

Universidad Central de Venezuela

Facultad de Ciencias

Escuela de Computación

Asignatura: Cálculo Científico (6105)

Estudiante: Naranjo Sthory Alexanyer Antonio

Cédula de identidad: V – 26.498.600

Tarea 6: El autovector de 458 billones de dólares (hasta ahora)

Para el desarrollo de la solución al problema planteado en el enunciado del ejercicio, se optó por investigar, trabajar e implementar el algoritmo **PageRank** desarrollado por los fundadores de Google, *Larry Page* y *Serguéi Brin*. Como siempre, se adjunta además del presente informe, una carpeta con el nombre **Codes** donde se almacenan los códigos fuentes escritos bajo el lenguaje de programación Matlab/Octave y una carpeta con el nombre **Images** con las imágenes de las gráficas que se hayan generado para sustentar lo explicado en cada sección del actual documento.

Algoritmo PageRank: ¿Qué es?

Una de las razones por las que *Google TM* es un motor de búsqueda tan eficaz es el algoritmo *PageRank TM* desarrollado por los fundadores de Google, *Larry Page* y *Serguéi Brin*, cuando eran estudiantes de posgrado en la Universidad de Stanford. El PageRank se determina enteramente por la estructura de enlaces de la **World Wide Web** (www). Se vuelve a calcular una vez al mes y no tiene en cuenta el contenido real de las páginas web ni las o consultas individuales. Entonces, para cualquier consulta concreta, Google encuentra las páginas de la que coinciden con esa consulta y las enumera en el orden de su PageRank.

Imagínese que navega por la web y pasa de una página a otra eligiendo al **azar** un enlace saliente de una página para llegar a la siguiente. Esto puede llevar a callejones sin salida en páginas sin enlaces salientes, o ciclos alrededor de

camarillas de páginas interconectadas. Así que, una cierta fracción de las veces, simplemente se elige una página al azar de la Web. Este paseo aleatorio teórico se conoce como **cadena de Markov** o **proceso de Markov**. La probabilidad límite de que un navegante aleatorio infinitamente dedicado visite una página es su PageRank. Una página tiene un alto rango si otras páginas con alto rango la enlazan a ella.

Algoritmo PageRank: Análisis Teórico

Sea W el conjunto de páginas web a las que se puede llegar siguiendo una cadena de hipervínculos que comienza en una página raíz, y que n es el número de páginas en W . Para Google, el conjunto W varía en realidad con el tiempo, pero en junio de 2004, n superaba los 4.000 millones.

Sea G la matriz de conectividad $n \times n$ de una parte de la web, es decir, $g_{ij} = 1$ si hay un hipervínculo a la página i desde la página j y $g_{ij} = 0$ en caso contrario. La matriz G puede ser enorme, pero es muy dispersa. Su j -enésima columna muestra los enlaces de la j -enésima página. El número de **nonzeros** (valor positivo o negativo, no igual a cero) en G es el número total de hipervínculos en W .

Sean r_i y c_j , las sumas de las filas y columnas de G :

$$r_i = \sum_j g_{ij}; c_j = \sum_i g_{ij}.$$

Las cantidades r_j y c_j son el grado de entrada y el grado de salida de la página j .

Sea p la probabilidad de que el paseo aleatorio siga un enlace. Un valor típico es $p = 0.85$. Entonces $1 - p$ es la probabilidad de que se elija una página arbitraria y $\delta = (1 - p)/n$ es la probabilidad de que se elija una página aleatoria concreta.

Sea A la matriz de $n \times n$ cuyos elementos son,

$$a_{ij} = \begin{cases} \frac{pg_{ij}}{c_j} + \delta; c_j \neq 0 \\ \frac{1}{n}; c_j = 0. \end{cases}$$

Obsérvese que A proviene de escalar la matriz de conectividad por sus sumas de columna. La j -enésima columna es la probabilidad de saltar de la j – enésima página a las demás páginas de la web. Si la j – enésima página es un callejón sin salida, es decir, no tiene enlaces de salida, entonces asignamos una probabilidad uniforme de $1/n$ a todos los elementos de su columna. La mayoría de los elementos de A son iguales a δ , la probabilidad de saltar de una página a otra sin seguir un enlace.

Si $n = 4 * 10^9$ y $p = 0.85$, entonces $\delta = 3.75 * 10^{-11}$.

