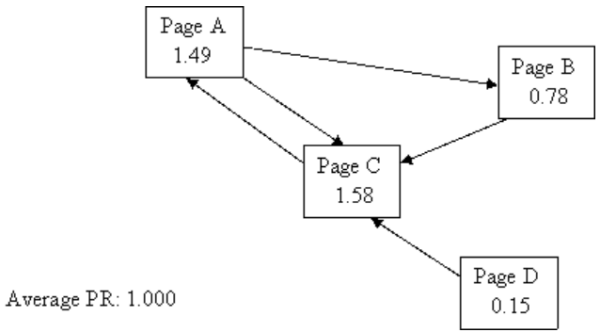
Untuk contoh yang lebih nyata dan akurat, akan digunakan model lain yang mempunyai lebih banyak halaman *website* dan lebih banyak rujukan dari suatu halaman ke halaman yang lain.



Gambar 12. Contoh Graf lain yang Memodelkan Hubungan Halaman Website
(Sumber: [5])

Dalam contoh ini, kita dapat melihat ada 2 simpul yang keluar dari A menuju B ke C ($C(A) = 2$), terdapat masing-masing 1 simpul keluar dari B, C, dan D ($C(B) = C(C) = C(D) = 1$). Selain itu, halaman A menerima 1 rujukkan, halaman B menerima 1 rujukkan, halaman C menerima 2 rujukkan, dan halaman D tidak menerima rujukan dari manapun.

Apabila kita menghitung menggunakan algoritma PageRank dengan nilai awal tiap halaman adalah 0, maka akan didapatkan hasil sebagai berikut



Gambar 13. Hasil Perhitungan PageRank pada Graf Gambar 12
(Sumber: [5])

Hal yang menarik pada hasil ini adalah suatu halaman yang tidak dirujuk oleh halaman manapun masih memiliki nilai 0.15. Hal ini disebabkan adanya nilai $(1 - d)$ dalam persamaan algoritma PageRank yang berfungsi agar perhitungan kita dapat mencapai nilai rata-rata PageRank 1 dari semua *website* yang dimodelkan.

Jika kita telaah lagi perhitungan pada halaman D,

$$PR(D) = (1 - d) + d * (0)$$

$$PR(D) = 0.15$$

dapat dilihat nilai D tidak akan berubah karena tidak ada faktor lain yang akan menambahkan nilai ke halaman D karena tidak ada halaman yang menambahkan referensi ke halaman D, sehingga tidak ada halaman yang dapat menyumbangkan nilai ke halaman D

C. Contoh Implementasi Sederhana PageRank dalam Program C

Untuk menggambarkan cara kerja algoritma lebih baik, akan dibuat suatu program sederhana yang dapat menghitung nilai PageRank jika diberikan gambaran dari jaringan yang terbentuk dalam bentuk graf. Dalam program ini, akan digunakan representasi *adjacency matrix*.

Berikut ini adalah *source code* program tersebut dalam C.

```
/* PROGRAM SIMULASI PAGERANK
   Dibuat oleh:

   Nama: I Kadek Yuda Budipratama Giri
   Tgl : 3 Desember 2017

   Mensimulasikan perhitungan PageRank
   dengan representasi graf
 */

#include <stdio.h>
#include <math.h>
#define EPSILON 0.000001
int n; /* JUMLAH DARI WEBSITE */
int countRow(int i, int n, int G[n][n]){
/* MENGHITUNG LINK RUJUKAN DARI  WEBSITE */
    int a;
    int sum = 0;
    for (a = 0; a<n ; a++){
       sum = G[i][a] + sum;
    }
    return sum;
}

float AvgArray(int n, float H[n]){
/* MENGHITUNG RATA-RATA HASIL PAGERANK */
    float sum = 0;
    int i;
    for (i = 0; i<n ; i++){
       sum = H[i] + sum;
    }
    return (float) sum/n;
}

int main(){
    printf("Berapa Jumlah Website yang akan disimulasikan?\n");
    scanf("%d",&n);
    int Graph[n][n];       //MENYIMPAN REPRESENTASI GRAF
    float Hasil[n];       //MENYIMPAN NILAI PAGERANK TIAP WEBSITE
    printf("Silakan ketikkan representasi graf dalam matriks\n");
    int i, j;
    float d = 0.85;     //NILAI DAMPING FACTOR
    int val;
    /* MENGISI ARRAY DENGAN GRAPH */
    for (i=0; i<n; i++){
       for(j=0; j<n; j++){
          scanf("%d",&Graph[i][j]);
       }
    }
    /* MENGINISIASIKAN NILAI AWAL PAGERANK, YAITU 0 */
    for (i = 0; i<n ; i++){
       Hasil[i] = 0;
    }
    int k;
    /* MELAKUKAN PERHITUNGAN PAGERANK UNTUK TIAP WEBSITE */
    /* MENGIKUTI RUMUS
       PR(A) = (1-d) + d*(PR(T1)/C(T1) + ... + PR(Tn)/C(Tn))
    */
    do{
       for(i = 0; i<n ; i++){
          float temp = 0;
          for(j = 0; j<n ; j++){
             if(Graph[j][i] > 0){
                temp = temp + (Hasil[j]/countRow(j, n, Graph));
             }
          }
          Hasil[i] = (1 - d) + d*temp;
          printf("%d: %.5f ",i,Hasil[i]);
       }
       printf("\n");
       printf("Average: %.5f\n",AvgArray(n,Hasil));
    }while(fabs( AvgArray(n,Hasil) - 1 ) >= EPSILON);
    /* PENGULANGAN DILAKUKAN SAMPAI RATA-RATANYA 1 */
    return 0;
 }
```

