暨南大学本科实验报告专用纸

课程名称	数值计算实验
成绩评定	
实验项目名称	PageRank problem
指导教师	Liangda Fang
实验项目编号	04
实验项目类型	综合性
实验地点	N116
学生姓名	甄洛生、吴凯一
学号	2018054625、2018050821
学院	信息科学技术学院
系	计算机系
专业	计算机科学技术

实验时间 2020 年 12 月 15 日下午 10:30~ 12:10

I. Problems

1. Implement a simple web crawler
2. Acquire the google matrix of the first 500 web pages from the main webpage of any university via the above web crawler, and give their adjacency matrix
3. Implement the Power Method
4. Compute the dominant eigenvector of the google matrix
5. List the top 20 web pages

II. Algorithm summary

● PageRank Algorithm

PageRank algorithm, designed by **Google**, can calculate **the importance of a web page** through **the relevance between web pages**.

Google search engine uses this algorithm to return the search results sorted by page importance. This algorithm is based on **Markov process** and **power iteration method**. Let's look at the following example:

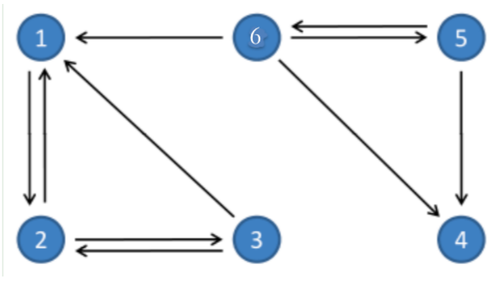


Figure 1 Page links diagram

This is a directed graph, and the out-link means a website refer another page. As you can see, Site 1 refers to site 2 and site 3. In computer science, we usually use the adjacent matrix to represent a directed graph. So, as follow:

$$\begin{bmatrix} 0 & 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

Figure 2 Adjacent matrix

OK, call the adjacent matrix A, and $A_{ij} = 1$ means that site j refers to site i. If not refer and $A_{ij} = 0$. That is why the adjacent matrix A can represent a directed graph.

Naturally, web pages should have a priority, and high-priority pages should be displayed at the top to make it easier for users to click. However, how do we know the importance of these pages? We only know the relevancy between web pages, maybe it can help us.

If a web page is referenced by many other web pages, it is likely to show that it is more important! In the same way, the importance of those pages referenced by high-priority pages should be important. And intuition tells us that people are more likely to stay on web pages with high importance.

Now, let's imagine that there is a student, called Rosen, who is surfing the Internet. Suppose he is on site i now. Rosen may then **move to a random site** with probability α, or **click a link on the current page** $\boldsymbol{i}$ with probability $1 - \alpha$. We assume that the probability of Rosen clicking on a link on site i is **equal** and n is the number of websites. Then, the probability of Rosen goes from site i to j can be expressed as:

$$p = \frac{\alpha}{n} + (1 - \alpha) * \frac{A_{ji}}{\sum_{k=1}^{n} A_{ki}} \qquad (1 - 1)$$

Let's take a look again at the random process of Rosen surfing the Internet. Given **the current state** (i.e. staying on the web page i) and **all the historical states**, **the conditional probability distribution** of **its future state** (i.e. going to the web page j) depends <u>**only on the current state**</u>. In other words, given the current state, it is conditionally independent of the historical state (i.e. the historical record when Rosen surfing the Internet). We say that the random process of Rosen surfing on the Internet has **Markov property**, and the process with Markov property is usually called **Markov process**.

However, have you noticed that **page 4** does **not reference any web pages**! In other words, when Rosen stays on page 4, he will move to a random page with a 100% probability! Therefore, we need to make a small adjustment for $(1 - 1)$:

$$p_{ji} = \begin{cases} \dfrac{\alpha}{n} + (1 - \alpha) * \dfrac{A_{ji}}{\sum_{k=1}^{n} A_{ki}}, & other \\ \dfrac{1}{n}, & \displaystyle\sum_{k=1}^{n} A_{ki} = 0 \end{cases} \qquad (1 - 2)$$

$$p_{ji}\ means\ Rosen\ goes\ from\ site\ i\ to\ j.$$

Now, let's transform our adjacent matrix A into a "surfing" matrix G by $(1 - 1)$:

