CS 470 Homework 4

Due by Thursday, March 30, 2023 at 6:00 PM

Submission instructions

Submit your assignment through the QTest system, using exam ID: **CS470.hw4**. Upload a single ZIP archive file named **hw4.zip**, containing all the files of your solution:

1. The program's source files.
2. A `README.txt` file explaining how to compile and run your program.
3. The input files, named "graph1.dot" ... "graph5.dot".
4. The output files, named "pagerank1.csv" ... "pagerank5.csv".
5. The report in PDF format.
6. The LaTeX source files used to typeset the report.

No email submissions are accepted. No late submissions are accepted.

At the top of your solution, include a section named "Collaboration statement" in which you acknowledge any collaboration, help, or resource you used or consulted to complete this assignment.

1 Graph Test Cases (15 points)

Create at least five directed graphs that you will use to test the algorithm that you will implement later in this assignment. Name these files "graph1.dot" ... "graph5.dot".

graph1.dot: A graph with 5 vertices, with no dead ends nor spider traps.

graph2.dot: A graph with 10 vertices, with no dead ends nor spider traps.

graph3.dot: A graph with 10 vertices, with one dead end and no spider traps.

graph4.dot: A graph with 10 vertices, with one spider trap and no dead ends.

graph5.dot: A graph with 50 vertices, which may or may not contain dead ends and/or spider traps.

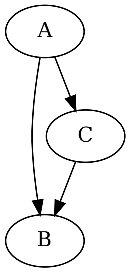
Your test cases should be in the following format:

- The first line should be: `digraph {`
- Each following line should represent a directed edge such as `V1 -> V2` where V1 and V2 are names of vertices.
- The last line should be: }

Example:

```
digraph {
A -> B
A -> C
C -> B
}
```

The example above corresponds to this graph:



The format described above can be visualized using the software Graphviz (https://www.graphviz.org). The basic syntax to run Graphviz from a command line terminal is the following:

```
dot -T pdf input_file.dot -o output_file.pdf
```

2 PageRank (55 points)

Using either Java or Python, implement the PageRank algorithm.

- The program should be executable with at least 2 parameters: the name of the input graph file (in the format described before), and the name of the output file.
- The first line of the output file should be the header: `vertex,pagerank`

- Each following line of the output file should correspond to a vertex in the input graph. Each line should contain exactly two items, separated by a comma: the name of the vertex, and the PageRank value of that vertex.
- The output file should be sorted by descending (highest to lowest) value of PageRank. In case of ties in PageRank value, lines are then sorted by ascending (lowest to highest) vertex name.
- Test your program on the graphs you created before, and on any other additional graphs you may want to experiment with. Name your output files "pagerank1.csv" ... "'pagerank5.csv".

3 Report (30 points)

Write a report in LaTeX to present your results. Describe the test cases you created, including pictures generated using Graphviz. Describe your implementation, and how you dealt with dead-ends and spider traps. Discuss your results, the insights, experiences and lessons you have learned from this assignment.

Grading criteria

- 3 points for each well-written test case, up to 15 points total.
- 55 points for correct and complete implementation of PageRank. -5 for minor mistakes; -10 if there is some mistake but the program still works in most cases; -20 for more serious mistakes but the program still works in several cases. Zero points if the program does not compile, or if the program compiles but gives mostly the wrong results or crashes.
- 30 points for a complete, clear, and well organized report. Zero points if the report is not typeset using LaTeX, or if the LaTeX source code is not provided.
- -10 points for insufficient comments in the code.
- -10 points for each deviation from the submission instructions.
- -10 points for missing collaboration statement.