GCRAWLER: GRAPHICAL WEB CRAWLER

The GCrawler project is a graphical web crawler which crawls a user-supplied website, following links on each page as it goes. It displays a color-coded graph of what pages were crawled and how it reached each page. Try it out here: GCrawler App

Website

The Website is responsible for:

1. Serving the user with the search form
2. Tracking the user's recent searches
3. Passing input to the Data Transfer program
4. Displaying the results of a search

```
router.post('/', function(req, res){

    r = {};
    var today = new Date();
    var dd = String(today.getDate()).padStart(2, '0');
    var mm = String(today.getMonth() + 1).padStart(2, '0');
    var yyyy = today.getFullYear();
    today = mm + '/' + dd + '/' + yyyy;

    var link = req.body.link;
    var search_type = req.body.search_type;
    var max = req.body.max;
    var keyword = req.body.keyword;

    if(!link || !search_type || !max)
    {
        r.error = "Invalid search...please try again!";
        r.jsscripts = ["static_search.js"];
        r.styles = ["search.css"];
        res.render('search', r);
        return;
    }
```

```
"""
Function Name: main
Description: The main controlling program of the crawler.
Inputs: takes a string representing the starting URL, a char flag for type of
    search, an integer representing the link limit and an optional keyword for
    halting the search
Outputs: Tree in JSON format
"""
def main(url, search_type, link_limit, keyword = None):
    #branch based on DFS or BFS (Use different functions for each, general flow below)
    if search_type == "dfs":
        tree, robots_flag = cu.depth_search(url, link_limit, keyword)
    elif search_type == "bfs":
        tree, robots_flag = cu.breadth_search(url, link_limit, keyword)
    #bad parameter
    else:
        return 1
    #log the resulting output
    string_list = []
    for node in tree:
        string_list.append(node.__str__())

    #check robots flag
    if robots_flag:
        print("robots", end = "")

    print('[' + ','.join(string_list) + ']')
    return 0
```

Data Transfer

The Data Transfer program is responsible for:

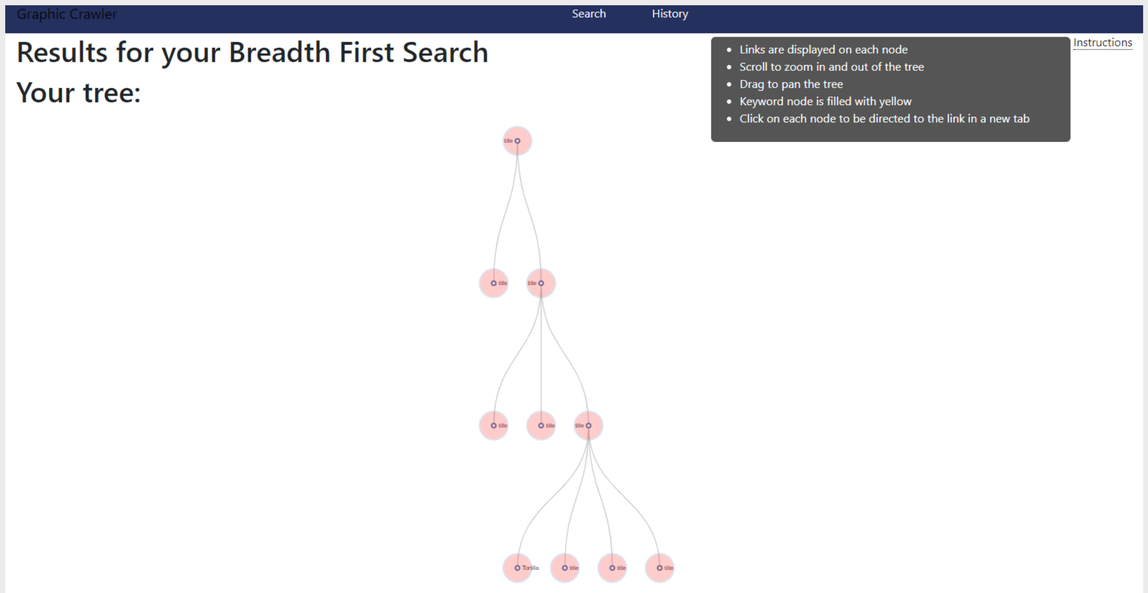
1. Receiving input from the Website
2. Passing the input to the Crawler
3. Receiving the output from the Crawler
4. Passing the output to the Website for rendering