$$G = \begin{bmatrix} \frac{\alpha}{6} & \frac{\alpha}{6}+\frac{(1-\alpha)}{2} & \frac{\alpha}{6}+\frac{(1-\alpha)}{2} & \frac{1}{6} & \frac{\alpha}{6} & \frac{\alpha}{6}+\frac{(1-\alpha)}{3} \\ \frac{\alpha}{6}+(1-\alpha) & \frac{\alpha}{6} & \frac{\alpha}{6}+\frac{(1-\alpha)}{2} & \frac{1}{6} & \frac{\alpha}{6} & \frac{\alpha}{6} \\ \frac{\alpha}{6} & \frac{\alpha}{6}+\frac{(1-\alpha)}{2} & \frac{\alpha}{6} & \frac{1}{6} & \frac{\alpha}{6} & \frac{\alpha}{6} \\ \frac{\alpha}{6} & \frac{\alpha}{6} & \frac{\alpha}{6} & \frac{1}{6} & \frac{\alpha}{6}+\frac{(1-\alpha)}{2} & \frac{\alpha}{6}+\frac{(1-\alpha)}{3} \\ \frac{\alpha}{6} & \frac{\alpha}{6} & \frac{\alpha}{6} & \frac{1}{6} & \frac{\alpha}{6} & \frac{\alpha}{6}+\frac{(1-\alpha)}{3} \\ \frac{\alpha}{6} & \frac{\alpha}{6} & \frac{\alpha}{6} & \frac{1}{6} & \frac{\alpha}{6}+\frac{(1-\alpha)}{2} & \frac{\alpha}{6} \end{bmatrix}$$

G is so-called "Google matrix/Stochastic Matrix/Markov Matrix". Noted that an interesting fact, all elements of G is positive and $\sum_{j=1}^{n} G_{ij} = 1$. That is to say, G exists a left eigenvector $x = (1,1,\dots,1)$:

$$xG = x \tag{1-3}$$

That is to say, G has a eigenvalue of 1. According to **Gershgorin circle theorem**, we know that the eigenvalues of G **cannot be greater than 1**. Because G is positive, according to **Perron-Frobenius theorem**, we know that **the absolute values of all the eigenvalues of the matrix are less than 1** except the eigenvalue we just find (i.e. 1).

Now, we say matrix G has a steady-state distribution. What does it mean? Suppose there is a vector P, and P_i means the number of people staying on page i. Assume:

$$P = (100,0,0,\dots,0)$$

It means 100 people staying on page 1 at initial time. After a long time, we are now looking for the number of people staying on the current web pages:

$$P_{now} = \lim_{n\to\infty} G^n P \tag{1-4}$$

You'll be surprised to find P_{now} converged! This magic is thanks to 1 is the most greatest absolute eigenvalue of G! Look at this:

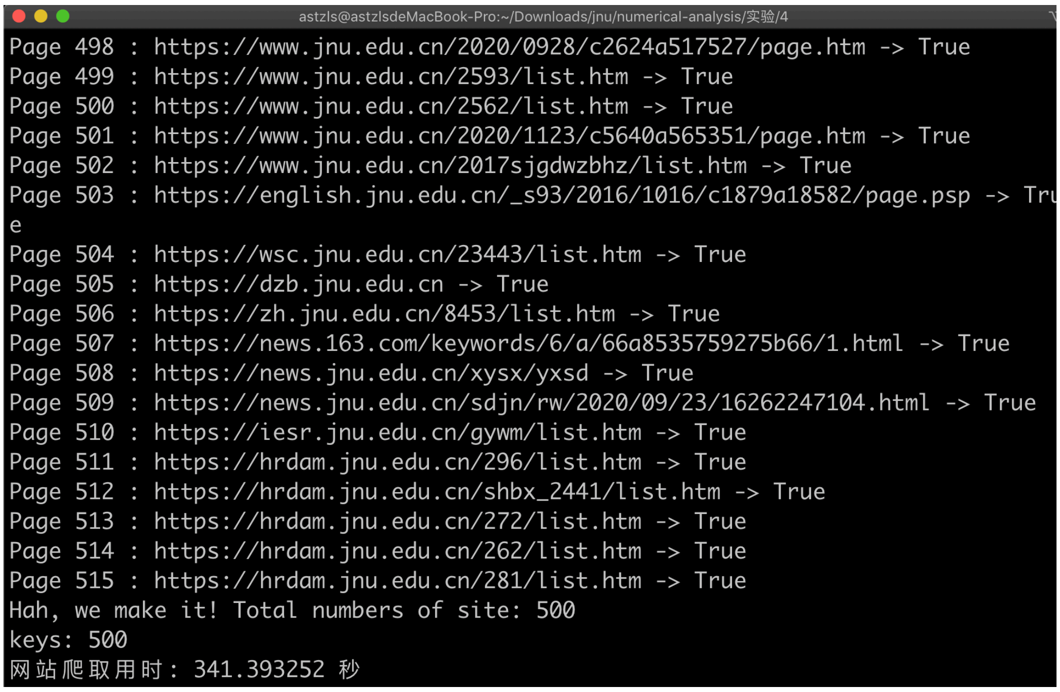
$$Assume\ c_1, c_2 \neq 0\ and\ v_i\ is\ eigenvector\ of\ G$$
$$And\ |\lambda_1| > |\lambda_2| \geq |\lambda_3| \geq \cdots \geq |\lambda_n|$$
$$P = c_1 v_1 + \cdots + c_n v_n$$
$$P_k = G^k P = c_1 \lambda_1^k v_1 + \cdots + c_n \lambda_n^k v_n$$
$$\frac{P_k}{\lambda_1^k} = c_1 v_1 + c_2 \left(\frac{\lambda_2}{\lambda_1}\right)^k v_2 + \cdots + c_n \left(\frac{\lambda_n}{\lambda_1}\right)^k v_n$$
$$\lim k \to \infty, \left(\frac{\lambda_j}{\lambda_1}\right)^k \to 0$$

