Practica Pagerank

Dia 4

Miguel Garcia Lafuente

Índice de contenido

Para esta practica se ha usado un ordenador portatil con las siguentes caracteristicas:

```
S.O: ArchLinux 64 bits.
Memoria RAM: 8076456 kB. (unos 7.7GB)
CPU: Intel(R) Core(TM) i5-4200M CPU @ 2.50GHz – 64 bits

~ » archey3

                +               OS: Arch Linux x86_64
                #
               ###              Kernel Release: 3.14.4-1-ARCH
              #####             Uptime: 9:53
              ######            WM: None
             ; #####;           DE: None
            +##.#####           Packages: 1759
           +##########          RAM: 3922 MB / 7887 MB
          ##############;       Processor Type: Intel(R) Core(TM) i5-4200M CPU @ 2.50GHz
         ###############+       $EDITOR: vim
        #######   #######       Root: 322G / 458G (70%) (ext4)
      .######;     ;###;`".
     .#######;     ;#####.
     #########.   .########`
    ######'           '######
   ;####                 ####;
   ##'                     '##
  #'                         `#
```

Version de Julia 0.3.3:

```
~ » julia
rock@neurotiko

               _
   _       _ _(_)_     |  A fresh approach to technical computing
  (_)     | (_) (_)    |  Documentation: http://docs.julialang.org
   _ _   _| |_  __ _   |  Type "help()" for help.
  | | | | | | |/ _` |  |
  | | |_| | | | (_| |  |  Version 0.3.3 (2014-11-23 20:19 UTC)
 _/ |\__'_|_|_|\__'_|  |
|__/                   |  x86_64-unknown-linux-gnu

julia>
```

Como apunte, a pesar de tener 4 cores, Julia esta en estado de crecimiento y por ahora solo funciona en un core, por lo que a efectos practico el procesador es de un solo nucleo a 2.5 GHz.

2. Cargar la matriz.

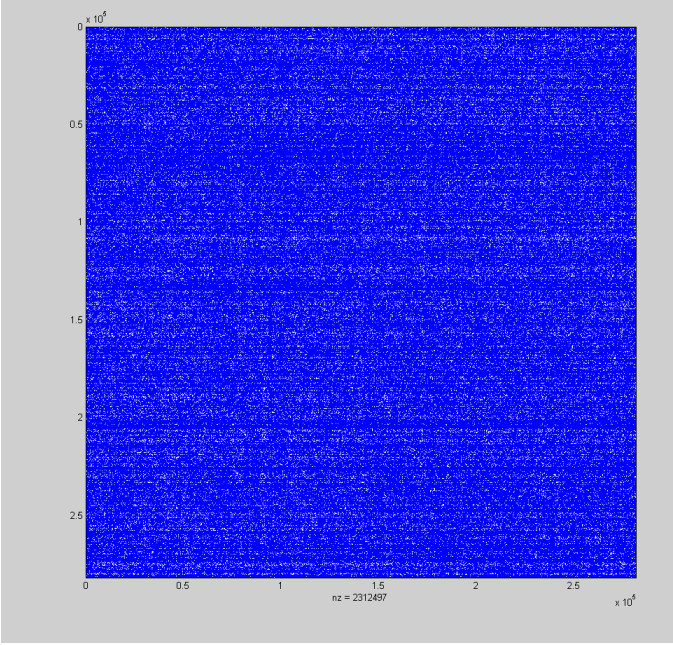
Julia no tiene spconvert, pero aquí esta mi codigo (bastante rapido y eficiente) para spconvert:

```
function spconvert(path::String)
    io = open(path,"r")

    max = 0
    I,J,V = Int64[], Int64[], Float64[]

    for line in EachLine(io)
        h = split(line)
        i,j = map(int64, h[1:2])
        v = float64(h[3])
        push!(I,i)
        push!(J,j)
        push!(V,v)
        ## max = (i > max) ? i : max
        ## max = (j > max) ? j : max
    end
    close(io)
    M::SparseMatrixCSC = sparse(J,I,V)
    ## M::SparseMatrixCSC = sparse(I,J,V,max,max)
    return M
end
```