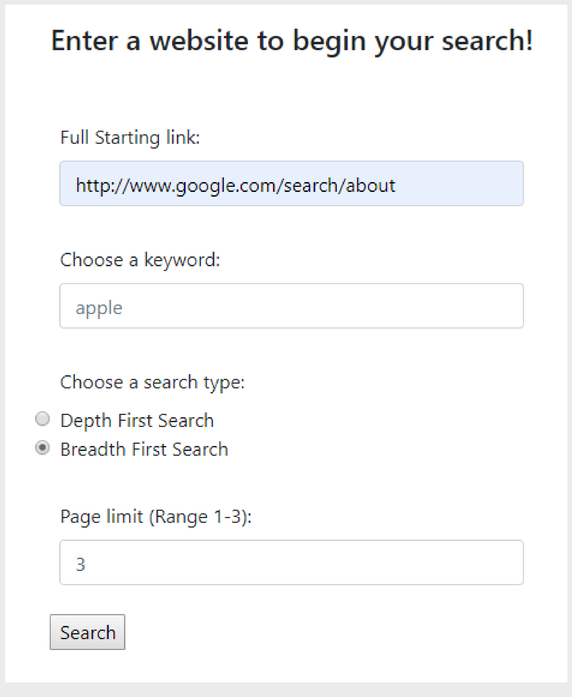
```
class Node {
    constructor(link) {
        this.name = link;
        this.children = [];
        this.parent = "";
    }
}


function makeLinkTree(jsonArray) {
    let dict = {};
    let root;

    jsonArray.forEach(function (urlObject) {
        let newNode;
        if (urlObject.URL in dict) {
            newNode = dict[urlObject.URL];
        }
        else {
            newNode = new Node(urlObject.URL);
            newNode.title = urlObject.Title;
            newNode.KeywordFound = urlObject.KeywordFound;
            dict[urlObject.URL] = newNode;
        }
        if (urlObject.Parent === 'null') {
            root = newNode;
        }
        else {
            let parentNode;
            if (!(urlObject.Parent in dict)) {
                parentNode = new Node(urlObject.Parent);
                dict[urlObject.Parent] = parentNode;
            }
            else {
                parentNode = dict[urlObject.Parent];
            }

            parentNode.children.push(newNode);

        }
```

Crawler

The Crawler is responsible for:

1. Receiving input from the Data Transfer program
2. Crawling each page from the starting link
3. Passing the formatted results to the Data Transfer program for further processing





GCRAWLER FEATURES:

- **Intuitive:** Complete a simple form to crawl any website. Supply a keyword to search for text on each page. GCrawler will halt when it finds the keyword. (*left*).
- **Retentive:** GCrawler stores your sessions' past searches on the History tab for easy re-use.
- **Flexible:** Choose from Depth First Search or Breadth First Search methods. Depth First Search will follow a random link on each page until the specified page limit (1-10) is reached. Breadth First Search will follow all links on each page until the specified depth limit (1-3) is reached. (*left*)
- **Informative:** GCrawler's color-coded results display the title of each page on the node. The URL can be seen by hovering over the node. A user can click the node to open the page. (*above*)
- **Good Bot:** GCrawler adheres to the Robot Exclusion Protocol by reading a page's robots.txt file and abiding by its rules. It also evaluates pages and links for other indicators that it should not crawl the page or follow the links.

Oregon State University

CONNECT WITH THE GCRAWLERS:

Christopher Beall (Data Transfer): https://github.com/beallch

Helen Jiang (UI/Website): https://github.com/hyjiang7

Brian Metzger (Crawler): https://github.com/metzgerb