Project Readme Template

Version 1 9/11/24

A single copy of this template should be filled out and submitted with each project submission, regardless of the number of students on the team. It should have the name readme_"team name"

Also change the title of this template to "Project x Readme Team xxx"

<table>
<tr><td>1</td><td>Team Name: Kwilli33</td></tr>
<tr><td>2</td><td>Team members names and netids: Kwilli33</td></tr>
<tr><td>3</td><td>Overall project attempted, with sub-projects: Given a graph, check if there's a hamiltonian path</td></tr>
<tr><td>4</td><td>Overall success of the project: Reasonably successful</td></tr>
<tr><td>5</td><td>Approximately total time (in hours) to complete: 16 hrs</td></tr>
<tr><td>6</td><td>Link to github repository: https://github.com/kw-nd/toc-project01</td></tr>
<tr><td>7</td><td>List of included files (if you have many files of a certain type, such as test files of different sizes, list just the folder): (Add more rows as necessary). Add more rows as necessary.
<table>
<tr><td>File/folder Name</td><td>File Contents and Use</td></tr>
<tr><td colspan="2">Code Files</td></tr>
<tr><td>hamiltonian_path_kwilli33.py</td><td>Python code checks if a Hamiltonian path exists in a given graph. It also plots execution time vs. graph size on a scatter plot using data from a CSV file.</td></tr>
<tr><td colspan="2">Test Files</td></tr>
<tr><td>check_kwilli33.py
data_kwilli33.cnf.csv</td><td>(1) Python code checks the program's accuracy by providing graphs of various sizes with expected outputs. If the output matches, the program is accurate.
(2) A CSV file is used as input for testing in</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><td></td><td>
<table>
<tr><td></td><td>hamiltonian_path_kwilli33.py and for creating the scatter plot.</td></tr>
<tr><td colspan="2">Output Files</td></tr>
<tr><td>output_kwilli33.png</td><td>Image of the output from check_kwilli33.py and its test cases.</td></tr>
<tr><td colspan="2">Plots (as needed)</td></tr>
<tr><td>plots_kwilli33.png</td><td>Scatter plot of time vs. size, using data from data_kwilli33.cnf.csv, showing how long it takes to find a Hamiltonian path.</td></tr>
</table>
</td></tr>
<tr><td>8</td><td>Programming languages used, and associated libraries:
- Language: Python
- Libraries: time, csv, numpy, matplotlib.pyplot, itertools (permutations)</td></tr>
<tr><td>9</td><td>Key data structures (for each sub-project):
- set, list, dictionary, tuple (hamiltonian_path_kwilli3.py)
- dictionary, list, tuple (check_kwilli33.py)</td></tr>
<tr><td>10</td><td>General operation of code (for each subproject)

1. Reads graphs from a CSV file, brute-forces permutations of vertices to find Hamiltonian paths, measures execution time, and plots time vs. graph size. It fits an exponential curve to the data and saves the plot as a PNG file. (hamiltonian_path_kwilli33.py)
2. Defines test cases for different graphs, runs Hamiltonian path checks on each graph using the hamiltonian_path_kwilli33.test_hamiltonian function, and compares the results to the expected outcomes. It prints whether a Hamiltonian path was found, the expected result, and the execution time for each test case. (check_kwilli33.py)</td></tr>
<tr><td>11</td><td>What test cases you used/added, why you used them, what did they tell you about the correctness of your code.

The test cases include both connected graphs (with Hamiltonian paths) and disconnected graphs (without paths), ranging from 4 to 12 vertices. They test the code’s ability to handle different sizes and structures, ensuring it correctly identifies whether a Hamiltonian path exists.</td></tr>
<tr><td>12</td><td>How you managed the code development:</td></tr>
</table>

	The code development was split into two parts: - **hamiltonian_path_kwilli33.py**: This main Python program reads graphs, generates permutations of vertices, and checks for Hamiltonian paths. It includes functions for parsing graph data from CSV files, validating paths, and timing execution to measure performance. - **check_kwilli33.py**: This program runs predefined test cases on the Hamiltonian path checker, using both connected and disconnected graphs of various sizes. It validates the correctness of the main code and tracks execution times, ensuring the algorithm works across different graph structures and sizes efficiently.
13	Detailed discussion of results: **output_kwilli33.png:** Correctly identifies whether a Hamiltonian path exists in both connected and disconnected graphs. For connected graphs, paths are found as expected, and for disconnected graphs, paths are not found, which is also expected. Additionally, the execution times for each test vary, with smaller graphs (e.g., 4-vertex and 6-vertex) completing quickly, while larger graphs, especially disconnected ones (like the 12-vertex graph), take exponentially longer to process, consistent with the brute force approach used to check all permutations. **plots_kwilli33.png:** The plot of time versus graph size highlights the exponential growth in execution time as the number of vertices increases. This exponential behavior is expected due to the factorial time complexity of the brute force permutation search for Hamiltonian paths.
14	How team was organized: There was no team involved. I worked independently, handling all the programming, data management, and development tasks myself.
15	What you might do differently if you did the project again: If I did the project again, I would use CNF files instead of CSV files. Initially, I wasn't aware of the instructor provided Hamiltonian path data, so I built my code around CSV input. Testing with 2SAT.csv caused long runtimes, leading me to modify the data for faster execution. Next time, I'd align the code with CNF files from the start.
16	Any additional material: N/A