3. Grafica de dispersion de A



4. Caracteristicas de A.

Este es el codigo que va a sacar toda la informacion para saber las caracteristicas:

```
function test()
    println("Estadisticas cargar datos: ")
    l = @time spconvert("stanford-web.dat")

    println()
    max_e = size(l,1)
    println("Nodos: ", max_e)

    println()
    n_elem = max_e*max_e
    tam = sizesp(l)
    println("Tamaño (B): ", tam)

    _,J,_ = findnz(l)

    println()
    n_no_nulos = length(J)
    n_nulos = n_elem - n_no_nulos
    println("Elementos no nulos: ", n_no_nulos)

    println()
    println("La relacion no_nulos / nulos es: ", n_no_nulos / n_nulos,
            " por lo que es dispersa.")

    println()
    suma = sum(l,1)
    println("Con Suma: ", length(suma))
    println("Todas las sumas dan 1?: ", any(map(x->x==1.0, suma)))
    println("Todos tienen suma, todo da 1, la matriz es S")

    println()
    dispers = n_no_nulos / max_e
    println("Dispersion: ", dispers)

    println()
    n_medio_out = length(J) / max_e
    println("Numero medio outlinks: ", n_medio_out)

    println()
    Is = Set(J)  # Quitamos repetidos. Nodos con salida
    nsalida = length(Is)
    println("Nodos con salida: ", nsalida)
    nsinsalida = max_e - nsalida
    println("Nodos sin salida: ", nsinsalida)

    println()
    relacion = n_elem / max_e ==  nsalida + nsinsalida
    println("Se cumple la relacion?: ", relacion)

    return l
end
```

Estas son los resultados:

```
Estadisticas cargar datos:
elapsed time: 4.835628879 seconds (1683609460 bytes allocated, 9.30% gc time)

Nodos: 281903

Tamaño (B): 6765712

Elementos no nulos: 2312497

La relacion no_nulos / nulos es: 2.910009592235599e-5 por lo que es dispersa.

Con Suma: 281903
Todas las sumas dan 1?: true
Todos tienen suma, todo da 1, la matriz es S

Dispersion: 8.203165627893283

Numero medio outlinks: 8.203165627893283

Nodos con salida: 281731
Nodos sin salida: 172

Se cumple la relacion?: true
5.
```

- Tipo de matriz:

Dispersa

- ¿Cuantos nodos tiene la red?

281903

- Tamaño en memoria de A

6765712 B = 6.45 MB

- Numero de elementos no nulos de A

4624994

- ¿Tipo de matriz?

Todos los nodos tienen suma por columna, y todas las columnas suman 1, por lo que la matriz es S.

- Indice de dispersion

16,4063312

- Numero medio de outlinks

8,203

- ¿Cuantos nodos sin salida hay en la red? ¿Y con salida?

Sin salida: 172

Con salida: 281731

- ¿Se cumple la relacion?

Si, se cumple.

5. pagerank de Stanford Web Matrix

Este es el codigo que lo ejecuta:

```
1 function test2(l::SparseMatrixCSC)
2     println("5.\n")
3     println("Estadisticas calcular PR")
4     x, precision = @time calculo_PR(l, 1e-13)
5     println(precision)
6     println(sizeof(x))
7     return x
8 end
```

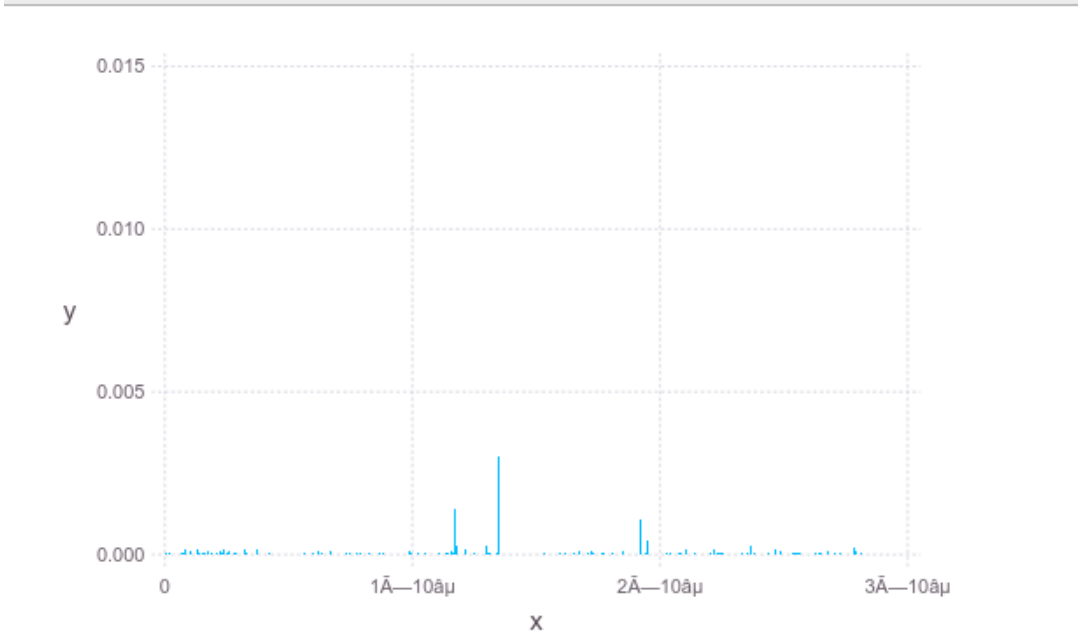
Este el resultado:

```
5.
Estadisticas calcular PR
Iteraciones del algoritmo: 155
elapsed time: 5.65081047 seconds (8650699208 bytes allocated, 22.11% gc time)
7.039717086039987e-12
2255224
```