La matriz A es la matriz de probabilidad de transición de la cadena de Markov. Sus elementos están todos estrictamente entre cero y uno y las sumas de sus columnas son todas iguales a uno. Un importante resultado de la teoría de matrices conocido como *teorema de Perron-Frobenius* se aplica a estas matrices. Concluye que una solución no nula de la ecuación,

$$x = Ax$$

Existe y es único dentro de un factor de escala. Si este factor de escala se elige de forma que,

$$\sum_i x_i = 1,$$

Entonces x es el vector de estado de la cadena de Markov y es el PageRank de Google. Los elementos de son todos positivos y menores que uno.

Además, el vector x es la solución del sistema lineal singular y homogéneo,

$$(I - A)x = 0.$$

Algoritmo PageRank: Análisis Práctico (Calculo para hallar x)

Para un n modesto, una forma fácil de calcular x en Matlab/Octave es empezar con alguna solución aproximada, como los PageRanks del mes anterior, o,

$$x = \text{ones}(n, 1)/n;$$

A continuación, basta con repetir la declaración de la asignación,

$$x = A * x;$$

Hasta que los vectores sucesivos coincidan dentro de una tolerancia determinada. Esto se conoce como el **método de la potencia** y es casi el único enfoque posible para n muy grandes.

En la práctica, las matrices G y A nunca se forman realmente. Un paso del método de potencia se haría mediante una pasada por una base de datos de páginas web, actualizando los recuentos de referencia ponderados generados por los hipervínculos entre páginas.

La mejor manera de calcular el PageRank en Matlab/Octave es aprovechar la estructura particular de la matriz de Markov. Este es un enfoque que preserva la dispersión de G . La matriz de transición puede escribirse como sigue,

$$A = pGD + ez^T$$

Donde D es la matriz diagonal formada a partir de los recíprocos de los grados exteriores,

$$d_{jj} = \begin{cases} 1/c_j; & c_j \neq 0 \\ 0; & c_j = 0 \end{cases}$$

Donde además, e es el n -vector de todos los unos, y z es el vector con componentes,

$$z_j = \begin{cases} \delta; & c_j \neq 0 \\ 1/n; & c_j = 0. \end{cases}$$

La matriz de rango uno, ez^T , da cuenta de las elecciones aleatorias de las páginas web que no siguen los enlaces.

La ecuación,

$$x = Ax,$$

Puede ser reescrita de la siguiente manera,

$$(I - pGD)x = \gamma e$$

Donde,

$$\gamma = z^T x.$$

No conocemos el valor de γ porque depende del vector desconocido x , pero podemos tomar temporalmente $\gamma = 1$.

Mientras p sea estrictamente menor que uno, la matriz de coeficientes $I - pGD$ es no singular y la ecuación,

$$(I - pGD)x = e$$

Puede resolverse para x . Entonces, la x resultante puede reescalarsse de forma que,

$$\sum_i x_i = 1.$$

Obsérvese que el vector z no interviene realmente en este cálculo.

Algoritmo PageRank: Análisis Práctico (Métodos alternativos)

Lo último mencionado en la sección anterior puede ser escrito en código Matlab/Octave de la siguiente manera,

```

c = sum(G, 1);
k = find(c ~= 0);
D = sparse(k, k, 1./c(k), n, n);
e = ones(n, 1);
I = speye(n, n);
x = (I - p*G*D)\e;
x = x/sum(x);

```

El *método de la potencia* también puede implementarse de forma que no se produzca la matriz de Markov y así preservar la dispersión. Para ello calculamos,

```

G = p*G*D;
z = ((1-p)*(c ~= 0) + (c == 0))/n;

```

Iniciamos con,

```

x = e/n;

```

Entonces, se repite la siguiente sentencia,

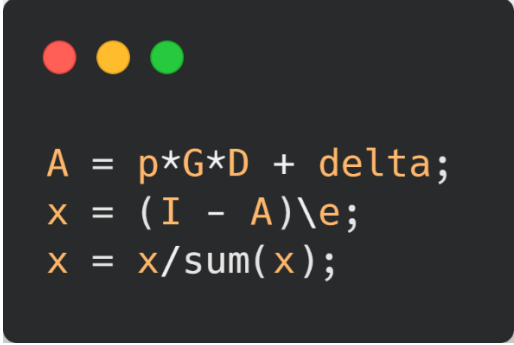
```

x = G*x + e*(z*x)

```

Hasta que x se establezca con varios decimales.

