Apellidos, Nombre: Naval Rodríguez, Ernesto
Apellidos, Nombre: Serrano Arrese, Julia

Instrucciones.

- La codificación deberá ser lo más eficaz, eficiente y precisa posible y fácilmente escalable (pensar que lo que codifiquéis para una matriz de tamaño académico, lo tendréis que reescalar a una dimensión muy grande).
- Tener en cuenta las prestaciones numéricas del código (recursos que utiliza, memoria, cpu…).
- Se recomienda utilizar indexación lógica.
- En cada apartado se deberá entregar: el código utilizado, volcado de los datos, las gráficas, contestar a las preguntas e incluir comentarios, según corresponda en cada caso.
- Debéis entregar, según corresponda:
  - Código,
  - Volcado de datos,
  - Gráficas (del comando bar),
  - Responder a las preguntas.
- La presentación de la práctica se valorará especialmente.

1.

1.A. Código para calcular la matriz G. Volcar el contenido de los vectores y matrices.

```
   %A: Calcular la matriz G de Google del grafo
% Nodos de entrada (Origen de las flechas)
i=[1 1 1 2 2 3 3 4 4 6 6 7 7];
% Nodos de salida (Destino de las flechas)
j=[2 4 5 3 7 4 6 2 7 5 7 2 4];
% Dimensión de la matriz
N=7;

% matriz dispersa. Matriz C con 13 1s (cada flecha),
C=sparse(j,i,1,N,N); %C(i,j)=1 si Pj -> Pi
% visualizar matriz completa (con 0s tambien)
full(C);
% Creamos la matriz completa
Ccompleta=full(C);
% Vemos el tamaño que ocupan en memoria las matrices C y Ccompleta
% (C menor espacio ya que no tiene los 0s)
whos;

% calcular vector Nj. No todos los nodos tienen salida. Nodo 5 = 0 links de
% salida
Nj = sum(C);

%calcular vector dj (solo el nodo 5 es un nodo sin salida dj(5) == 1)
dj=zeros(1,N);
dj(find(Nj==0))=1;

% Calcular matriz de transición modificada S
A = C./Nj;
S=C;
```

```
% Convertir en array logico
dj_loj = logical(dj)

% calculamos S = A + 1/N * e * dj
S(:,dj_loj)=1/N
S(:,~dj_loj)=S(:,~dj_loj)./Nj(~dj_loj);


%calcular matriz G con param alfa=0.85
G = getG(S,0.85)
% suma de elementos de sus columnas == 1 --> matriz estocástica
sum_G = sum(G)
```

Contenido de vectores y matrices:

```
C =

    (2,1)        1
    (4,1)        1
    (5,1)        1
    (3,2)        1
    (7,2)        1
    (4,3)        1
    (6,3)        1
    (2,4)        1
    (7,4)        1
    (5,6)        1
    (7,6)        1
    (2,7)        1
    (4,7)        1
```

```
Ccompleta =

     0     0     0     0     0     0     0
     1     0     0     1     0     0     1
     0     1     0     0     0     0     0
     1     0     1     0     0     0     1
     1     0     0     0     0     1     0
     0     0     1     0     0     0     0
     0     1     0     1     0     1     0
```

```
  Name           Size              Bytes  Class     Attributes

  C              7x7                 272  double    sparse
  Ccompleta      7x7                 392  double
```

```
Nj =

    (1,1)         3
    (1,2)         2
    (1,3)         2
    (1,4)         2
    (1,6)         2
    (1,7)         2
```

```
dj =

     0     0     0     0     1     0     0
```

```
dj_loj =

  1×7 logical array

   0   0   0   0   1   0   0

S =

   (2,1)        1.0000
   (4,1)        1.0000
   (5,1)        1.0000
   (3,2)        1.0000
   (7,2)        1.0000
   (4,3)        1.0000
   (6,3)        1.0000
   (2,4)        1.0000
   (7,4)        1.0000
   (1,5)        0.1429
   (2,5)        0.1429
   (3,5)        0.1429
   (4,5)        0.1429
   (5,5)        0.1429
   (6,5)        0.1429
   (7,5)        0.1429
   (5,6)        1.0000
   (7,6)        1.0000
   (2,7)        1.0000
   (4,7)        1.0000
```

```
G =

    0.0214    0.0214    0.0214    0.0214    0.1429    0.0214    0.0214
    0.3048    0.0214    0.0214    0.4464    0.1429    0.0214    0.4464
    0.0214    0.4464    0.0214    0.0214    0.1429    0.0214    0.0214
    0.3048    0.0214    0.4464    0.0214    0.1429    0.0214    0.4464
    0.3048    0.0214    0.0214    0.0214    0.1429    0.4464    0.0214
    0.0214    0.0214    0.4464    0.0214    0.1429    0.0214    0.0214
    0.0214    0.4464    0.0214    0.4464    0.1429    0.4464    0.0214

sum_G =

    1.0000    1.0000    1.0000    1.0000    1.0000    1.0000    1.0000
```

