电工导 C 第二次实验报告

姓名: 宋士祥　　学号:521030910013　　班级:F2103001

2022 年 9 月 29 日

1 实验概览

本次实验在第一次实验的基础上，学习了 HTTP 协议与爬虫的相关概念。

对于 HTTP 协议，本次实验介绍了 HTTP 请求的相关基本知识和基础架构，学习了 HTML 表单的相关知识。在此基础上，本次实验利用 Python 模拟 GET 请求从而实现在搜索框等位置输入检索的目的。在此基础上，又学习了 Python 添加 header 和添加 Post 模拟登录的相关知识。其中部分内容已在实验一学习过。

对于爬虫，网络爬虫是一种按照一定的规则，自动抓取万维网信息的程序或者脚本。实验介绍了国际通用的`robots.txt` 协议（道德）规范。并介绍了两种常见的搜索方法：广度优先搜索策略（Breadth First Search, BFS）和深度优先搜索策略（Depth First Search, DFS）。BFS 是根据每个节点的宽度遍历每层节点，而 DFS 则是会沿着树枝尽量到达树的尽端，达到尽端后返回。在爬虫中，BFS 一般爬取当前页面的界面的页面不断延伸，而 DFS 则是优先向深度方向进行爬取。

在上述知识内容的基础上，本次实验共需完成四个任务：

1. 模拟登录药智网`https://www.yaozh.com/`爬取并返回个人信息
   (Bonus) 模拟登录知乎网`https://www.zhihu.com/`爬取并返回个人信息，以及赞同的前两个问题。（采用`lxml` 中`xpath` 的方式返回需要抓取内容的地址）。
2. 完成 BFS 算法的代码
3. 利用给定的代码结构分别利用 BFS 和 DFS 将图的结构补全
4. 利用给定的代码和在练习 2，练习 3 中任务的基础上进行网络页面爬取的实操。对 `https://www.sjtu.edu.cn` 的超链接进行爬取

2 实验环境

本次实验采用所需的实验环境如下：

- Docker 中的`sjtumic/ee208` 镜像
- Python3（使用 VSCode 编译）
- `BeautifulSoup4` 扩展以及`lxml` 扩展。

3 练习 1

3.1 问题重述与代码说明

练习 1 要求我们首先在药智网（https://www.yaozh.com/）注册并输入个人信息，利用 cookie 登录自己的个人信息页（https://www.yaozh.com/member/basicinfo/），并爬取自己的相关信息。

大部分代码内容在example1.py 均已给出，仅需修改报头信息和账号密码并打印出所需结果即可。出于对个人隐私信息的保护，本题和 Bouns 题账号均为_Pikachu，密码均为SJTUEE208。

根据 ppt 中的提示，所需爬取的信息在 中，因此仅需利用与第一次 Lab 相似的内容即可完成任务。

核心代码展示：

```
# 10. The rest is done by you
# 创建一个并爬取对应的链接soup
soup = BeautifulSoup(response,features="lxml")
my_identity = soup.find("div",{"class":"U_myinfo clearfix"})
# 注意到值在<input value=...> 中，因此需要打印出该信息即可
print("真实姓名:\t"+my_identity.contents[3].find("input").get("value",""))
print("用户名: \t"+my_identity.contents[5].find("input").get("value",""))
print("性别:  \t"+my_identity.contents[7].find("input").get("value",""))
print("出生年月: \t"+my_identity.contents[9].find("input").get("value",""))
# 个人简介为直接的文字
print("个人简介: \t"+my_identity.contents[11].find("textarea").contents[0])
```

3.2 结果展示

```
root@8b76667ea0f2:/workspaces/lab2-Crawler/code# python example1.py
真实姓名:       宋士祥
用户名:         _Pikachu
性别:           1
出生年月:       2003-02-08
个人简介:       This ia an assignment for SJTU EE208.
```

3.3 问题讨论

3.3.1 页面的登录问题

当我们运行example1.py 后，我们如果刷新界面，我们会发现系统会自动退出登录。这是因为，我们在 Python 模拟登录时是执行了一次登录操作。而对于该网站而言不支持多端同时登录的操作，因此会自动退出。