También es posible utilizar un algoritmo conocido como *iteración inversa*.



```
A = p*G*D + delta;  
x = (I - A)\e;  
x = x/sum(x);
```

A primera vista, esto parece una idea muy mala. Como $I - A$ es teóricamente singular, con el cálculo exacto algún elemento diagonal del factor triangular superior de $I - A$ debería ser cero y este cálculo debería fallar. Pero con el error de redondeo, la matriz calculada $I - A$ probablemente no es exactamente singular. Incluso si es singular, el redondeo durante la eliminación gaussiana probablemente impedirá que haya elementos diagonales exactamente nulos.


Sabemos que la eliminación gaussiana con pivoteo parcial siempre produce una solución con un residuo pequeño, en relación con la solución calculada, incluso si la matriz está mal condicionada. El vector que se obtiene con la operación de barra invertida, $(I - A) \backslash N$ suele tener componentes muy grandes. Si se reescala por su suma, el residuo se escala por el mismo factor y se vuelve muy pequeño. En consecuencia, los dos vectores x y $A * x$ son iguales entre sí con un error de redondeo. En esta situación, la resolución del sistema singular con la eliminación gaussiana explota, pero lo hace exactamente en la dirección correcta.

Algoritmo PageRank: Código Fuente

El código fuente se ha estructurado en diferentes módulos en el que, dentro de cada uno se ha realizado la implementación de los diferentes métodos mencionados en la sección anterior. Además, se ha tomado como punto de partida para la ejecución de la aplicación el archivo cuyo nombre es **main.m**, donde también se realiza la definición del caso de prueba utilizado para colocar a la práctica la teoría explicada en secciones anteriores y también la invocación de los diferentes métodos ya mencionados.


A continuación, se adjunta una imagen del código fuente definido en cada archivo a entregar para el desarrollo del algoritmo *PageRank*.

- *power_method.m*




```
function x = power_method(p, c, n, G, D, e)
    z = ((1-p)*(c ~= 0) + (c == 0))/n;
    A = p*G*D + e*z;
    x = e/n;
    oldx = zeros(n,1);
    while norm(x - oldx) > .01
        oldx = x;
        x = A*x;
    endwhile
    x = x/sum(x)
endfunction
```


- *sparse_power_method.m*



```
function x = sparse_power_method(p, G, D, e, n, c)
    z = ((1-p)*(c ~= 0) + (c == 0))/n;
    G = p*G*D;
    x = e/n;
    oldx = zeros(n,1);
    while norm(x - oldx) > .01
        oldx = x;
        x = G*x + e*(z*x);
    endwhile
    x = x/sum(x)
endfunction
```

- *inverse_iteration.m*



```
function x = inverse_iteration(I, G, D, e, p, c, n)
    z = ((1-p)*(c ~= 0) + (c == 0))/n;
    A = A = p*G*D + e*z;
    x = (I - A)\e;
    x = x/sum(x)
endfunction
```

