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_"teamname"

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

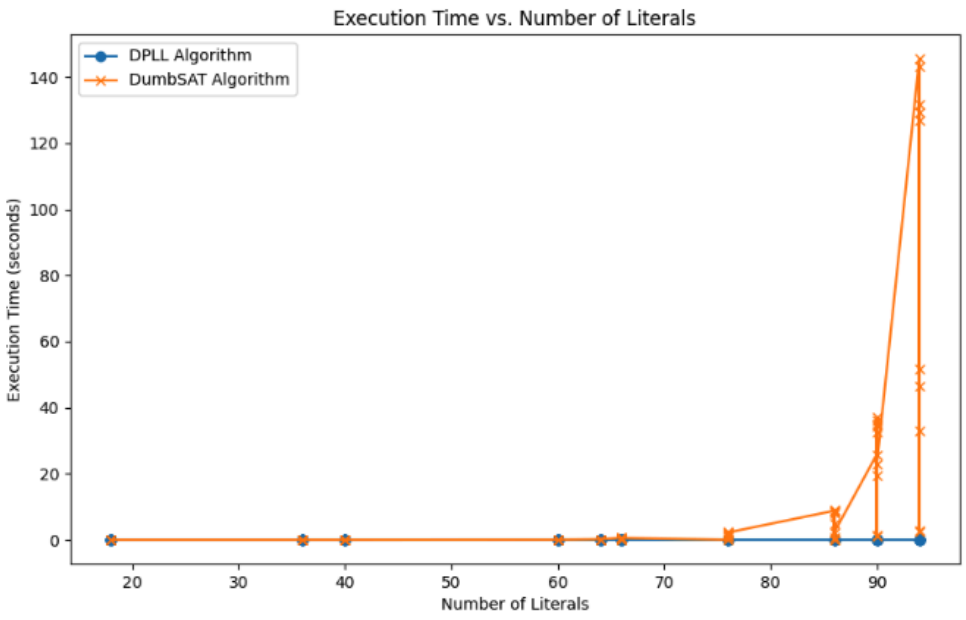
<table>
<tr><td>1</td><td>Team Name: Brown</td></tr>
<tr><td>2</td><td>Team members names and netids: Declan Brown - dbrown39</td></tr>
<tr><td>3</td><td>Overall project attempted, with sub-projects: 2SAT DPLL Algorithm</td></tr>
<tr><td>4</td><td>Overall success of the project: Successful</td></tr>
<tr><td>5</td><td>Approximately total time (in hours) to complete: 8 hours</td></tr>
<tr><td>6</td><td>Link to github repository: https://github.com/dbrown39nd/Brown-2SAT-DPLL</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.
<table>
<tr><th>File/folder Name</th><th>File Contents and Use</th></tr>
<tr><td colspan="2">Code Files</td></tr>
<tr><td>DPLL_Brown.py</td><td>This file contains the DPLL algorithm, a two helper functions that deal with formatting output</td></tr>
<tr><td>DumbSAT_Brown.py</td><td>This file was given by Prof. I cut it down to just the dumb_sat algorithm (formerly check()) and changed it to return a formatted assignment list).</td></tr>
<tr><td colspan="2">Test Files</td></tr>
<tr><td>verify_Brown.py</td><td>verify.py has a generator that yields expressions read from the 2SAT.cnf.csv file provided by Prof. Then, it runs both the Dumsat and DPLL algorithms on it and checks the output and execution time. This file outputs to output.txt and trace.txt, as well as generating the execution_time graph.</td></tr>
<tr><td colspan="2">Output Files</td></tr>
</table>
</td></tr>
</table>

	log_Brown.txt	This file contains the results of all 100 WFFS from the 2SAT.cnf.csv. This file Summarizes the outcome (50 WFF's SAT, 50 WFF's UNSAT, 100 WFFS with matching DumbSAT and DPLL results…). It also has the Success / Failure and SAT / UNSAT for each WFF. Success is True if both DumbSAT and DPLL output the same result (sat/unsat).
	trace_Brown.txt	This file contains the wff, DPLL sat/unsat, DumbSAT sat/unsat, assignments to variables if Satisfiable, and the execution time of each algorith.
	Plots (as needed)	
	plots_Brown.png	Graph of Execution Time vs. Number of literals for DumbSAT and DPLL. Exponential growth in time observed for DumbSAT.