Untuk membuktikan kebenaran program, akan disimulasikan perhitungan untuk model *website* pada Gambar 12. Jika direpresentasikan dengan *adjacency matrix*, maka matriksnya akan menjadi sebagai berikut

$$\begin{bmatrix} 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

Tiap halaman A, B, C, D dienumerasi menjadi 0, 1, 2, 3 sebagai identitas simpul pada matriks di atas. Setelah itu, program akan dijalankan sesuai data yang ada pada matriks di atas.

```
C:\Users\USER\Desktop>PageRankSim
Berapa Jumlah Website yang akan disimulasikan?
4
Silakan ketikkan representasi graf dalam matriks
0 1 1 0
0 0 1 0
1 0 0 0
0 0 1 0
0: 0.15000 1: 0.21375 2: 0.39544 3: 0.15000
Average: 0.22730
0: 0.48612 1: 0.35660 2: 0.78721 3: 0.15000
Average: 0.44498
0: 0.81913 1: 0.49813 2: 1.04904 3: 0.15000
Average: 0.62908
0: 1.04169 1: 0.59272 2: 1.22403 3: 0.15000
Average: 0.75211
0: 1.19042 1: 0.65593 2: 1.34097 3: 0.15000
Average: 0.83433
0: 1.28982 1: 0.69818 2: 1.41912 3: 0.15000
Average: 0.88928
0: 1.35626 1: 0.72641 2: 1.47136 3: 0.15000
Average: 0.92600
0: 1.40065 1: 0.74528 2: 1.50626 3: 0.15000
Average: 0.95055
0: 1.43032 1: 0.75789 2: 1.52959 3: 0.15000
Average: 0.96695
0: 1.45015 1: 0.76632 2: 1.54518 3: 0.15000
Average: 0.97791
0: 1.46341 1: 0.77195 2: 1.55560 3: 0.15000
Average: 0.98524
0: 1.47226 1: 0.77571 2: 1.56257 3: 0.15000
Average: 0.99013
0: 1.47818 1: 0.77823 2: 1.56722 3: 0.15000
Average: 0.99341
0: 1.48214 1: 0.77991 2: 1.57033 3: 0.15000
Average: 0.99559
0: 1.48478 1: 0.78103 2: 1.57241 3: 0.15000
Average: 0.99706
0: 1.48655 1: 0.78178 2: 1.57380 3: 0.15000
Average: 0.99803
0: 1.48773 1: 0.78228 2: 1.57473 3: 0.15000
Average: 0.99868
0: 1.48852 1: 0.78262 2: 1.57535 3: 0.15000
Average: 0.99912
0: 1.48904 1: 0.78284 2: 1.57576 3: 0.15000
Average: 0.99941
0: 1.48940 1: 0.78299 2: 1.57604 3: 0.15000
Average: 0.99961
0: 1.48963 1: 0.78309 2: 1.57622 3: 0.15000
Average: 0.99974
0: 1.48979 1: 0.78316 2: 1.57635 3: 0.15000
Average: 0.99982
0: 1.48990 1: 0.78321 2: 1.57643 3: 0.15000
Average: 0.99988
0: 1.48997 1: 0.78324 2: 1.57649 3: 0.15000
Average: 0.99992
0: 1.49001 1: 0.78326 2: 1.57652 3: 0.15000
Average: 0.99995
0: 1.49004 1: 0.78327 2: 1.57655 3: 0.15000
Average: 0.99996
0: 1.49007 1: 0.78328 2: 1.57656 3: 0.15000
Average: 0.99998
0: 1.49008 1: 0.78328 2: 1.57657 3: 0.15000
Average: 0.99998
0: 1.49009 1: 0.78329 2: 1.57658 3: 0.15000
Average: 0.99999
0: 1.49009 1: 0.78329 2: 1.57659 3: 0.15000
Average: 0.99999
0: 1.49010 1: 0.78329 2: 1.57659 3: 0.15000
Average: 1.00000
0: 1.49010 1: 0.78329 2: 1.57659 3: 0.15000
Average: 1.00000
0: 1.49010 1: 0.78329 2: 1.57659 3: 0.15000
Average: 1.00000
0: 1.49010 1: 0.78329 2: 1.57659 3: 0.15000
Average: 1.00000
0: 1.49011 1: 0.78329 2: 1.57660 3: 0.15000
Average: 1.00000
```

