Project 01 Readme Team erhee

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: erhee
2	Team members names and netids: Elizabeth Rhee (erhee)
3	Overall project attempted, with sub-projects: The attempt was to improve the DumbSat into a 2SAT polynomial using DPLL.
4	Overall success of the project: somewhat successful- it ran through everything it was supposed to do and the plot looks somewhat how it is supposed to be as a 2SAT is a special case and the time complexity is often close to linear; however all the output is unsatisfiable, which is not correct
5	Approximately total time (in hours) to complete: 20 hours
6	Link to github repository: https://github.com/ejr88/nd-cse-30151-project01-erhee
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_code_erhee.py	Code to implement DPLL and create a 2SAT
Test Files	
check_2SAT_code.cnf.csv	Csv file given by professor to test the 2SAT code to determine satisfiability per clause and to plot the time analysis. Testing was completed in code.
Output Files	
Output_test_erhee.csv	Csv output file to see the

	<table><tr><td>satisfiable_statements.png</td><td>points on the graph. Used matplotlib to create plots The png is a screenshot of the statements of satisfiability</td></tr><tr><td colspan="2">Plots (as needed)</td></tr><tr><td>plot1_erhee.png plot2_erhee.png</td><td>Plots to see the trend line and all the points. Matplotlib used to create plots</td></tr></table>
8	Programming languages used, and associated libraries: Python using matplotlib, csv, and time libraries
9	Key data structures (for each sub-project): lists
10	General operation of code (for each subproject): This code attempts to implement dpll. Multiple lists are initialized in the beginning. It begins by opening a csv file and reading each line. My code checks the first value to see if it is a c or a p and will append specific values depending on which one. For example, if there is a c, it will append the problem number. If there is no letter as the first value of a line, it appends that clause to a clause list, which will contain the clauses for one problem. That clause list is one argument in the dpll function. Dpll calls the unit_propagate function, which is supposed to find clauses of length 1, which means there is only one literal, and thus that literal can only have one assignment. It will remove clauses that have this literal because it will always be true for one assignment. Next in dpll the pure_literal_elim function which is supposed to identify if there a literal is only positive or negative in all the problems clauses. If yes, it updates truth values and removes it. The algorithm chooses the literal and then recursively calls itself until you get a satisfiability statement
11	What test cases you used/added, why you used them, what did they tell you about the correctness of your code. I did not add any new tests, but I used the 2SAT cnf csv file from the professor to test my code. I used this file to check if the code properly printed out the satisfiability per clause and to see the number of literals per time on a graph.
12	How you managed the code development: I began by researching the algorithm and spending time to understand how it worked. I decided on functions to separate into, specifically a dpll function, a unit propagation function, a pure literal elimination function, and a function to create an output file. Instead of making a main, I just did not indent the main code. I started with unit propagation function, then the pure literal elimination function, and dpll. I then worked on main. To debug, I used multiple print and break statements, especially in main, so I could check singular values.

13	Detailed discussion of results: Unfortunately, my graph is incorrect, so these results most likely do not follow what it is supposed to be. The points on the graph are somewhat increasing as it goes to the right, but not by much. My graphs have a somewhat linear looking line, which makes sense for 2 SAT. Typically, DPLL is O(2^v); however 2SAT is a special case and it can have almost linear time.
14	How team was organized: I was alone so I did everything
15	What you might do differently if you did the project again: I would probably spend a little more time understanding the nuances between differences in how to implement DPLL to really hone in on which method is the most efficient. I would start earlier.
16	Any additional material: N/A