- *main.m*

```
function main
    % Definition Case
    n      = 8;
    i      = [2 3 5 4 8 1 3 8 7 1 5 6 7 1 3 6 1];
    j      = [1 1 1 1 1 2 2 3 3 4 4 5 6 6 7 7 8];
    G      = sparse(i, j, 1, n, n);
    spy(G);

    % PageRank
    p      = 0.85;
    delta  = (1-p)/n;
    c      = sum(G, 1);
    k      = find(c ~= 0);
    D      = sparse(k, k, 1./c(k), n, n);
    e      = ones(n, 1);
    I      = speye(n, n);
    x      = (I - p*G*D)\e;
    x      = x/sum(x)

    % Conventional power method
    %x      = power_method(p, c, n, G, D, e);

    % Sparse power method
    %x      = sparse_power_method(p, G, D, e, n, c);

    % Inverse iteration
    %x      = inverse_iteration(I, G, D, e, p, c, n);

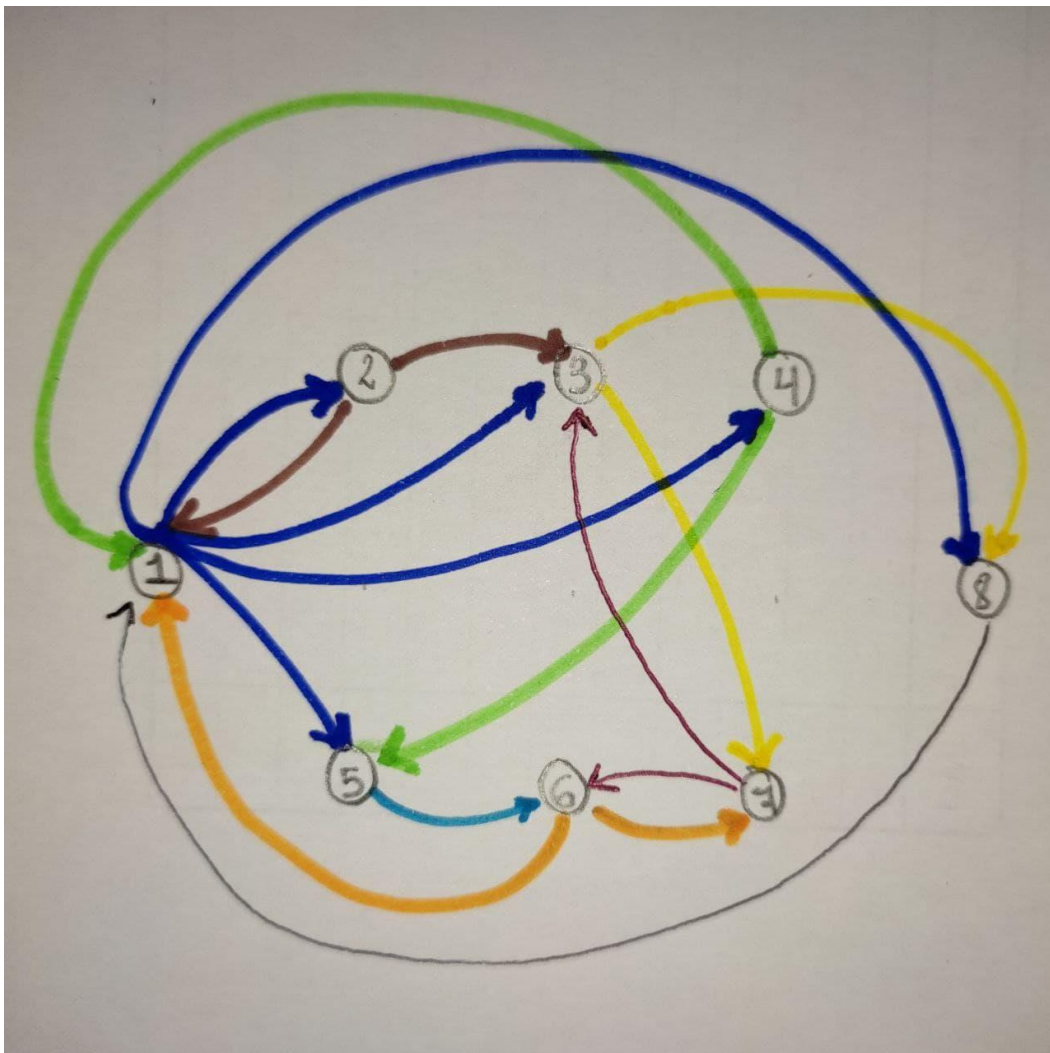
    % Bar graph
    bar(x)
    title('Page Rank')
endfunction

main();
```

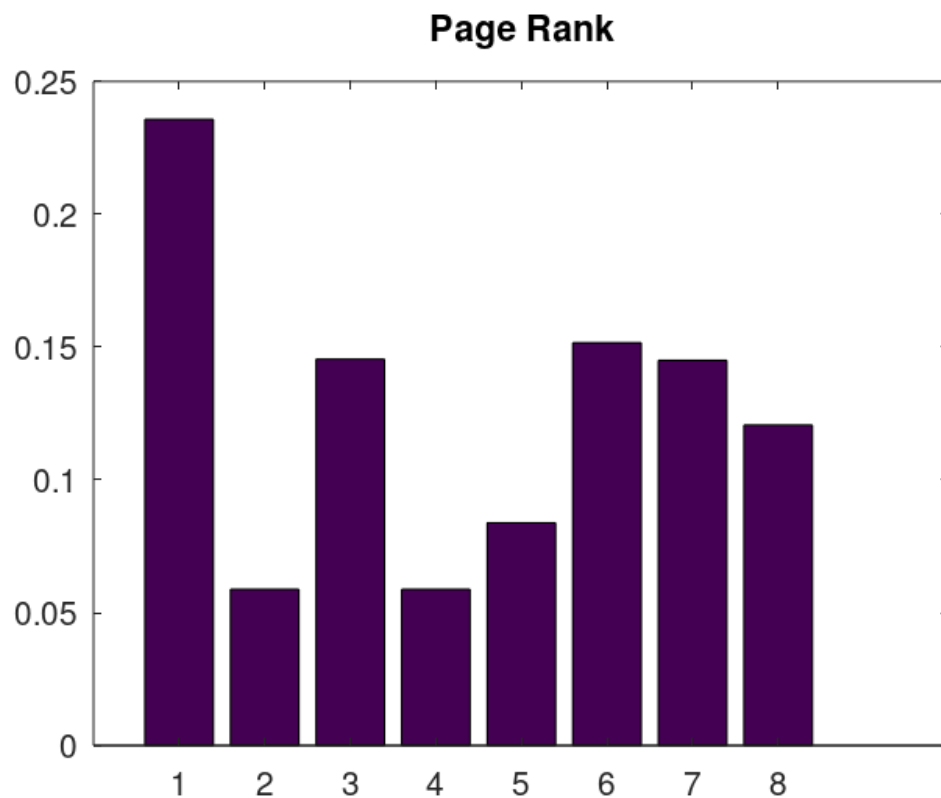
Algoritmo PageRank: Caso de Prueba y Resultados