1.B. Comprobar que G verifica las condiciones (de entrada y de salida) del teorema de Perron-Frobenius.

Condiciones de entrada:

```
% Para que la matriz G verifique el teorema de Perron-Frobenius se deben
% cumplir las siguientes condiciones de entrada:

    %a. Matriz irreducible
    %b. Matriz sin elementos negativos
    %c. Suma de elementos de columnas == 1

% suma de elementos de sus columnas == 1 --> matriz estocástica
sum_G = sum(G);
```

Comprobación:

- Como se puede ver en el grafo dibujado, se puede acceder a cualquier nodo del grafo desde cualquier otro nodo del grafo, por lo que se verifica que se trata de una matriz irreducible
- La matriz no tiene ningún elemento negativo y la suma de los elementos de sus columnas es 1, por lo que se verifica las condiciones de: no negativa y estocástica

Condiciones de salida:

```
% Segun el Teorema de Perron-Frobenius: Si S es una matriz irreducible,
% estostática y no negativa se deberan cumplir las siguientes condiciones
% de salida --> mayor autovalor de S = 1 (autovalor dominante)
% y su autovector (autovector dominante) tiene todos sus elementos no negativos

%matriz diagonal D de autovalores y matriz de columnas de autovectores
[V, D] = eig(G);

% Encuentra el autovalor dominante y su índice
[~, index] = max(abs(diag(D)));

% Extrae el autovalor y el autovector dominante
autovalor_dominante = D(index, index);  %1
autovector_dominante = V(:, index);     %autovector con todos los elementos no
negativos
```

1.C. Código de la rutina potencia. Volcar los resultados. Contestar a las preguntas.

```
    %C: Método de la potencia
%metodo potencia
[lambda,x] = potencia(G,50); %lambda y x son los autovalor y autovector obtenido
anteriormente


%utilizando definicion G * x = lambda * x. Se verifica la definicion
G_x = G*x;
lambda_x = lambda * x;

% precisiones obtenidas con metodo potencia
precision1 = norm(G * x - x,2);
precision2 = abs(lambda - 1);
precision = max(precision1,precision2);
```

Resultados:

```
lambda =

     1

x =

    0.0710
    0.5344
    0.2981
    0.4639
    0.1751
    0.1976
    0.5793
```

```
lambda_x =

    0.0710
    0.5344
    0.2981
    0.4639
    0.1751
    0.1976
    0.5793

precision1 =

   1.3878e-16


precision2 =

     0


precision =

   1.3878e-16
```

Preguntas:
¿Son lambda y x el autovalor y autovector dominantes de la matriz G obtenidos antes con el comando eig?.
Si, los lambda y x obtenidos a partir del método de la potencia son los mismos obtenidos con el commando eig

¿Todas las componentes del autovector dominante son del mismo signo?.
Sí, todas las componentes son positivas

1.D. Código para calcular el pagerank del grafo. Volcar los resultados. Contestar a las preguntas.