And that is why P_{now} will be converged! And it will converge to the eigenvector of the greatest absolute eigenvalue of G to a multiple. And that is the so-called **power iteration method.**

Well done, we just sorted the vector P_{now} from big to small, and we get the rank of page! We make it, isn't it?

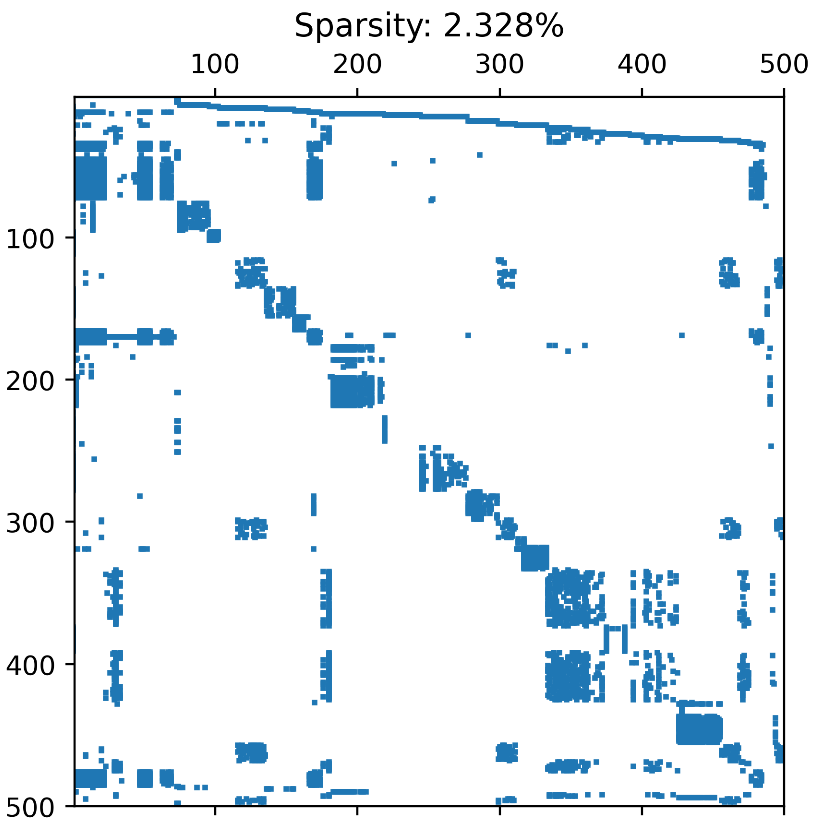
III. Result analysis

Crawling websites:

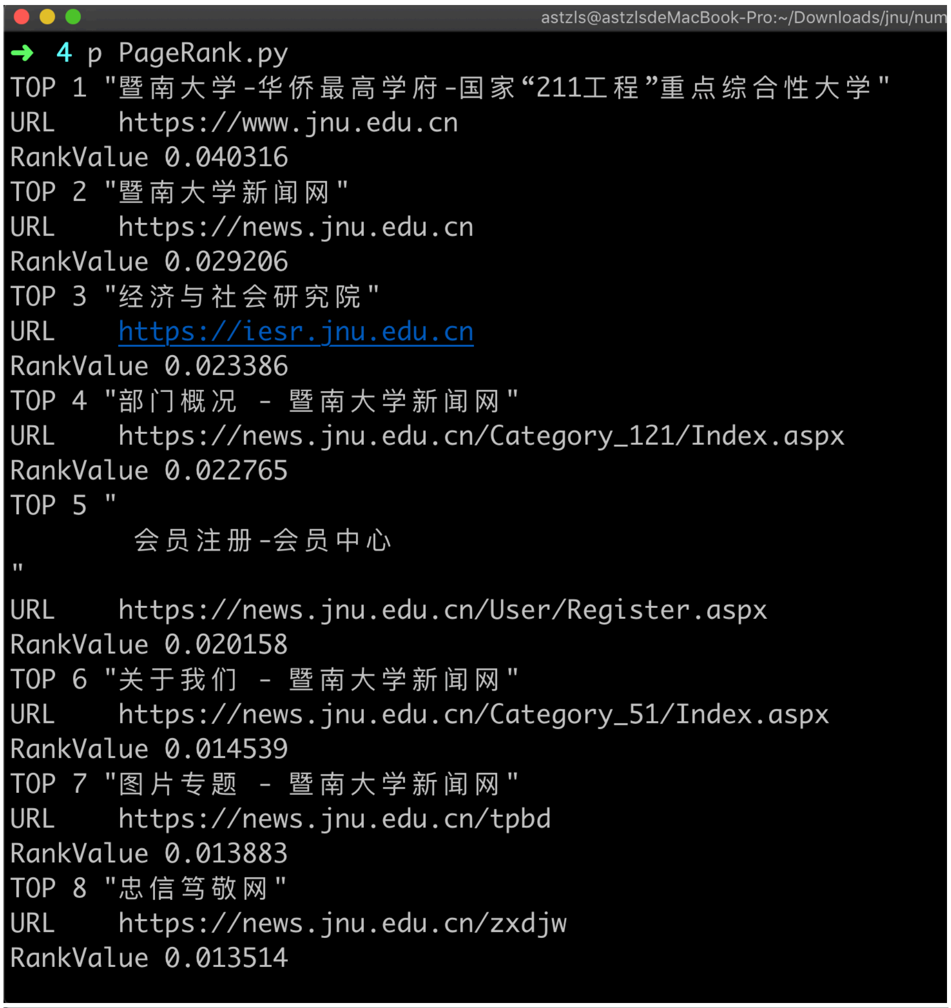


```
astzls@astzlsdeMacBook-Pro:~/Downloads/jnu/numerical-analysis/实验/4
Page 498 : https://www.jnu.edu.cn/2020/0928/c2624a517527/page.htm -> True
Page 499 : https://www.jnu.edu.cn/2593/list.htm -> True
Page 500 : https://www.jnu.edu.cn/2562/list.htm -> True
Page 501 : https://www.jnu.edu.cn/2020/1123/c5640a565351/page.htm -> True
Page 502 : https://www.jnu.edu.cn/2017sjgdwzbhz/list.htm -> True
Page 503 : https://english.jnu.edu.cn/_s93/2016/1016/c1879a18582/page.psp -> Tru
e
Page 504 : https://wsc.jnu.edu.cn/23443/list.htm -> True
Page 505 : https://dzb.jnu.edu.cn -> True
Page 506 : https://zh.jnu.edu.cn/8453/list.htm -> True
Page 507 : https://news.163.com/keywords/6/a/66a8535759275b66/1.html -> True
Page 508 : https://news.jnu.edu.cn/xysx/yxsd -> True
Page 509 : https://news.jnu.edu.cn/sdjn/rw/2020/09/23/16262247104.html -> True
Page 510 : https://iesr.jnu.edu.cn/gywm/list.htm -> True
Page 511 : https://hrdam.jnu.edu.cn/296/list.htm -> True
Page 512 : https://hrdam.jnu.edu.cn/shbx_2441/list.htm -> True
Page 513 : https://hrdam.jnu.edu.cn/272/list.htm -> True
Page 514 : https://hrdam.jnu.edu.cn/262/list.htm -> True
Page 515 : https://hrdam.jnu.edu.cn/281/list.htm -> True
Hah, we make it! Total numbers of site: 500
keys: 500
网站爬取用时: 341.393252 秒
```