Gambar 14.Hasil Program Simulasi Penghitungan PageRank
(Sumber: Dokumen Pribadi)

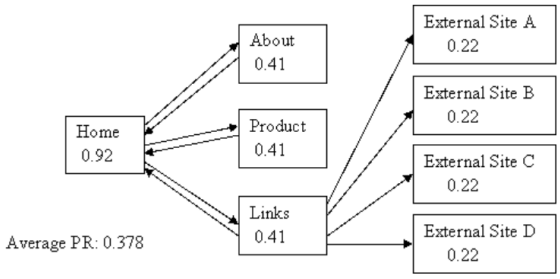
Nilai yang ditunjukkan program dengan hasil sebenarnya pada gambar 13 benar, menndakan program ini dapat mensimulasikan jalannya perhitungan PageRank.

Perlu diperhatikan bahwa program ini hanya bekerja jika tidak ada halaman yang buntu (*dead end*) karena tidak akan tercapai rata-rata nilai PR 1.

VI. OPTIMISASI SEO BERDASARKAN PAGERANK

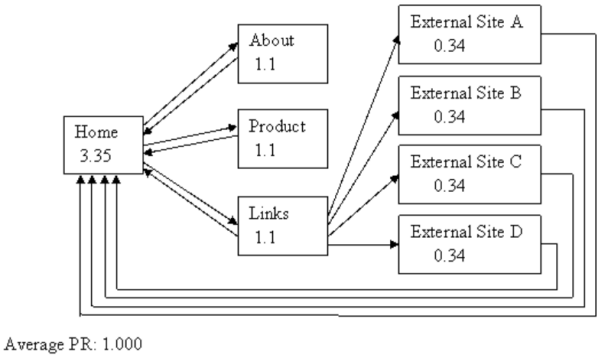
Telah diperlihatkan bagaimana perhitungan nilai PageRank dilakukan. Seperti yang telah dibahas sebelumnya, peringkat dan nilai PageRank ini akan mepengaruhi urutan kemunculan suatu *website* pada pencarian oleh *search engine*. Dengan pengetahuan tentang PageRank, kita akan mencoba mengaplikasikan pengetahuan mengenai PageRank untuk meningkatkan teknik SEO untuk suatu *website*.

Kita akan mengambil kasus ini



Gambar 15. Contoh Kasus 1 sebuah *website*
(Sumber: [5])

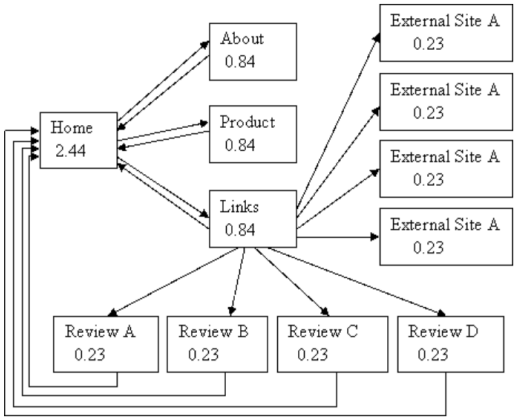
setelah dihitung dengan algoritma PageRank, model graf ini mendapatkan nilai yang terbesar pada bagian Home, tetapi hanya mendapatkan nilai rata-rata sebesar 0.378. Padahal, nilai akhir PR seharusnya menjadi 1. Hal ini disebabkan karena link *external site* tidak merujuk kembali ke halaman home, sehingga terjadi buntu (*dead* end) dan akhirnya membuang kesempatan halaman tersebut untuk "voting" dan menyumbang poin pada halaman lain. Hal ini tentu merugikan karena tidak adanya link rujukan kembali akan membuat nilai PR suatu *website* mengecil. Solusinya adalah merujuk pranala Home dari tiap *external site*.