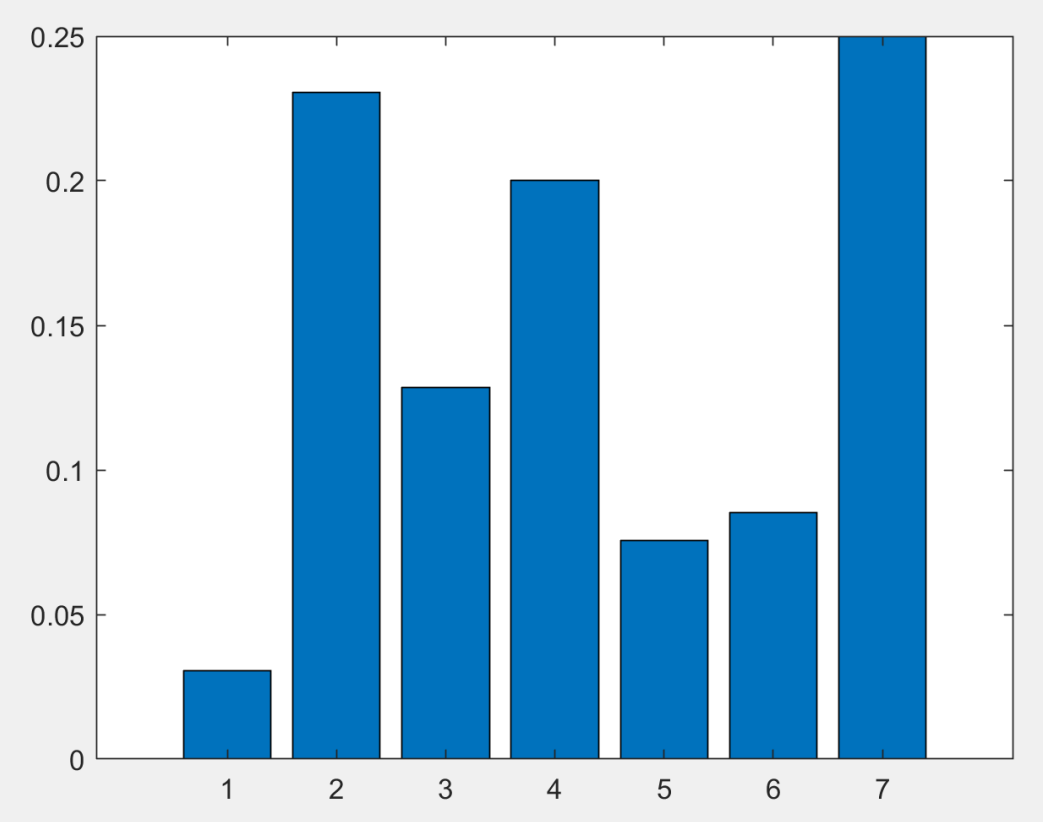
```
        %D: Calcular pagerank del grafo

    pagerank = x/sum(x);
    sum(pagerank);  %=1
```

```
% metodo de la potencia con pagerank como salida, al menos 12 cifras prec.
[lambda,pagerank,precision1,precision2,precision]=potencia_mod(G,50);


orden_pagerank = get_ranking(pagerank');  %P7 > P2 > P4 > P3 > P6 > P5 > P1
bar(pagerank)
```

Visualizar resultados:



Orden:
P7 > P2 > P4 > P3 > P6 > P5 > P1

2. Script para calcular el pagerank del grafo

```
% EJERCICIO 2

i = [3 3 5 7 7 8 11 11];    %nodos de entrada
j = [8 10 11 8 11 9 9 10];  %nodos de salida
N = 11;                      %dimension matriz
alfa = 0.85;
niter = 50;
[pagerank, orden_pagerank] = get_pagerank(N,i,j,alfa,50);
```

Script utilizado:

```
function [pagerank, orden_pagerank] = get_pagerank(N,i,j,alfa,niter)
```

```
% i = [3 3 5 7 7 8 11 11];    %nodos de entrada
% j = [8 10 11 8 11 9 9 10];  %nodos de salida
% N = 11;                      %dimension matriz

C=sparse(j,i,1,N,N);        %matriz dispersa
Nj = sum(C);                %vector Nj
dj=zeros(1,N);
dj(find(Nj==0))=1;          %vector dj

% Calcular matriz de transición modificada S
A = C./Nj;
S=C;

% Convertir en array logico
dj_loj = logical(dj);

% calculamos S = A + 1/N * e * dj
S(:,dj_loj)=1/N;
S(:,~dj_loj)=S(:,~dj_loj)./Nj(~dj_loj);

%calcular matriz G
G = getG(S,alfa);

[lambda,pagerank,precision1,precision2,precision]=potencia_mod(G,niter);

orden_pagerank = get_ranking(pagerank');
bar(pagerank)

end
```

