Homework 5

Achille Salaün

27 janvier 2016

Problem 1

Pareto Distribution to South Korea

The first path I found is the following one :

⋆ https://en.wikipedia.org/wiki/Pareto_ditribution
→ https://en.wikipedia.org/wiki/Economist
→ https://en.wikipedia.org/wiki/List_of_economists
→ https://en.wikipedia.org/wiki/Ha-Joon_Chang
→ https://en.wikipedia.org/wiki/South_Korea

However, in order to be sure to find a shortest path between this two pages, I decided to implement a tiny web-crawler (take in arguments starting point's and ending point's last part of the URL) :

```
import urllib, re, sys

def loadListURL(url):
      page = urllib.urlopen(url).read().splitlines()
      listURL = list()
      sublist = list()
      for i in range(0,len(page)):
            sublist = re.findall(r'href="/wiki/[∧"]"'[i])
            for j in range(0,len(sublist)):
                  listURL.append(sublist[j])
      for i in range(0,len(listURL)):
            listURL[i] = re.sub(r'href="','https://en.wikipedia.org',listURL[i])
            listURL[i] = re.sub(r'"','',listURL[i])
      return listURL

def noNone(tree):
      while None in tree:
            tree.remove(None)

def checkDuplicate(url,reference):
      if url in reference:
            return False
      else:
            reference.append(url)
            return True

def visit(tree,i,url,ref):
      noNone(tree)
      node = tree[i]
      tree.remove(tree[i])
      listURL = loadListURL(node[-1])
      if url not in listURL:
            for i in range(0,len(listURL)):
                  if checkDuplicate(listURL[i],ref):
                        new = list(node)
                        new.append(listURL[i])
```

```
                        tree.append(new)
                return []
        else:
                node.append(url)
                return node

def lookForNode(tree):
        n=0
        for i in range(1,len(tree)):
                if len(tree[i])<len(tree[n]):
                        n = i
        return n

start = "https://en.wikipedia.org/wiki/"+sys.argv[1]
print('start point = '+start)
end = "https://en.wikipedia.org/wiki/"+sys.argv[2]
print('end point = '+end)
ref=list()
listURL = loadListURL(start)
listPaths = list()
for i in range(0,len(listURL)):
        node = [start]
        node.append(listURL[i])
        listPaths.append(node)

shortestPath = []
k = 0
while shortestPath == []:
        noNone(listPaths)
        n = lookForNode(listPaths)
        k+=1
        #print(k)
        #print('n= '+str(n))
        #print('length= '+str(len(listPaths[n])))
        shortestPath = visit(listPaths,n,end,ref)

print('I visited '+str(k)+' pages to find, between '+start+' and '+end+',
the following shortest path :')
print(shortestPath)
print('You can link your two pages by only '+str(len(shortestPath)-1)+'
clicks!')
```

Thus, after having visited 407 pages, we get the following result :

⋆ https://en.wikipedia.org/wiki/Pareto_ditribution
→ https://en.wikipedia.org/wiki/Portal:Current_events
→ https://en.wikipedia.org/wiki/South_Korea

Cheeseburger to Political theory

Manually, I managed to obtain the following path :

$\star$ https://en.wikipedia.org/wiki/Cheeseburger
$\rightarrow$ https://en.wikipedia.org/wiki/Halakha
$\rightarrow$ https://en.wikipedia.org/wiki/Law\#Legal_theory
$\rightarrow$ https://en.wikipedia.org/wiki/Political_philosophy

After having browsed 1048 pages, the shortest path suggested by my web-crawler is :

$\star$ https://en.wikipedia.org/wiki/Cheeseburger
$\rightarrow$ https://en.wikipedia.org/wiki/Cheese
$\rightarrow$ https://en.wikipedia.org/wiki/United_States
$\rightarrow$ https://en.wikipedia.org/wiki/Political_philosophy

Problem 2

Q1

The degree distribution is the following one :

$$\begin{array}{cccc} P & : & \mathbf{N} & \rightarrow & \mathbf{R}^+ \\ & & k=1 & \mapsto & \frac{N-1}{N}(1-p)^2 \\ & & k=2 & \mapsto & \frac{N-1}{N}p(1-p) \\ & & k=3 & \mapsto & \frac{N-1}{N}p^2 \\ & & k=(N-1) & \mapsto & \frac{1}{N} \end{array}$$

Q2

A node is either the center with a probability $\frac{1}{N}$ whose the degree is $N-1$, or a leaf with a probability $\frac{N-1}{N}$ whose the degree is $1+2p$ (linked to the center, and maybe with the leaves before and after). Thus, the average degree is :

$$<k>=\overbrace{\frac{N-1}{N}(1+2p)}^{leaves}+\overbrace{\frac{1}{N}(N-1)}^{center}=2\frac{N-1}{N}(1+p)$$

Q3

The clustering coefficient is :

$$C=\frac{3p(N-1)}{\frac{(N-1)(N-2)}{2}}=\frac{6p}{N-2}$$

Q4

Let derive the new average degree :

$$< k > (q) = \frac{N-1}{N}(q+2p) + \frac{q}{N}(N-1) = 2\frac{N-1}{N}(p+q)$$

Furthermore, the giant component disapears as soon as $< k > (q) < 1$, so :

$$q < \frac{N}{2(N-1)} - p$$