	Tiempo (s)	Memoria (MB)	Iteraciones:	Precision
N=281x10^3	5,650810	2.1507	155	7.0397170860e-12

6. Visualizar y ordenar los resultados.

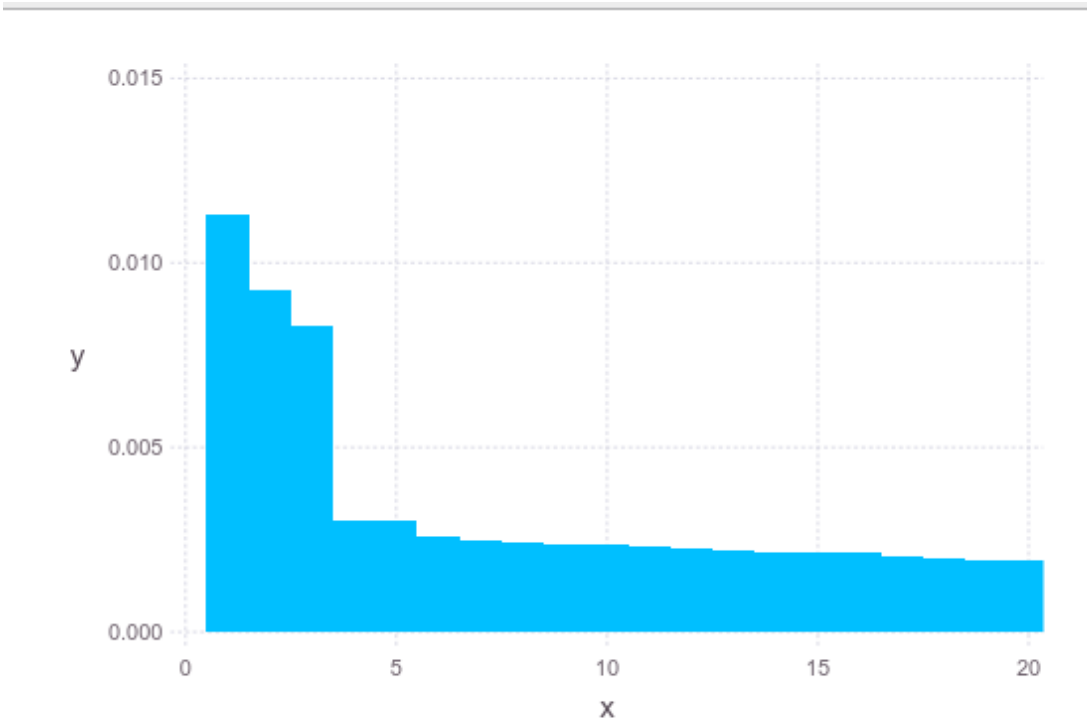
- Pagerank:



- 20 mayores:

Orden	Nodo	Pagerank
1	89073	0.011302835789176058
2	226411	0.009287655679880824
3	241454	0.008297234843864873
4	262860	0.00302311651092151
5	134832	0.003001266413474004
6	234704	0.0025722562437877432
7	136821	0.0024537026510304417
8	68889	0.0024307797066600366
9	105607	0.002397427523884677
10	69358	0.0023640036048740674
11	67756	0.0023009902138876625
12	225872	0.0022674038284480474
13	186750	0.002232431543744009
14	95163	0.0021465381070870483
15	251796	0.0021465381070870444

16	272442	0.0021465381070870444
17	119479	0.0020362393693691906
18	231363	0.0020076908449220823
19	55788	0.0019570367748536575
20	167295	0.0019164905885763845



- Los 20 menores.

Estos son los nodos con su pagerank:

```
Last 20:
(281537,5.333661624417512e-7)
(281559,5.333661624417512e-7)
(281567,5.333661624417512e-7)
(281573,5.333661624417512e-7)
(281574,5.333661624417512e-7)
(281581,5.333661624417512e-7)
(281585,5.333661624417512e-7)
(281593,5.333661624417512e-7)
(281627,5.333661624417512e-7)
(281646,5.333661624417512e-7)
(281647,5.333661624417512e-7)
(281700,5.333661624417512e-7)
```

```
(281715,5.333661624417512e-7)
(281728,5.333661624417512e-7)
(281778,5.333661624417512e-7)
(281785,5.333661624417512e-7)
(281813,5.333661624417512e-7)
(281849,5.333661624417512e-7)
(281865,5.333661624417512e-7)
(281888,5.333661624417512e-7)
```

Se puede comprobar que el pagerank es el mismo, y eso es porque son los nodos que no tienen salida, que siempre tienen el mismo pagerank, ya que la columna se de la matriz S es 1/n en todas sus posiciones.

