Markov Chains, Random Walks and PageRank

The purpose of this notebook is to explore the algorithm of PageRank, developed by Google, and create toy examples while exploring extensive concepts like Markov matrices, probability and ranking websites.

Importing Libraries

Here we import all necessary libraries.

In [1]:
```
using LinearAlgebra
```

Processing

We create algorithms to process and conduct a mini PageRank.

In [3]:
```
# function userInput which gets user input, processing into L matrix
function userInput()
    # getting number of sites
    #println("How many websites do you have?")
    #websites = readline()
    #websites = parse(Int, websites)
    println("You have five websites.")
    websites = 5
    println("Format: 2 3 4 (website 1 points to 2, 3, 4)")

    v1 = []
    v2 = []
    v3 = []
    v4 = []
    v5 = []

    # iterating through sites
    for i in 1:websites
        # getting links each site points to
        print("For website number ")
        print(i)
        println(" input the links which it points to.")
        links = readline()
        # converting input to array
        arr = split(links, " ")
        tempVector = zeros(websites, 1)
        # looping through links
        for j in arr
            number = parse(Int, j)
            # normalizing and inputting into vector
            tempVector[number] = 1/length(arr)
            if i == 1
                v1 = tempVector
            elseif i == 2
                v2 = tempVector
```

```
            elseif i == 3
                v3 = tempVector
            elseif i == 4
                v4 = tempVector
            elseif i == 5
                v5 = tempVector
            end
        end
    end

    L = [reshape(v1, 1, :); reshape(v2, 1, :); reshape(v3, 1, :); reshape(v4, 1,

    return L
end
```

Out[3]:
```
userInput (generic function with 1 method)
```

Using Functions

Here, we assign variables to our functions and use them as such.

In [4]:
```
# this is our L matrix
L = transpose(userInput())
```

```
You have five websites.
Format: 2 3 4 (website 1 points to 2, 3, 4)
For website number 1 input the links which it points to.

For website number 2 input the links which it points to.

For website number 3 input the links which it points to.

For website number 4 input the links which it points to.

For website number 5 input the links which it points to.
```

Out[4]:
```
5×5 transpose(::Matrix{Float64}) with eltype Float64:
 0.0       0.5  0.0  1.0  1.0
 0.333333  0.0  0.5  0.0  0.0
 0.333333  0.0  0.0  0.0  0.0
 0.333333  0.5  0.5  0.0  0.0
 0.0       0.0  0.0  0.0  0.0
```

In [5]:
```
# this is our r vector
r = ones(5, 1)
for i in 1:length(r)
    r[i] = r[i]/5
end
```

Now, we create our Page Rank algorithm.

In [6]:
```
function PageRank(L, r, iterations)
    v = r
    for i in 1:iterations
        v = L * v
    end
    sorted = sort!(copy(v), dims = 1, rev = true)
    ranks = []
```

```
    for i in sorted

        for j in 1:5
            if v[j] == i && j ∉ ranks
                append!(ranks, j)
            end
        end
    end
    return v, ranks
end
```

Out[6]:
```
PageRank (generic function with 1 method)
```

In [72]:
```
# using PageRank with 100 iterations
v, ranks = PageRank(L, r, 100)
# page ranks for each website, top being rank
println(v)
println(ranks)
```

```
[0.3870967741935477; 0.19354838709677386; 0.1290322580645159; 0.290322580645160
8; 0.0]
Any[1, 4, 2, 3, 5]
```

In [47]:
```
# using PageRank function with 200 iterations
v2, ranks2 = PageRank(L, r, 200)
# page ranks for each website, top being rank
println(v2)
println(ranks2)
# can see that the steady state vector v2 does not change with more iterations
```

```
[0.3870967741935477; 0.19354838709677386; 0.1290322580645159; 0.290322580645160
8; 0.0]
Any[1, 4, 2, 3, 5]
```

Extension

We create the HITS algorithm as follows as an extension of what we've learned from PageRank.

In [8]:
```
# function userInput which gets user input, processing into L matrix
function userInput2()
    # getting number of sites
    #println("How many websites do you have?")
    #websites = readline()
    #websites = parse(Int, websites)
    println("You have five websites.")
    websites = 5
    println("Format: 2 3 4 (website 1 points to 2, 3, 4)")

    v1 = []
    v2 = []
    v3 = []
    v4 = []
    v5 = []

    # iterating through sites
    for i in 1:websites
        # getting links each site points to
```

```
        print("For website number ")
        print(i)
        println(" input the links which it points to.")
        links = readline()
        # converting input to array
        arr = split(links, " ")
        tempVector = []
        # looping through links
        for j in arr
            number = parse(Int, j)
            # normalizing and inputting into vector
            append!(tempVector, number)
            if i == 1
                v1 = tempVector
            elseif i == 2
                v2 = tempVector
            elseif i == 3
                v3 = tempVector
            elseif i == 4
                v4 = tempVector
            elseif i == 5
                v5 = tempVector
            end
        end
    end

    return [v1, v2, v3, v4, v5]
end
```

Out[8]:
```
userInput2 (generic function with 1 method)
```

In [9]:
```
# creating another matrix of websites and sites using
all = userInput2()
all
```

```
You have five websites.
Format: 2 3 4 (website 1 points to 2, 3, 4)
For website number 1 input the links which it points to.

For website number 2 input the links which it points to.

For website number 3 input the links which it points to.

For website number 4 input the links which it points to.

For website number 5 input the links which it points to.
```

Out[9]:
```
5-element Vector{Vector{Any}}:
 [2, 3, 4]
 [1, 4]
 [2, 4]
 [1]
 [1]
```

In [19]:
```
function hits(all, iterations)
    authority = [1.0, 1.0, 1.0, 1.0, 1.0]
    hub = [1.0, 1.0, 1.0, 1.0, 1.0]
    # number of iterations
    for i in 1:iterations
        # looping through each vector
```

```
        for j in 1:length(all)
            authoritySum = 0
            hubSum = 0
            # getting authority sum, to be used as hub score
            for k in 1:length(all[j])
                authoritySum += authority[all[j][k]]
            end
            # getting hub sum, to be used as authority score
            for n in 1:length(all)
                if j in all[n]
                    hubSum += hub[n]
                end
            end
            # setting authority and hub scores
            authority[j] = hubSum
            hub[j] = authoritySum
        end
        # normalizing the hubs and authorities
        hubNormalize = 0
        authorityNormalize = 0
        for p in 1:5
            hubNormalize += hub[p]*hub[p]
            authorityNormalize += authority[p]*authority[p]
        end
        hubNormalize = sqrt(hubNormalize)
        authorityNormalize = sqrt(authorityNormalize)
        # setting authority scores and hub scores to normalized scores
        for m in 1:5
            hub[m] = hub[m]/hubNormalize
            authority[m] = authority[m]/authorityNormalize
        end
    end
    return authority, hub
end
```

Out[19]:
```
hits (generic function with 1 method)
```

In [20]:
```
authority, hub = hits(all, 100)
println(authority)
println(hub)
```

```
[0.12416174248946872, 0.31935612933443747, 0.20826633263821837, 0.91608987498385
33, 0.0]
[0.3552126814227838, 0.43716229741559676, 0.7700798682565879, 0.211766467105475
9, 0.2117664671054759]
```

References

- https://en.wikipedia.org/wiki/PageRank#Simplified_algorithm
- http://blog.kleinproject.org/?p=280
- https://www.youtube.com/watch?v=F5fcEtqysGs