REPORT

- Following is the project report for the Distributed Web Crawler
    - Created by Team Number 15

Demo for the project is available at : https://www.youtube.com/watch?v=CVS8KFEYYPA

ASSUMPTIONS FOR THE CODE:

1. we assume 4 worker nodes/clients which run on ports immediately succeeding port 5000.(5001, 5002, 5003, 5004)
    - These said ports will have to entered as arguments when the nodes are run.
2. The number of worker nodes that we are starting with can be changed, easily by making a trivial change in the code.
    - (A tutorial for the same shall be given in the demo submission).
3. We assume non failure of nodes.
4. The graph/Tree is in the form of list of edges, i.e. each element is (a,b) such that a, b are the urls of the graph, which represent the URL's of websites, and (a,b) is the edge connecting these vertices.
5. show_graph shall only work for those URLs that have been scanned/crawled at some stage.

Steps to run the code

CLIENT NODES

1. Set up 4 client nodes by running

```
python3 node.py  <port_number>
```

on four different terminal screens.

WORKER NODES

1. Setting up worker client requires you to run the following

```
python3 main.py
```

Running this file will take you to a command line structure. You will have to send commands in real time to see outputs
Following are the formats for the commands

```
-> Command : crawl
-> Enter Website : <website>
-> Enter Level : <level>

-> command : make_graph
-> Enter Website : <website>
-> Enter Depth : <depth>

-> command : show_html
-> Enter Website : <website>

-> command : End ( This will close the main.py file
```

PS : Note that closing the main.py will NOT lead to loosing the data that we got from crawling. That will persist for as long as the nodes are running

Code Architecture:

- The code is divided into the following portions

- A main.py file. This file take in input, assigns functions as per requirements and sends the requests.
- A node.py file. This file is the main home for all the code that is run in our web Crawler. The crawling and the graph making, all happens through this code
- A crawler.py file. This is the file that crawls website as per demand. Html storage also happens through this file only
- Utility functions : We have some extra files that contain some trivial utility functions

Our system is entirely distributed in *both* crawling and retrieving data.
Our main.py redirects commands to the Flask nodes, which are instances of node.py. Each node only crawls one level at a time and then distributes the children links among other clients and this happens recursively.
Even the graph is formed in a similar manner