7. Buscar otras matrices.

- Busqueda

He encontrado dos webs donde hay muchas matrices de conectividad.

La principal es esta:

http://www.cise.ufl.edu/research/sparse/matrices/list_by_dimension.html

Donde la matriz mas grande es http://www.cise.ufl.edu/research/sparse/matrices/LAW/webbase-2001.html con 118,142,155 nodos.

Por desgracia, los 3 modos que tiene de dar la matriz son: MATLAB mat-file, Matrix Market y Rutherford/Boeing. No estoy usando MATLAB, y los otros dos, ninguno es parecido al que nosotros usamos.

El mas parecido es el Matrix Market, asi que he creado una funcion en Julia para transformar de ese modo, al que nosotros conocemos:

```
function transform_file(path::String)
    io = open(path,"r")
    path_write = string(path, ".dat")

    I = (Int64,Int64)[]
    values = Dict()

    max_n = 0

    value = 0
    last = -1

    aument = 0

    for line in EachLine(io)
        h = split(line)
        if h == SubString{ASCIIString}[]
            values[last] = 1/value
            value = 0
            last = -1
            break
        end
        i,j = map(int64, h[1:2])
        if i == 0 || j == 0
            aument = 1
        end
        if aument == 1
            i = i+1
            j = j+1
        end

```

```
32          if i > max_n
33              max_n = i
34          elseif j > max_n
35              max_n = j
36          end
37
38          push!(I,(i,j))
39
40          value += 1
41          if i != last
42              values[last] = 1/(value-1)
43              value = 1
44              last = i
45          elseif last == -1
46              last = i
47          end
48
49      end
50      close(io)
51
52      sort!(I, by=x -> x[1])
53
54      default = 1/max_n
55      iow = open(path_write,"w")
56      for n in 1:length(I)
57          i = I[n][1]
58          j = I[n][2]
59          v = get(values, i, -1)
60          if v > 0
61              line = string(i, " ", j, " ", v)
62              println(iow,line)
63              #v = default
64          end
65      end
66      close(iow)
67  end
```

Aun asi el tamaño de la matriz superaria mi memoria RAM con creces (tiene 1,019,903,190 de no-ceros, los cuales serian (en 64 bits) (1019903190 * 8) / 1024 / 1024 / 1024 = 7.59 GB

La otra web que he encontrado es esta:

http://snap.stanford.edu/data/index.html

Donde vamos a usar dos:

web-Google: http://snap.stanford.edu/data/web-Google.html

- Estadisticas

```
Estadisticas cargar datos:
elapsed time: 12.160998068 seconds (3597467348 bytes allocated, 9.16% gc time)

Nodos: 916428

Tamaño (B): 21994312

Elementos no nulos: 5105027

La relacion no_nulos / nulos es: 6.078605846824962e-6 por lo que es dispersa.

Con Suma: 916428
Todas las sumas dan 1?: true
Todos tienen suma, todo da 1, la matriz es S

Dispersion: 5.570570737690249

Numero medio outlinks: 5.570570737690249

Nodos con salida: 739453
Nodos sin salida: 176975

Se cumple la relacion?: true
```

- Aplicacion del PageRank

```
Estadisticas calcular PR
Iteraciones del algoritmo: 162
elapsed time: 26.901678826 seconds (23043867384 bytes allocated, 15.37% gc time)
1.0147005206017322e-10
7331424
```

- 20 primeros:

Orden	Nodo	Pagerank
1	597622	0.0009145823650059178
2	41910	0.0009120143321654649
3	163076	0.0008950570007366836
4	537040	0.0008899355990041465
5	384667	0.0007791041598990437
6	504141	0.0007575433000608432
7	486981	0.0007177651982299156
8	605857	0.0007108492880017228

9	32164	0.0007055191574424469
10	558792	0.0007021667573242408
11	551830	0.0006950760029787621
12	765335	0.0006762284550594336
13	751385	0.0006546566632780029
14	425771	0.000616848808483503
15	908352	0.0006146228568605593
16	173977	0.0006031159399041975
17	7315	0.0005926650389814526
18	213433	0.0005894481857213108
19	885606	0.0005812667510718777
20	167295	0.0005765197022834081