Resultados:

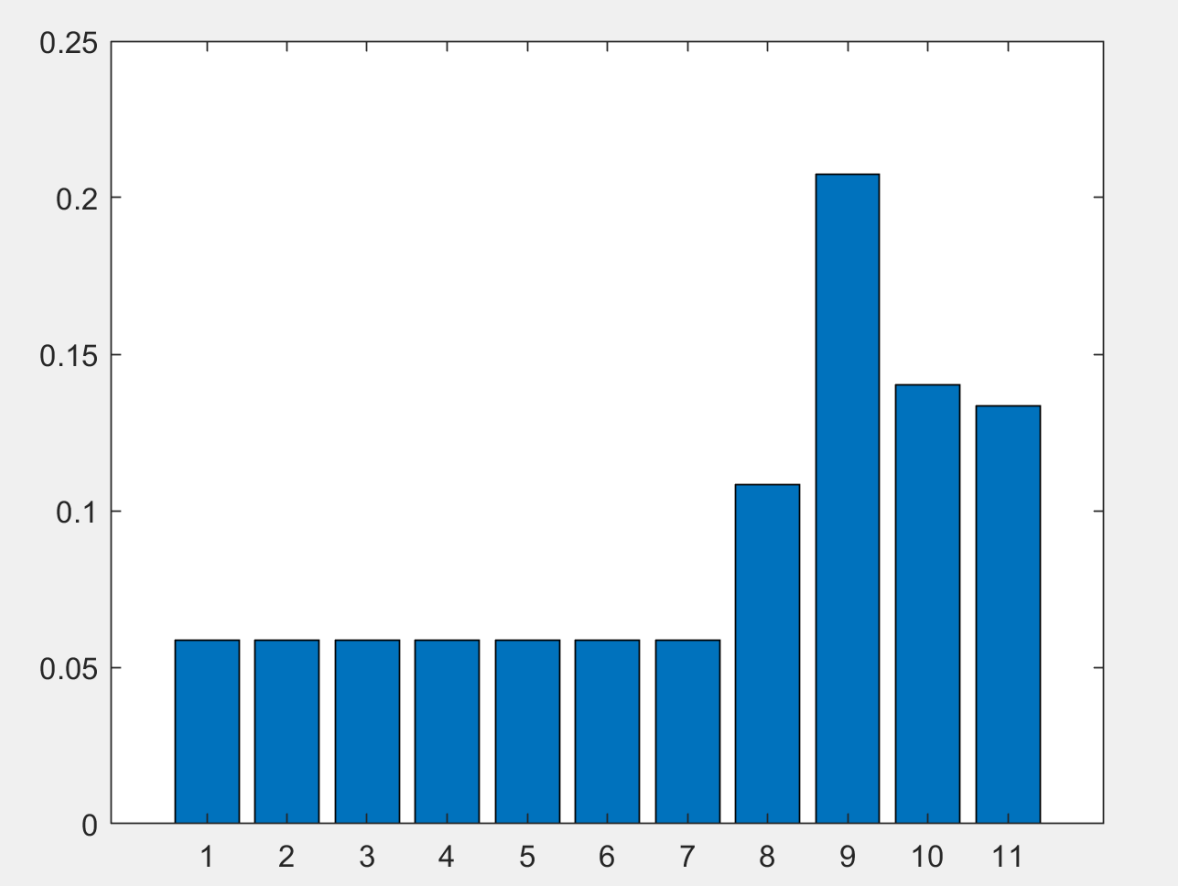
```
pagerank =

    0.0586
    0.0586
    0.0586
    0.0586
    0.0586
    0.0586
    0.0586
    0.1085
    0.2075
    0.1402
    0.1334


orden_pagerank =

     9    10    11     8     1     2     3     4     5     6     7
```

Visualización de resultados:



Orden:
P9 > P10 > P11 > P8 > P1 > P2 > P3 > P4 >P5 > P6 > P7

3. Calculo de prestaciones numéricas. Para cada N indicar la memoria, tiempo de cpu y precisión obtenidos. Hacer una tabla con los resultados.

```
% EJERCICIO 3
% EJERCICIO 3

[ordenpagerank1,precision1, memoria_usada1, tiempo_cpu1] =
CalculoPageRank(1000,20,100);
[ordenpagerank2,precision2, memoria_usada2, tiempo_cpu2] =
CalculoPageRank(10000,20,1000);
[ordenpagerank3,precision3, memoria_usada3, tiempo_cpu3] =
CalculoPageRank(100000,20,10000);
```