3.3.2 页面的 Cookie

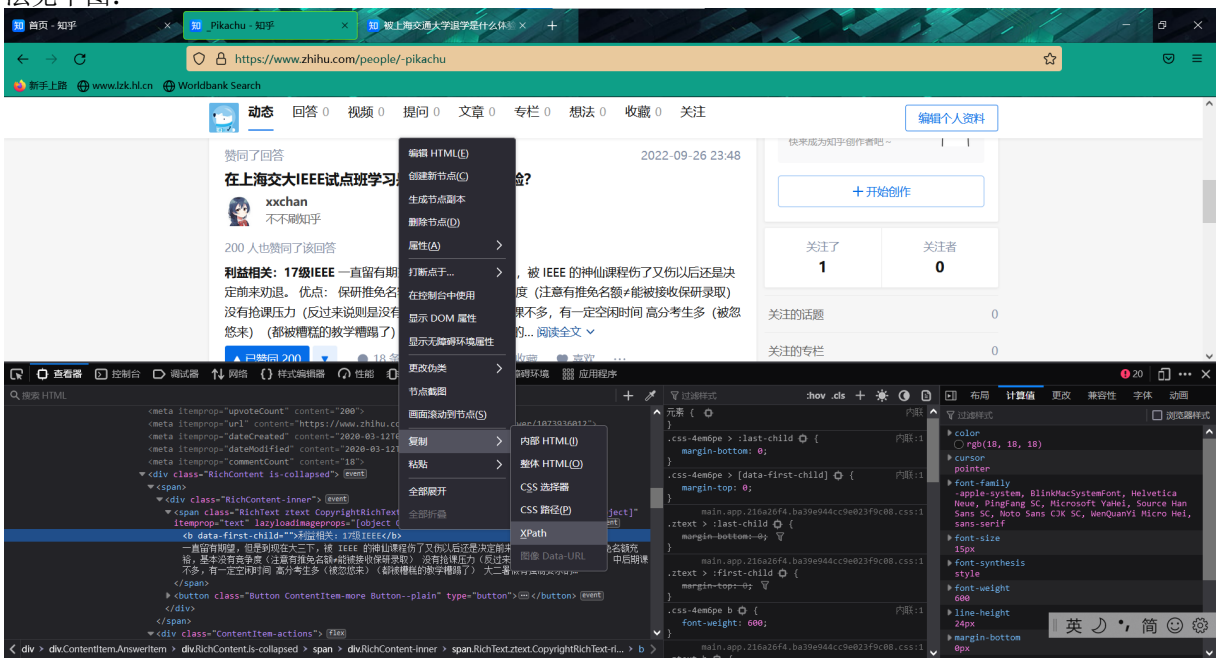
在给定代码中的 cookie，其实是一种临时保留在本地的文件形式。Cookie 在网站内与特定的 Web 关联在一起，保存了这个文本也即保存了该客户端访问这一 Web 时的相关信息。当用户再次访问这一网站时这些信息可以被复用。药智网的登录正是利用了这一特性。

