3.2 网站图的爬取路径

从前一节看到深度优先与广度优先方法都是遍历树的一种方法，但是网站的各个网页之间的关系未必是树的结构，它们可能组成一个复杂的图形结构，即有回路。如果在前面的网站中每个网页都加一条Home的语句，让每个网页都能回到主界面，那么网站的关系就是一个有回路的图。

3.2.1 复杂的 Web 网站

(1) books.htm

```
<h3>计算机</h3>
<ul>
<li><a href="database.htm">数据库</a></li>
<li><a href="program.htm">程序设计</a></li>
<li><a href="network.htm">计算机网络</a></li>
</ul>
```

(2) database.htm

```
<h3>数据库</h3>
<ul>
<li><a href="mysql.htm">MySQL 数据库</a></li>
</ul>
<a href="books.htm">Home</a>
```

(3) program.htm

```
<h3>程序设计</h3>
<ul>
<li><a href="python.htm">Python 程序设计</a></li>
<li><a href="java.htm">Java 程序设计</a></li>
</ul>
<a href="books.htm">Home</a>
```

(4) network.htm

```
<h3>计算机网络</h3>
<a href="books.htm">Home</a>
```

(5) mysql.htm

```
<h3>MySQL 数据库</h3>
<a href="books.htm">Home</a>
```

(6) python.htm

```
<h3>Python 程序设计</h3>
<a href="books.htm">Home</a>
```

(7) java.htm

```
<h3>Java 程序设计</h3>
<a href="books.htm">Home</a>
```

这是深度优先与广度优先方法要做一点改进，我们用一个 Python 中的列表 urls 来记住已经访问过的网站，如果一个网址 url 没有访问过就访问它，并把 url 加到 urls 中保存起来，如果 url 已经访问过就不再访问了，这样就可以避免形成回路，导致无限循环。

3.2.2 改进深度优先客户端程序

假设给定图 G 的初态是所有顶点均未曾访问过。在 G 中任选一顶点 v 为初始出发点(源点)，则深度优先遍历可定义如下：首先访问出发点 v，并将其标记为已访问过；然后依次从 v 出发搜索 v 的每个邻接点 w。若 w 未曾访问过，则以 w 为新的出发点继续进行深度优先遍历，直至图中所有和源点 v 有路径相通的顶点(亦称为从源点可达的顶点)均已被访问为止。

图的深度优先遍历类似于树的前序遍历。采用的搜索方法的特点是尽可能先对纵深方向进行搜索。这种搜索方法称为深度优先搜索(Depth-First Search)。相应地，用此方法遍历图就很自然地称之为图的深度优先遍历，基本实现思想：

（1）访问顶点 v；

（2）从 v 的未被访问的邻接点中选取一个顶点 w，从 w 出发进行深度优先遍历；

（3）重复上述两步，直至图中所有和 v 有路径相通的顶点都被访问到。

1、使用递归的程序：

```
from bs4 import BeautifulSoup
import urllib.request

def spider(url):
    global urls
    if url not in urls:
        urls.append(url)
        try:
            data=urllib.request.urlopen(url)
            data=data.read()
            data=data.decode()
            soup=BeautifulSoup(data,"lxml")
            print(soup.find("h3").text)
            links=soup.select("a")
            for link in links:
                href=link["href"]
                url=start_url+"/"+href
                spider(url)
        except Exception as err:
            print(err)

start_url="http://127.0.0.1:5000"
```

```
urls=[]
spider(start_url)
print("The End")
```

2、使用栈的程序

```
from bs4 import BeautifulSoup
import urllib.request

class Stack:
    def __init__(self):
        self.st=[]
    def pop(self):
        return self.st.pop()
    def push(self,obj):
        self.st.append(obj)
    def empty(self):
        return len(self.st)==0

def spider(url):
    global urls
    stack=Stack()
    stack.push(url)
    while not stack.empty():
        url=stack.pop()
        if url not in urls:
            urls.append(url)
            try:
                data=urllib.request.urlopen(url)
                data=data.read()
                data=data.decode()
                soup=BeautifulSoup(data,"lxml")
                print(soup.find("h3").text)
                links=soup.select("a")
                for i in range(len(links)-1,-1,-1):
                    href=links[i]["href"]
                    url=start_url+"/"+href
                    stack.push(url)
            except Exception as err:
                print(err)

start_url="http://127.0.0.1:5000"
urls=[]
spider(start_url)
print("The End")
```

这两个程序的结果都一样：
计算机
数据库
MySQL 数据库
计算机
程序设计
Python 程序设计
Java 程序设计
计算机网络
The End

3.2.3 改进广度优先客户端程序

图的广度优先遍历 BFS 算法是一个分层搜索的过程，和树的层序遍历算法类同，它也需要一个队列以保持遍历过的顶点顺序，以便按出队的顺序再去访问这些顶点的邻接顶点。基本实现思想：

（1）顶点 v 入队列。
（2）当队列非空时则继续执行，否则算法结束。
（3）出队列取得队头顶点 v；访问顶点 v 并标记顶点 v 已被访问。
（4）查找顶点 v 的第一个邻接顶点 col。
（5）若 v 的邻接顶点 col 未被访问过的，则 col 入队列。
（6）继续查找顶点 v 的另一个新的邻接顶点 col，转到步骤（5）。直到顶点 v 的所有未被访问过的邻接点处理完。转到步骤（2）。

广度优先遍历图是以顶点 v 为起始点，由近至远，依次访问和 v 有路径相通而且路径长度为 1，2，……的顶点。为了使“先被访问顶点的邻接点”先于“后被访问顶点的邻接点”被访问，需设置队列存储访问的顶点。

```
from bs4 import BeautifulSoup
import urllib.request

class Queue:
    def __init__(self):
        self.st=[]
    def fetch(self):
        return self.st.pop(0)
    def enter(self,obj):
        self.st.append(obj)
    def empty(self):
        return len(self.st)==0

def spider(url):
    global urls
    queue=Queue()
    queue.enter(url)
```

```
        while not queue.empty():
            url=queue.fetch()
            if url not in urls:
                try:
                    urls.append(url)
                    data=urllib.request.urlopen(url)
                    data=data.read()
                    data=data.decode()
                    soup=BeautifulSoup(data,"lxml")
                    print(soup.find("h3").text)
                    links=soup.select("a")
                    for link in links:
                        href=link["href"]
                        url=start_url+"/"+href
                        queue.enter(url)
                except Exception as err:
                    print(err)

start_url="http://127.0.0.1:5000"
urls=[]
spider(start_url)
print("The End")
```

程序结果：

```
计算机
数据库
程序设计
计算机网络
MySQL 数据库
计算机
Python 程序设计
Java 程序设计
The End
```