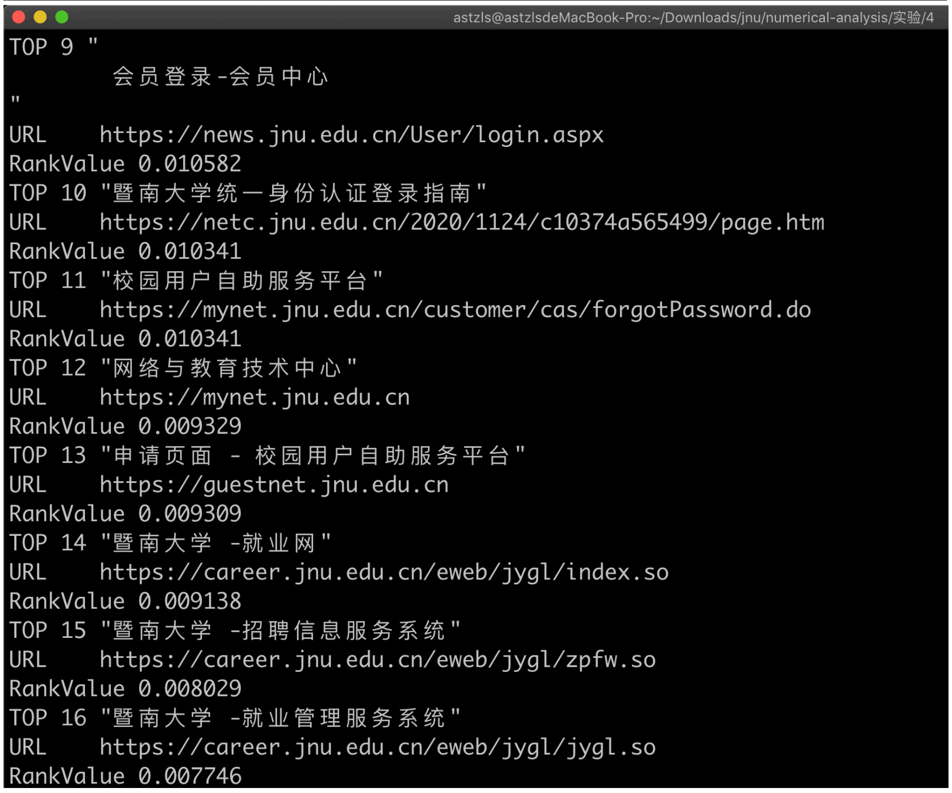
The 500 pages adjacent matrix:



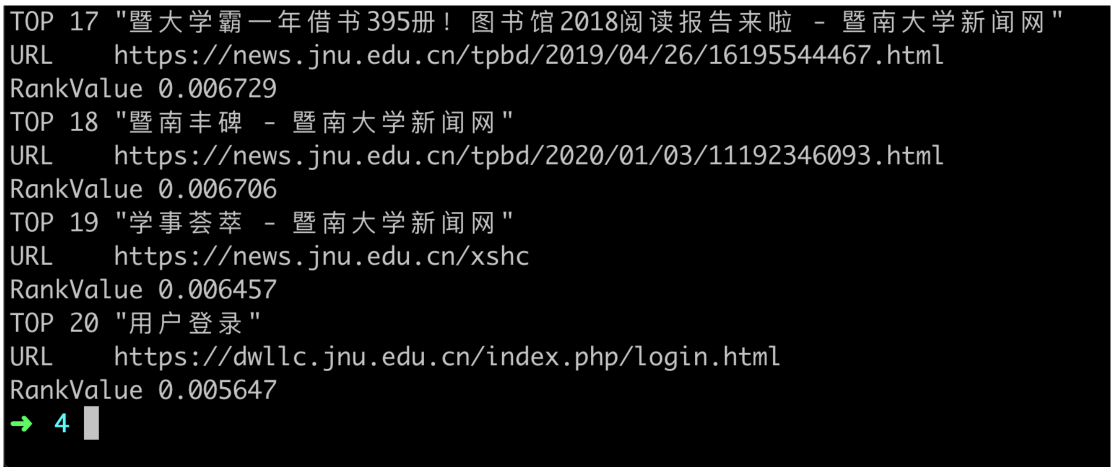
The top 20 web pages:



```
➜ 4 p PageRank.py
TOP 1 "暨南大学-华侨最高学府-国家“211工程”重点综合性大学"
URL    https://www.jnu.edu.cn
RankValue 0.040316
TOP 2 "暨南大学新闻网"
URL    https://news.jnu.edu.cn
RankValue 0.029206
TOP 3 "经济与社会研究院"
URL    https://iesr.jnu.edu.cn
RankValue 0.023386
TOP 4 "部门概况 - 暨南大学新闻网"
URL    https://news.jnu.edu.cn/Category_121/Index.aspx
RankValue 0.022765
TOP 5 "
        会员注册-会员中心
"
URL    https://news.jnu.edu.cn/User/Register.aspx
RankValue 0.020158
TOP 6 "关于我们 - 暨南大学新闻网"
URL    https://news.jnu.edu.cn/Category_51/Index.aspx
RankValue 0.014539
TOP 7 "图片专题 - 暨南大学新闻网"
URL    https://news.jnu.edu.cn/tpbd
RankValue 0.013883
TOP 8 "忠信笃敬网"
URL    https://news.jnu.edu.cn/zxdjw
RankValue 0.013514
```