Por último, se ha definido el siguiente caso de prueba donde colocamos ocho nodos de ejemplo con sus correspondientes enlaces entre ellos. Recordemos que para el algoritmo que nos encontramos trabajando actualmente, cada nodo (vértice, si nos enfocamos en la terminología de teoría de grafos), representa una página web, mientras que cada arista o arco, representa un enlace de la página i a la j , es decir, podemos encontrarnos con un enlace que nos permitirá navegar de una página a otra partiendo desde una página en determinada.

A continuación, se adjunta una imagen del grafo a ser estudiado y así tener una representación visual de cada página definida con sus correspondientes conjuntos de enlaces.



Si aplicamos los métodos implementados en el código fuente, éste nos dará como resultado los siguientes PageRank.



A partir de la gráfica anterior, podremos organizar cada página web según su PageRank obtenido y así conocer la importancia de cada una ya teniendo a nuestro conocimiento estos valores.

- 1) *Página #1: 0.235751*
- 2) *Página #6: 0.151637*
- 3) *Página #3: 0.145462*
- 4) *Página #7: 0.145017*
- 5) *Página #8: 0.120649*
- 6) *Página #5: 0.083829*
- 7) *Página #2: 0.058828*
- 8) *Página #4: 0.058828*

Algoritmo PageRank: Conclusiones

El algoritmo PageRank se desprende de la idea de que se puede juzgar la importancia de una página web mirando las páginas que contienen un enlace hacia la misma. Si una página A contiene un vínculo hacia otra página B se interpreta que la página A considera que el contenido de B es relevante para la temática abordada en A. Si existen muchas páginas con enlaces hacia B se considera que es de común acuerdo que la página B es importante. Por otro lado, si la página B tiene solamente un enlace, pero este proviene de una página C con **autoridad**, decimos que C transfiere su autoridad a B, es decir, indica que B es importante. Utilizando estos conceptos de importancia y autoridad el algoritmo PageRank asigna un rango a cada página basándose en las páginas que dirigen a ellas. Aunque, para el presente trabajo, no se ha considerado la autoridad de una página web, sino que simplemente se consideran que todas tienen un mismo nivel en este aspecto.

La gran extensión de la web lleva a utilizar algoritmos cuya complejidad excede el alcance de este trabajo. Sin embargo, se puede alcanzar una comprensión de los rudimentos del funcionamiento de estos algoritmos partiendo de una versión acotada de la web.

Si se parte de una representación de la web mediante una red de grafos fuertemente conectados se podrá representar la web con una matriz estocástica.

Es de relevancia mencionar que se ha demostrado una aplicación de las *Cadenas de Markov* en la cual el fenómeno aleatorio abordado no se desarrolla en el tiempo, como es lo habitual con este tipo de procesos. En este caso los algoritmos de búsqueda se limitan a simular infinitas búsquedas en un instante para llegar al vector de importancia de equilibrio en un brevísimo lapso.

Por último, se menciona la gran motivación y entusiasmo tenido durante el desarrollo de esta asignación dado a que se ha tenido la oportunidad de aplicar lo aprendido en una de las asignaturas que más ha llamado la atención durante la carrera (Probabilidad y Estadística) junto con la actual en curso.