Project 1 Readme Team hlee26

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"

1	Team Name: hlee26
2	Team members names and netids: Henry Lee, hlee26
3	Overall project attempted, with sub-projects: Implementing a 2SAT solver
4	Overall success of the project: Successful at getting the code to work however, had trouble with the dpll algorithm executing in fast time. The graph showed worst time case complexity with a graph looking very exponential.
5	Approximately total time (in hours) to complete: 8
6	Link to github repository:
7	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.

File/folder Name	File Contents and Use
Code Files	
2sat_solver_hlee26.py	Implements dpll, solves 2SAT problems by reading in problems from a csv file.
Test Files	
data-hlee26.csv check-hlee26.csv	Data used for graphing time and number of variables. Check used for checking correctness of 2SAT solver.
Output Files	
Output	Results of test cases 1-100 and whether or not they are satisfiable

	or not.	
	Plots (as needed)	
	2SAT_plot	Plot of time on the y axis and number of variables on the x axis.

8	Programming languages used, and associated libraries: Python using time, matplotlib, and google colab files libraries.
9	Key data structures (for each sub-project): Implemented a list of lists to store cnf clauses, dictionary to track variable assignments, list to hold clauses with one literal in unit propagation, a set for pure literal elimination, and multiple other lists for tracking results.
10	General operation of code (for each subproject): Generally the code parses inputs, performs unit propagation, pure literal elimination, implements the dpll algorithm with recursion, then 2SAT solver uses these functions to solve the problem for each parsed input which is then printed as satisfiable or not, and then plot on a graph against time.
11	What test cases you used/added, why you used them, what did they tell you about the correctness of your code: I just used some fairly simple 2SAT questions that I already knew the answers to so that I would know whether or not my code was giving me correct results when outputs satisfiable or not satisfiable. I used these to ensure I'm not just making up random results and that my code was accurate.
12	How you managed the code development: I went to the wikipedia page that showed the dpll algorithm and read it a few times. There were some basic steps it showed as well as some sudo code to get me started so I followed those instructions and went from there.
13	Detailed discussion of results: The results of the graph vary slightly each time I run it but over all the more the number of variables in the problem the longer it takes for the solver to figure it out whether it be satisfiable or not.
14	How team was organized: I worked on a single team.
15	What you might do differently if you did the project again: If I did the project again I might have gone over the prompt a little bit more carefully as well as do a little bit more background research so as to not run over as many snags.
16	Any additional material: none