4 练习 1-Bonus

4.1 问题重述与代码说明

该练习与练习 1 相似。不同之处在于我们采用了`lxml` 进行爬取而非`BeautifulSoup`。此外，我们同时需要爬取对应的问题和解答。

对于爬取的 Xpath，最快的方法是直接通过元素查找找到对应的完整 XPath，具体的查找方法见下图：



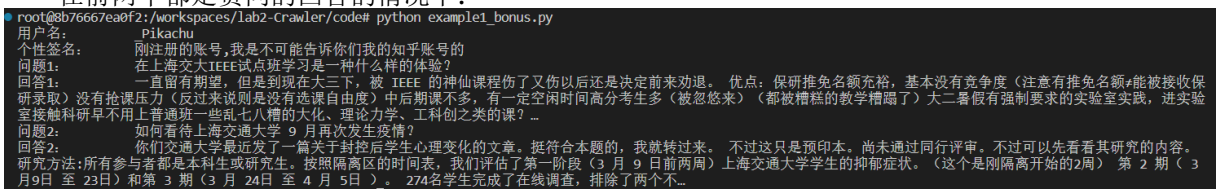
核心代码展示：

```
# The rest is done by you:
myname = tree.xpath("/html/body/div[1]/div/main/div/div[1]/div/div[2]/div/div[2]/div[1]/
                                                h1/span[1]")[0].text
my_signature = tree.xpath("/html/body/div[1]/div/main/div/div[1]/div/div[2]/div/div[2]/
                                                div[1]/h1/span[2]")[0].text
print("用户名: \t"+myname)
print("个性签名: \t"+my_signature)
# 以防止这个问题不是赞同回答是关注问题
try:
    question1 = tree.xpath("/html/body/div[1]/div/main/div/div[2]/div[1]/div/div[3]/div/
                                                    div[2]/div[1]/div[2]/div/h2/div/meta[2]"
                                                    )[0].get("content")
    answer1 = tree.xpath("/html/body/div[1]/div/main/div/div[2]/div[1]/div/div[3]/div/
                                                    div[2]/div[1]/div[2]/div/div[2]/span/div
                                                    /span/text()")[0]
    print("问题: 1  \t"+question1)
    print("回答: 1  \t"+answer1)
except:
    print("问题1 Error:\t 这个标签不是赞同的回答")
try:
    question2 = tree.xpath("/html/body/div[1]/div/main/div/div[2]/div[1]/div/div[3]/div/
                                                    div[2]/div[2]/div[2]/div/h2/div/meta[2]"
                                                    )[0].get("content")
    answer2 = tree.xpath("/html/body/div[1]/div/main/div/div[2]/div[1]/div/div[3]/div/
                                                    div[2]/div[2]/div[2]/div/div[2]/span/div
                                                    /span/text()")[0]
```

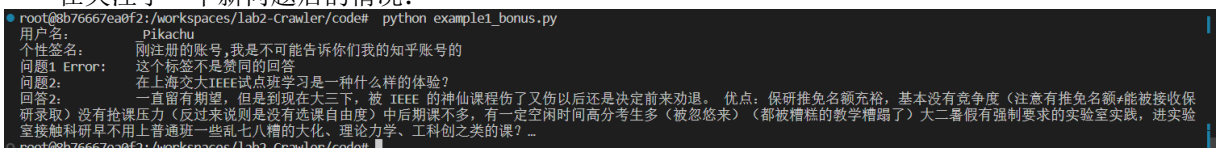
```
    print("问题：2  \t"+question2)
    print("回答：2  \t"+answer2)
except:
    print("问题2 Error:\t 这个标签不是赞同的回答")
```