Script:

```
function [ordenpagerank, precision, memoria_usada, tiempo_cpu]  =
CalculoPageRank(N,p,niter)
% N = dimension
% p = numero medio de links
% niter = numero de iteraciones

% Iniciar temporizador
tic;

i=randi(N,1,p*N);
j=randi(N,1,p*N);
C=sparse(i,j,1,N,N);
alpha = 0.85;

Nj = sum(C);                %vector Nj
dj=zeros(1,N);
dj(find(Nj==0))=1;          %vector dj

% Calcular matriz de transición modificada S
A = C./Nj;
S=C;

% Convertir en array logico
dj_loj = logical(dj);

% calculamos S = A + 1/N * e * dj
S(:,dj_loj)=1/N;
S(:,~dj_loj)=S(:,~dj_loj)./Nj(~dj_loj);
```

```
%calcular matriz G
G = getG(S,alpha);

% metodo de la potencia con pagerank como salida
[lambda,pagerank,precision1,precision2,precision]=potencia_mod(G,niter);

ordenpagerank = get_ranking(pagerank');

 % Detener el temporizador
 tiempo_cpu = toc;

 % Medir la memoria utilizada
 memoria_info = memory;
 memoria_usada = memoria_info.MemUsedMATLAB / (1024 ^ 2); % Convertir a
megabytes


% Mostrar resultados
 fprintf('N = %d, p = %d, niter = %d\n', N, p, niter);
 fprintf('Tiempo de CPU: %.4f segundos\n', tiempo_cpu);
 fprintf('Memoria utilizada: %.4f MB\n', memoria_usada);
 fprintf('Precision1 obtenida: %.4e\n', precision1);
 fprintf('Precision2 obtenida: %.4e\n', precision2);
 fprintf('Precision obtenida: %.4e\n', precision);
 fprintf('---------------------------------------\n');



end
```

Resultados:

```
N = 1000, p = 20, niter = 100
Tiempo de CPU: 0.0197 segundos
Memoria utilizada: 3313.6250 MB
Precision1 obtenida: 8.2541e-17
Precision2 obtenida: 0.0000e+00
Precision obtenida: 8.2541e-17
---------------------------------------
N = 10000, p = 20, niter = 1000
Tiempo de CPU: 26.1874 segundos
Memoria utilizada: 4084.3789 MB
Precision1 obtenida: 1.7718e-16
Precision2 obtenida: 0.0000e+00
Precision obtenida: 1.7718e-16
---------------------------------------
Error using *
Requested 100000x100000 (74.5GB) array exceeds maximum array size preference (31.7GB). This might cause MATLAB to become unresponsive.
```

N	p	niter	Tiempo CPU	Memoria utilizada	Precisión1	Precisión2	Precisión
1000	20	100	0.0197 segundos	3313.6250 MB	8.2541e-17	0.0000e+00	8.2541e-17
10000	20	100	26.1874 segundos	4084.3789 MB	1.7718e-16	0.0000e+00	1.7718e-16

Scripts utilizados (aparte de los añadidos anteriormente):

```
function indices_ordenados = get_ranking(vector)
    [~, indices_ordenados] = sort(vector, 'descend');
end


function G=getG(S, alfa)
    N=size(S,1);
    e=ones(N,1);
    G=alfa*S+(1-alfa)*(1/N)*(e*e');
return


function [autovalor,x] = potencia(A,niter)
N = length(A);
x1 = ones(N,1);
for k = 1:niter
    x = x1;
    x = x / norm(x);
    x1 = A * x;
end
autovalor = x' * x1;

function [autovalor,pagerank,precision1,precision2,precision] =
potencia_mod(A,niter)
N = length(A);
x1 = ones(N,1);
for k = 1:niter
    x = x1;
    x = x / norm(x);
    x1 = A * x;
end
autovalor = x' * x1;
pagerank = x/sum(x);
precision1 = norm(A * x - x,2);
precision2 = abs(autovalor - 1);
precision = max(precision1,precision2);
```