Gambar 16. Solusi dari Kasus 1
(Sumber: [5])

Setelah kita tambahkan link dari tiap *external site* ke Home, nilai PR dari halaman Home naik drastic dari nilai 0.92 menjadi nilai 3.35. Sementara itu, nilai PR rata-ratanya juga naik menjadi 1. Hal ini menunjukkan bahwa semakin banyak *backlink* (halaman yang merujuk/memberikan link ke halaman kita) yang dimiliki oleh sebuah halaman, semakin tinggi juga nilai PR halaman tersebut.

Contoh solusi lain adalah memberikan *page review* terhadap halaman kita sendiri. Contoh permodelannya sebagai berikut.



Gambar 17. Solusi lain Kasus 1
(Sumber: [5])

Pada solusi ini, dapat dilihat halaman *review* yang dibuat menunjuk kembali ke halaman utama Home, sehingga menghilangkan adanya *dead end* dan pada akhirnya menambahkan nilai PR pada *website*.

Dalam kasus ini, dapat diambil kesimpulan bahwa untuk meningkatkan SEO suatu website, ada hal yang harus dihindari, yaitu adanya halaman *dead end* yang dapat disebabkan kurangnya perujukan atau adanya error pada suatu halaman. Untuk menghilangkan adanya *dead end*, permasalahan/error baik dari internal server atau hilangnya file *website* harus segera diatas untuk menghindari penurunan PageRank. Selain itu, perujukan kembali ke halaman asal harus dilakukan dari link eksternal maupun dibuat halaman review atau dapat juga membuat anak dari suatu website yang menrujuk ke halaman kita. Akan tetapi, metode yang terakhir harus digunakan secara hati-hati karena bisa jadi menyebabkan *spam* yang tidak baik. Hasil dari spam memang cepat dalam menaikkan PR suatu website, akan tetapi, hal yang seperti itu rawan membuat *website* Anda diblok oleh *search engine*, sehingga dampak yang didapat malah negatif.

Selain *backlink*, ada banyak cara untuk meningkatkan SEO Anda, seperti memperbaiki konten, memperbaiki struktur HTML, menyertakan informasi kontak lengkap, memberikan fakta yang benar, originialitas konten, dsb.

V. KESIMPULAN

PageRank merupakan algoritma yang digunakan untuk menghitung ukuran kepentingan dan relevansi suatu halaman terhadap *query* yang dicari *search engine*. Dengan mengetahui sistem PageRank, pemilik *website* dapat mengoptimisasi SEO dari *website* yang dimiliki dengan meningkatkan *backlink* dengan cara yang benar dan legal, seperti memperbaiki konten, merujuk ke situs yang lebih terpercaya, dan sebagainya.

VI. UCAPAN TERIMA KASIH

Penulis ingin mengucapkan rasa syukur kepada Tuhan Yang Maha Esa karena telah membuka jalan untuk hamba-Nya agar dapat menyelesaikan makalah ini dengan baik. Saya juga berterima kasih kepada Dr. Ir. Rinaldi Munir sebagai Dosen Mata Kuliah Matematika Diskrit IF2120 atas didikan dan ajarannya selama saya berkuliah. Saya juga ingin berterima kasih kepada orangtua saya yang telah mendukung saya selama kuliah dan pengerjaan makalah ini. Selain itu, saya juga ingin berterima kasih kepada teman-teman saya dalam membantu saya memberikan inspirasi untuk topic makalah ini.

REFERENSI

[1] Search Engine Histroy.
http://www.searchenginehistory.com/
Diakses 2 Desember 2017 pukul 10.51

[2] Munir, Rinaldi. 2006. Diktat Kuliah IF2120 Matematika Diskrit. Institut Teknologi Bandung : Bandung.

[3] Slide Kuliah Graf IF2120
informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2015-2016/Graf%20(2015).pdf
Diakses 2 Desember pukul 13.26

[4] SEO Guide: Types of Search Engine Ranking Factors
https://searchengineland.com/guide/seo/types-of-search-engine-ranking-factors
Diakses 2 Desember pukul 18.29

[5] PageRank Explained Correctly with Example
http://www.cs.princeton.edu/~chazelle/courses/BIB/pagerank.htm
Diakses 2 Desember pukul 18.36

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, 3 Desember 2017

TTD

I Kadek Yuda Budipratama Giri
13516115