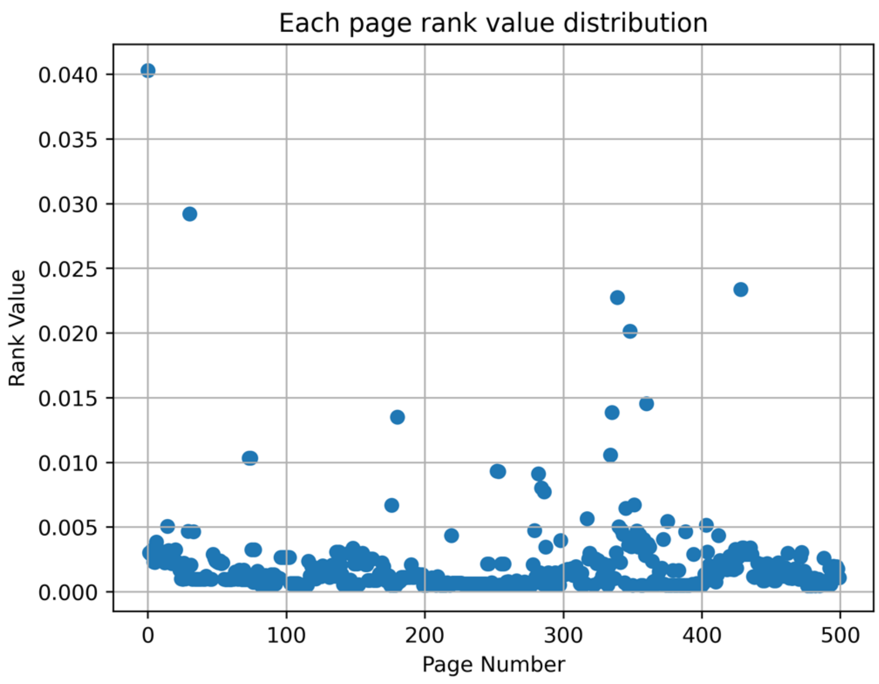


```
TOP 9 "
        会员登录-会员中心
"
URL    https://news.jnu.edu.cn/User/login.aspx
RankValue 0.010582
TOP 10 "暨南大学统一身份认证登录指南"
URL    https://netc.jnu.edu.cn/2020/1124/c10374a565499/page.htm
RankValue 0.010341
TOP 11 "校园用户自助服务平台"
URL    https://mynet.jnu.edu.cn/customer/cas/forgotPassword.do
RankValue 0.010341
TOP 12 "网络与教育技术中心"
URL    https://mynet.jnu.edu.cn
RankValue 0.009329
TOP 13 "申请页面 - 校园用户自助服务平台"
URL    https://guestnet.jnu.edu.cn
RankValue 0.009309
TOP 14 "暨南大学 -就业网"
URL    https://career.jnu.edu.cn/eweb/jygl/index.so
RankValue 0.009138
TOP 15 "暨南大学 -招聘信息服务系统"
URL    https://career.jnu.edu.cn/eweb/jygl/zpfw.so
RankValue 0.008029
TOP 16 "暨南大学 -就业管理服务系统"
URL    https://career.jnu.edu.cn/eweb/jygl/jygl.so
RankValue 0.007746
```



```
TOP 17 "暨大学霸一年借书395册！ 图书馆2018阅读报告来啦 - 暨南大学新闻网"
URL    https://news.jnu.edu.cn/tpbd/2019/04/26/16195544467.html
RankValue 0.006729
TOP 18 "暨南丰碑 - 暨南大学新闻网"
URL    https://news.jnu.edu.cn/tpbd/2020/01/03/11192346093.html
RankValue 0.006706
TOP 19 "学事荟萃 - 暨南大学新闻网"
URL    https://news.jnu.edu.cn/xshc
RankValue 0.006457
TOP 20 "用户登录"
URL    https://dwllc.jnu.edu.cn/index.php/login.html
RankValue 0.005647
➜ 4
```

Website rank value distribution:



IV. Experimental summary

In this experiment, we crawled a total of **500 websites** from **https://www.jnu.edu.cn**, the official website of Jinan University, and obtained the adjacent matrix A.

