Project 2 - Simplified PageRank

Due Nov 14 by 11:59pm **Points** 100 **Submitting** an external tool
Available until Nov 18 at 11:59pm

Simplified PageRank

Problem Statement

In this project you will be programming a simplified version of the Google's PageRank algorithm using an Adjacency List or equivalent implementation of the web graph. The project description handout can be found here: **Project2_SimplifiedPageRank.pdf (https://ufl.instructure.com/courses/489296/files/79990741?wrap=1) (https://ufl.instructure.com/courses/489296/files/79990741/download?download_frd=1)**

Testing

- Test your code on Gradescope before submitting your implementation. You have five available test cases and you can submit any number of times.
- Create your own tests and test as much as possible.
- **We will stick to the input format.** No need to test for off-input statements such as inputting one URL instead of two in a line or testing whether a URL is valid or not.

Grading

- **Implementation [75 points]**
    - **You are supposed to implement an Adjacency List** data structure to represent the graph. **Failure to implement this will incur a 25 points deduction.**
    - Your code will be tested on 15 test cases each worth 5 points:
        - 5 publicly available test cases.
        - 10 test cases that will be added by the course staff and are not shown to the students.
    - **Documentation [15 Points]**
        - Submit a document addressing all these prompts:
            - Describe the data structure you used to implement the graph and why? [2.5 points]

        - What is the computational complexity of each method in your implementation in the worst case in terms of Big O notation? [5 points]
        - What is the computational complexity of your main method in your implementation in the worst case in terms of Big O notation? [5 points]
        - What did you learn from this assignment and what would you do differently if you had to start over? [2.5 points]
    - **Code Style and Design [10 Points]**
        - 5 points for design decisions, a well-designed Graph API, and code modularity
        - 2 points for appropriate comments
        - 1 point for white spaces
        - 2 points for consistent naming conventions
- **Catch Tests Bonus [5 Points]**
    - You can score 5 bonus points if you submit a separate file containing 5 test cases (1 point/test) using the Catch Framework. These tests should be different than the public test cases. Your score is however capped to 100 points for this project. This means that if you pass 14 tests and submit bonus test files, you will get a 100 provided you score full points on the documentation. Also, if you pass 15 tests and score 23 on documentation and design, + 5 points on bonus, you will still score 100 points [your score is 103 but is capped to 100 in this case].

Submission

- One or more .cpp or.h files that have your implementation. **Test on gradescope early and often.**
- One pdf file that has your documentation. **Upper limit – 3 pages** with 20% deduction in report per extra page. Cover page is not required, just your name on Page 1 is suffice.

Frequently Asked Questions

- The course staff will maintain an active FAQ Google document to answer your questions. Post your questions in Slack, but we will answer in this document and send you the link. The link to the document is: **https://docs.google.com/document/d/1a9hR1Ep2IYK-MnsXwl2VxyotO1Yd-XCC8NWUkdp24JM/** **(https://docs.google.com/document/d/1a9hR1Ep2IYK-MnsXwl2VxyotO1Yd-XCC8NWUkdp24JM/edit#)**

Additional Optional Resources