4.2 结果展示

在前两个都是赞同的回答的情况下：



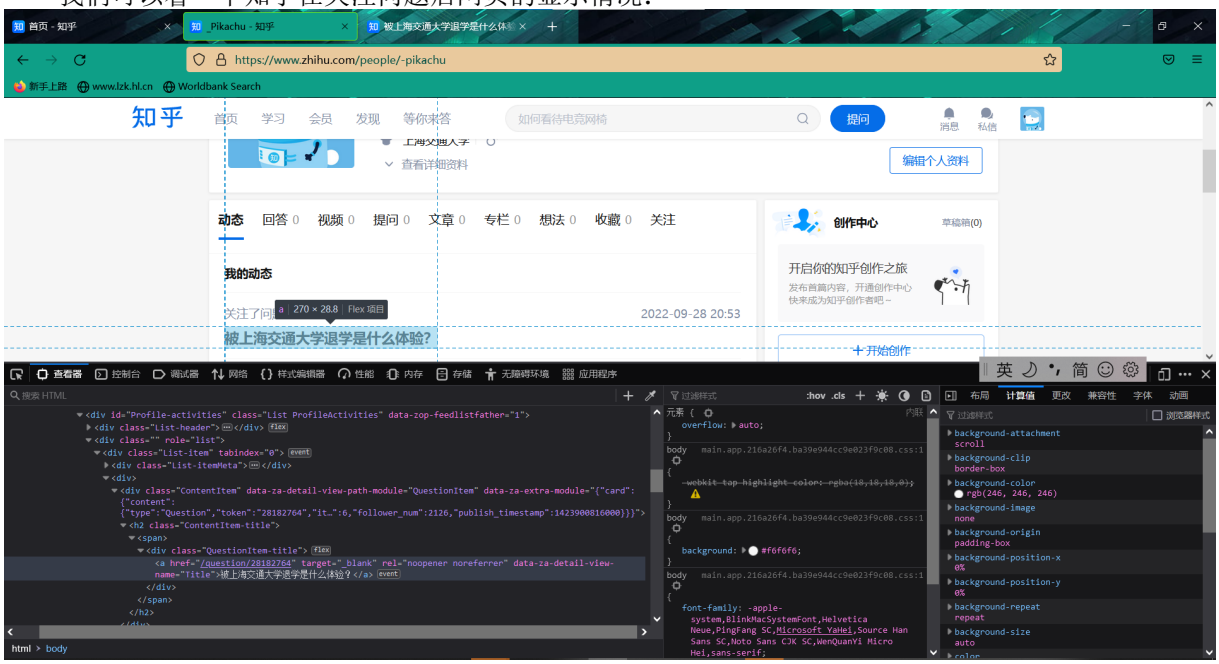
在关注了一个新问题后的情况：



4.3 问题讨论

4.3.1 为什么关注问题会报错

我们可以看一下知乎在关注问题后网页的显示情况：



我们可以很明显地发现，知乎关注的问题和赞同的回答的标签比较明显，这也是为什么问题 1 会无法找到。

但观察可以发现，每个动态实际处于一个相对独立的状态，因此不会影响问题 2。

4.3.2 BeautifulSoup 和 lxml 的比较

二者都是非常常用的网页提取工具。lxml 的优点在于可以准确提取某一位置的信息，不需遍历整个网页，效率较为可观。而 Soup 的优点在于可以模糊查找，避免出现像本次练习中报错的

情况出现。

5 练习 2 和练习 3

5.1 问题重述与代码说明

练习 2 需要我们完成 BFS 搜索。根据助教在 ppt 中给出的提示，我们只需要将每次爬到的链接放到队首即可。代码如下：

```
# 这里是第二题的答案
def union_bfs(a, b):
    for e in b:
        if e not in a:
            a.insert(0, e)
```

练习 3 需要我们完成图的完全构建。我们只需要在爬到外链之后将每个`outlinks` 放到图中即可，添加的代码如下：

```
    ...
    outlinks = get_all_links(content)
    graph[page] = outlinks #这里是第三题的答案
    globals()['union_%s' % method](tocrawl, outlinks)
    ...
```

5.2 结果展示

```
root@8b76667ea0f2:/workspaces/lab2-Crawler/code# python crawler_sample.py
graph_dfs: {'A': ['B', 'C', 'D'], 'D': ['G', 'H'], 'H': [], 'G': ['K', 'L'], 'L': [], 'K': [], 'C': [], 'B': ['E', 'F'], 'F': [], 'E': ['I', 'J'], 'J': [], 'I': [
]}
crawled_dfs: ['A', 'D', 'H', 'G', 'L', 'K', 'C', 'B', 'F', 'E', 'J', 'I']
graph_bfs: {'A': ['B', 'C', 'D'], 'B': ['E', 'F'], 'C': [], 'D': ['G', 'H'], 'E': ['I', 'J'], 'F': [], 'G': ['K', 'L'], 'H': [], 'I': [], 'J': [], 'K': [], 'L': [
]}
crawled_bfs: ['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K', 'L']
root@8b76667ea0f2:/workspaces/lab2-Crawler/code#
```

5.3 问题讨论

5.3.1 为什么本次实验结果和示例结果不完全一致

在 Python3.8 及更高版本中，`dict` 数据结构会根据输入的情况有序输出，而非是低版本的乱序输出。这也意味着，对于该图是严格倒序输出的。

5.3.2 DFS 和 BFS 图的区别

这一区别在 ppt 中已经较为明确的指出。对比二者的输出结果，虽然图的结构是一样的，但是从输出的顺序中可以看出 DFS 和 BFS 在爬取结构时的区别。

6 练习 4

6.1 问题重述与代码说明

该联系需要我们在练习 2 和练习 3 的基础上写出一个完整的爬虫功能。本次实验除练习 2 和练习 3 中已完成的部分外，仍需补充部分代码以实现功能。大部分功能在助教提供的代码模板中