Using A we construct Google matrix G, use **the power iterative algorithm PI** to analyze the importance of these sites and get their rankings.

However, when we construct the adjacent matrix A, we should make sure that the **column sum** is **1**, **not the row sum is 1**.

Also, when crawling a web page, you should **limit** the number of **relative URL** in a web page. For example, in https://www.jnu.edu.cn , we limit the number of relative url (e.g. https://www.jnu.edu.cn/balabala.html) to 20 at most. The reason for this is to avoid that nodes are made up of sub-pages of the same site. At the same time, we found a large number of the same sites (such as http://www.jnu.edu.cn and https://www.jnu.edu.cn and https://www.jnu.edu.cn/ are the same), so we only crawled those url, that began with **https** and removed the invalid "/" at the end.

V. Appendix: Source code

In this experiment, we code in python. For running code as follows, you should make sure those libraries has been installed which are:

1. beautifulsoup
2. requests
3. matplotlib
4. numpy

Crawler.py

```python
from requests import get, head
from bs4 import BeautifulSoup
from random import sample
from urllib.parse import urljoin
from time import sleep

# 设置每个节点至多出边、节点总数
NUM_SITE_LIM = 500
RELATIVE_OUTLINKS_LIM = 20

class Crawler:
    headers = {
        'user-agent': 'Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14_5) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/87.0.4280.88 Safari/537.36'
    }
    num_site = 0                # 已爬取网站个数
    crawling = []               # 待爬池
    checked = set()             # 已爬池
    unreachable_site = set()    # 坏节点池
    link_graph = {}             # 网站连接图

    def __init__(self, baseURL, verbose):
        # verbose 为 True 时，打印 debug 信息
        self.crawling.append(baseURL)
        self.num_site = self.num_site + 1
        self.verbose = verbose

    def extractOutLink(self, url):
        try:
            response = head(url)
            if 'text/html' not in response.headers['content-type']:
                raise Exception
            response = get(url, headers=self.headers, timeout=10)
        except:
            # 网站在指定时间到不了，或者 url 不是返回 html
            self.checked.remove(url)
            self.unreachable_site.add(url)
            self.num_site = self.num_site - 1
            return False

        # 设置为 utf-8，否则中文乱码
        response.encoding = 'utf-8'
        soup = BeautifulSoup(response.text, 'lxml')

        # 这里保证 outlinks 是绝对 url
        # 并且限制相对 url 的数目（避免太多非首页的 url）
        # 并且只爬 https
        outlinks = soup.find_all('a', href=True)
        relative_urls = [url['href'] for url in outlinks if url['href'].startswith('/')]
        absolute_urls = [url['href'] for url in outlinks if url['href'].startswith('https')]
        relative_urls = sample(relative_urls, min(len(relative_urls), RELATIVE_OUTLINKS_LIM))
        absolute_urls = absolute_urls + [urljoin(url, outlink) for outlink in relative_urls]
        outlinks = absolute_urls
        # 去除 link 后无效的'/'
        outlinks = [outlink[:-1] if outlink.endswith('/') else outlink for outlink in outlinks ]

        for outlink in outlinks:
            if outlink not in self.crawling and \
               outlink not in self.checked  and \
               outlink not in self.unreachable_site and \
               self.num_site < NUM_SITE_LIM:
                self.crawling.append(outlink)
                self.num_site = self.num_site + 1

        self.link_graph[url] = outlinks
        response.close()
        return True

    def run(self):
        #print('要是我 ip 被封了你妈买菜必涨价')
        index = 1
        while self.crawling:
            url = self.crawling.pop(0)
            self.checked.add(url)
            status = self.extractOutLink(url)
            if self.verbose:
                print('Page', index,':', url, '->', status)
                index = index + 1
        if self.verbose:
            print('Hah, we make it! Total numbers of site:', self.num_site)
            print('keys:', len(self.link_graph.keys()))

    def getAdjacentMatrix(self):
        # 从有向图转成相邻矩阵
        keys = tuple(self.link_graph.keys())

        for key in keys:
            for site in self.link_graph[key]:
                if site not in keys or site == key:
                    self.link_graph[key] = [x for x in self.link_graph[key] if x != site]

        M = [ [0]*self.num_site for i in range(self.num_site) ]

        for site, outlinks in self.link_graph.items():
            i = keys.index(site)
            for outlink in outlinks:
                j = keys.index(outlink)
                M[i][j] = 1

        return M

def getUrlTitle(url):
    # 返回对应 URL 的 <title>
    headers = {
        'user-agent': 'Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14_5) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/87.0.4280.88 Safari/537.36'
    }
    response = get(url, headers=headers)
    response.encoding = 'utf-8'
    soup = BeautifulSoup(response.text, 'lxml')
    response.close()
    return soup.find('title').text
```