- ☑ DPLL_Brown.py
- ☑ DumbSAT_Brown.py
    i. Modified the given DumbSAT solver.
- ☑ verify_Brown.py
    i. This file verifies that DPLL matches DumbSAT
- ☑ data_Brown.csv
    i. This is the 2SAT.cnf.csv file provided
- ☑ readme_Brown.pdf
- ☑ log_Brown.txt
    i. Summary of results with SAT/UNSAT for every wff. Basically a log of the console as its running
- ☑ trace_Brown.txt
    i. This file contains the expression for every wff, SAT/UNSAT, and assignments for both DPLL and DumbSAT if SAT.
- ☑ plots_Brown.png
    i. Exponential plot of execution time vs. number of literals for DPLL and DumbSAT.

8	Programming languages used, and associated libraries: **Python language with the following libraries random, time, matplotlib, pandas, and tqdm.**
9	Key data structures (for each sub-project): **Only 2D arrays for the expression, where each expression is a lists of clauses, and each clause is a list. Each clause has string literals.**

10	General operation of code (for each subproject) **The verify.py file acts as the driver for my project. It imports my implementation of the dpll algorithm and the given DumbSAT algorithm. It loops through the wffs inputted from the provided 2SAT.cnf.csv file and executes both algorithms on each input. I then log the success (sat/unsat), assignments and execution time, and check to make sure the DPLL has the same success as the DumbSAT. In my project, every test case of the 100 test cases had matching sat/unsat outputs. Then I plot the results after the 100 test cases.**
11	What test cases you used/added, why you used them, what did they tell you about the correctness of your code. **I used the provided 2SAT.cnf.csv file, which includes 100 wffs. After running my code on these inputs, I observed the exponential growth in time to execute with an increasing number of literals. I also verified that the DPLL output (sat/unsat) matched the DumbSAT output for every wff.**
12	How you managed the code development: **First I researched what the DPLL algorithm is, and how it works. Then I implemented and tested it against the 2SAT.cnf.csv. I did some debugging and added assignment tracking to the algorithm which was tricky. After I was confident I had seemingly valid outputs, I tested it against DumbSAT, worked out a few more small bugs, then every output matched. I then added matplotlib to make the plot and added nice formatting to output files. Formatting the output files took substantial time.**
13	Detailed discussion of results: **Of the 100 test cases from 2SAT.cnf.csv, 50 wffs were Satisfiable and 50 wffs were satisfiable. The DPLL and Dumbsat algorithm both agree on sat/unsat for every wff, which gives me confidence that the DPLL algorithm functions properly. The algorithm took around 15 minutes to execute, I have a pretty fast computer. I was not surprised that the DumbSAT algorithm took exponentially longer with the more literals. That is very logical. I was surprised that the DPLL algorithm performed very well for 2SAT.**

	
14	How team was organized: **N/A (Individual)**
15	What you might do differently if you did the project again: **I approached this project in an effective way. First I researched what the DPLL algorithm is, and how it works. Then I implemented and tested it against the 2SAT.cnf.csv. I did some debugging and added assignment tracking to the algorithm which was tricky. After I was confident**
16	Any additional material: If you need to run the code yourself, do the following. > git pull git@github.com:dbrown39nd/Brown-2SAT-DPLL.git > python3 -m pip install --upgrade pip > python3 -m venv env > source env/bin/activate > pip install requirements.txt > caffeinate -i python3 verify.py