业已给出，仅需补充部分函数即可。

6.1.1 get_all_links 函数

根据 ppt 中的提示，我们需要获得页面中的所有`href` 链接且可正常运行（因此不能爬到`javascript;;` 等的情况），故我们采用正则表达式的形式对页面进行爬取。对于子链接的情况，需要进行 urllink。

代码如下:

```
# 返回页面下所有链接
def get_all_links(content, page):
    # 如果爬不到会返回None
    if content == None:
        return []
    links = []
    soup = BeautifulSoup(content, features="lxml")
    for link in soup.findAll('a',{'href' : re.compile('^http|^/')}): #正则表达式返回链接
        mylink = link.get("href")
        if mylink[0] == "/":
            mylink = urllib.parse.urljoin(page,mylink) #连接链接
            links.append(mylink)
        else:
            links.append(mylink)
    return links
```

6.1.2 get_page 函数

其目的是要返回页面的内容，方法 Lab1 中已经讲过。但需要注意的是，考虑一些页面不一定能够完全打开，我们需要通过 try 的方法来规避这一情况的出现。

代码如下：

```
# 获取页面
def get_page(page):
    content = ''
    try:
        req = urllib.request.Request(page)
        req.add_header('User-Agent ', "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:105.
                                                    0) Gecko/20100101 Firefox/105.0")#添
                                                    加报头（本机Edge ）
        content = urllib.request.urlopen(req,timeout = 500).read()# 内连接上500ms
    except:
        return None
    return content
```

6.1.3 crawl 函数

对于 crawl 函数，我们首先需要对爬取的页面加以限制。如果被爬取的页面超过`max_page` 时，我们需要终止循环。同时，考虑图的结构，我们仍需要按照练习 3 的方法构造我们的图。最终的 crawl 函数如下所示：

```python
# 真正的主函数
def crawl(seed, method, max_page):
    tocrawl = [seed]
    crawled = []
    graph = {}
    count = 0

    while tocrawl:
        page = tocrawl.pop()
        if page not in crawled:
            print(page)
            # 这里是计数用
            count += 1
            if count > max_page:
                return graph, crawled
            content = get_page(page)
            add_page_to_folder(page, content)
            outlinks = get_all_links(content, page)
            graph[page] = outlinks
            globals()['union_%s' % method](tocrawl, outlinks)
            crawled.append(page)

    return graph, crawled
```

在此基础上，对原代码结构做微小的调整，以适应最终使用。

6.2 结果展示

为了保证代码不被污染，以及可复用，因此在`carwler.py` 并未提供任何可执行的代码。结果的展示可通过以下三种方式进行。

6.2.1 方法 1

直接运行`sjtu_crawler.py`，结果如下。

```
root@8b76667ea0f2:/workspaces/lab2-Crawler/code# python sjtu_crawler.py
https://www.sjtu.edu.cn
https://vs.sjtu.edu.cn/
https://vs.sjtu.edu.cn/jtdx/index/imagesList
https://vs.sjtu.edu.cn/jtdx/index/imagesList?themeId=51
https://vs.sjtu.edu.cn/jtdx/index/imagesList?themeId=41
https://vs.sjtu.edu.cn/jtdx/index/imagesList?themeId=49
https://vs.sjtu.edu.cn/jtdx/index/imagesList?themeId=45
https://vs.sjtu.edu.cn/jtdx/index/imagesList?themeId=54
https://vs.sjtu.edu.cn/jtdx/index/imagesList?themeId=40
https://vs.sjtu.edu.cn/jtdx/index/imagesList?themeId=42
https://vs.sjtu.edu.cn/jtdx/index/imagesList?themeId=37
↑DFS------我是分割线------BFS↓
https://www.sjtu.edu.cn
https://mp.weixin.qq.com/s/YpFEwb9KjOYzk4HeKWbnCw
https://mp.weixin.qq.com/s/W0gHJmoHRL-F9gHIvV7t0w
https://mp.weixin.qq.com/s/b41k3w0tHCYzOy_3r3LsNg
https://mp.weixin.qq.com/s/BKLtqr-XCkU2SPutJn1cBQ
https://news.sjtu.edu.cn/jdyw/20220927/174628.html
https://mp.weixin.qq.com/s/AtjtZb-u1AVqtrdO3P9xKQ
https://news.sjtu.edu.cn/jdyw/20220926/174584.html
https://news.sjtu.edu.cn/jdyw/20220928/174643.html
https://news.sjtu.edu.cn/jdyw/20220926/174548.html
https://news.sjtu.edu.cn/jdyw/20220926/174551.html
```