PowerIteration.py

```python
import numpy as np

alpha = 0.15

def power_iteration(M):
    n = M.shape[0]

    # x0 = [1 0 0 ... 0]
    x = [0 for i in range(n)]
    x[0] = 1
    x = np.array(x)

    # e = [1 1 ... 1]
    e = np.array([1 for i in range(n)])


    # 常规解法
    while True:
        x_new = np.dot(M,x)
        dist = np.linalg.norm(x_new - x)
        if dist < 1e-20:    # 我随便设的阈值
            break
        else:
            x = x_new
    return x_new
```

PageRank.py

```python
from Crawler import Crawler, getUrlTitle
from PowerIteration import power_iteration
from FigGen import gen_result_fig, gen_sparse_fig
from os.path import exists
from json import dump, load
from datetime import datetime
import numpy as np

if not exists('data.json'):
    crawler = Crawler(
        baseURL='https://www.jnu.edu.cn',
        verbose=True # False 则不显示 debug 信息
    )

    start = datetime.now()
    crawler.run()
    end = datetime.now()
    gap = end - start
    print('网站爬取用时:', gap.total_seconds(), '秒')

    data = dict()
    data['adjacent_matrix'] = crawler.getAdjacentMatrix()
    data['directed_graph'] = crawler.link_graph
    data['num_site'] = crawler.num_site

    with open('data.json', 'w', encoding='utf-8') as f:
        dump(data, f, ensure_ascii=False, indent=4)

with open('data.json', 'r') as f:
    data = load(f)

M = data.get('adjacent_matrix')
G = data.get('directed_graph')
num_site = data.get('num_site')
alpha = 0.15

gen_sparse_fig(M, filename='Assets/adjacent_matrix.png')   # 绘制稀疏矩阵

# 相邻矩阵 M to Google 矩阵 M
for row in range(num_site):
    row_sum = sum(M[row])
    if row_sum == 0:
        for col in range(num_site):
            M[row][col] = 1 / num_site
    else:
        for col in range(num_site):
            random_surf = alpha * (1 / num_site)
            outlink_surf = (1 - alpha) * (M[row][col] / row_sum)
            M[row][col] = random_surf + outlink_surf


# 幂迭代
result = power_iteration(np.array(M).transpose())
gen_result_fig(result, 'Assets/result.png')
result = [(key, result[i]) for i, key in enumerate(G.keys())]

# Rank
page_rank = sorted(result, key=lambda x:x[1], reverse=True)

```

```python
# 显示前 20
for i in range(20):
    msg = 'TOP %d "%s"\nURL    %s\nRankValue %f' % (i+1, getUrlTitle(page_rank[i][0]), page_rank[i][0], page_rank[i][1])
    print(msg)

# 保存 PageRank
with open('page_rank.json', 'w') as f:
    dump(page_rank, f)
```

FigGen.py

```python
import matplotlib.pyplot as plt
import numpy as np

def gen_sparse_fig(M, filename):
    n = len(M)
    plt.spy(M, markersize=1)
    plt.xlim([1,500])
    plt.ylim([500,1])
    # 统计稀疏率
    non_zero_elements = 0
    for i in range(len(M)):
        non_zero_elements = non_zero_elements + sum(M[i])

    title = 'Sparsity: {:.3f}%'.format((non_zero_elements*100)/(n*n))
    plt.title(title)

    plt.savefig(
        filename,
        dpi=400,
        bbox_inches='tight'
    )
    plt.close()

def gen_result_fig(result, filename):
    X = [x for x in range(len(result))]
    Y = result
    plt.scatter(X, Y)
    plt.title('Each page rank value distribution')
    plt.xlabel('Page Number')
    plt.ylabel('Rank Value')
    plt.grid('on')
    plt.savefig(
        filename,
        dpi=400
    )
    plt.close()
```