6.2.2 方法 2(不推荐)

调用 Python IDLE 进行编译。需要输入以下代码：

```
python
>>> from crawler import *
>>> crawl("www.sjtu.edu.cn", "bfs", 10)
```

结果如下（仅展示 DFS）：

```
root@8b76667ea0f2:/workspaces/lab2-Crawler/code# python
Python 3.8.5 (default, Sep 10 2020, 16:47:10)
[GCC 8.3.0] on linux
Type "help", "copyright", "credits" or "license" for more informatio
>>> from crawler import*
>>> crawl("www.sjtu.edu.cn", "dfs", 10)
www.sjtu.edu.cn
({'www.sjtu.edu.cn': []}, ['www.sjtu.edu.cn'])
>>> crawl("https://www.sjtu.edu.cn", "dfs", 10)
https://www.sjtu.edu.cn
https://vs.sjtu.edu.cn/
https://vs.sjtu.edu.cn/jtdx/index/imagesList
https://vs.sjtu.edu.cn/jtdx/index/imagesList?themeId=51
https://vs.sjtu.edu.cn/jtdx/index/imagesList?themeId=41
https://vs.sjtu.edu.cn/jtdx/index/imagesList?themeId=49
https://vs.sjtu.edu.cn/jtdx/index/imagesList?themeId=45
https://vs.sjtu.edu.cn/jtdx/index/imagesList?themeId=54
https://vs.sjtu.edu.cn/jtdx/index/imagesList?themeId=40
https://vs.sjtu.edu.cn/jtdx/index/imagesList?themeId=42
https://vs.sjtu.edu.cn/jtdx/index/imagesList?themeId=37
```

6.2.3 方法 3

该方法只需要在系统终端直接输入参数即可，结果如下：

```
root@8b76667ea0f2:/workspaces/lab2-Crawler/code# python crawler.py https://www.bilibili.com dfs 10
https://www.bilibili.com
https://www.bilibili.com/festival/2021bnj
https://www.bilibili.com/blackboard/activity-Uijsrg4dP.html
https://www.bilibili.com/h5/mall/suit/detail?navhide=1&id=4019&from=bnj
https://space.bilibili.com/1868902080
https://www.bilibili.com/cheese/
https://www.bilibili.com/v/customer-service
https://www.bilibili.com/list/recommend/1.html
https://www.bilibili.com/video/online.html
https://www.bilibili.com/video/BV1rd4y167kC
https://www.bilibili.com/video/BV14B4y177Es
root@8b76667ea0f2:/workspaces/lab2-Crawler/code# python crawler.py https://www.bilibili.com bfs 10
https://www.bilibili.com
https://www.bilibili.com/anime/
https://game.bilibili.com/platform/
https://live.bilibili.com
https://show.bilibili.com/platform/home.html?msource=pc_web
https://manga.bilibili.com?from=bill_top_mnav
https://www.bilibili.com/match/home/
https://app.bilibili.com
https://www.bilibili.com/
https://t.bilibili.com
https://www.bilibili.com/v/popular/all
```

6.3 问题讨论

6.3.1 关于直接调用 Python IDLE 的问题

由于 Python IDLE 在调用函数时会一并输出函数结果，而我们返回图的结构的方法时会直接将能爬到的所有链接塞在图里，这样也就导致了会同时输出很长的图结构（并未体现在报告中），

这种情况应当极力避免。

6.3.2 try 的作用

使用 try 的作用为：如果 try 下的语句不报错，则正常运行。而如果 try 下面的语句报错，则执行 except 下的语句。这种语法的优点在于避免因为报错而无法正常运行，同时也提供了补救措施。