Project Incremental DumbSAT Readme Team Alliegator

1	Alliegator
2	Allie Sproul asproul
3	Rewrite DumbSAT to use an incremental search through possible solutions
4	Overall success of the project: Mild - it took some effort but I got there eventually
5	Approximately total time (in hours) to complete: 8
6	https://github.com/alliesproul/theory_alliegator
7	List of included files:

File/folder Name	File Contents and Use
Code Files	
DumbSAT_alliegator.ipynb	Contains DumbSAT solves with incremental search
Test Files	
data_alliegator.txt	Contains test cases to be parsed through in the format <Nvars> <NClauses> <LitsPerClause> <Ntrials>
Output Files	
output_cnffile_alliegator.cnf	Contains SAT solver's inputs and outputs in CNF
output_resultsfile_alliegator.csv	Contains the results of running the SAT solver on various test cases
output_tracefile_alliegator.csv	Contains detailed steps during the execution of each SAT solver run, useful for debugging or understanding how the solver arrived at a solution
output_program_alliegator.png	Contains output of program with execution times and if clause was satisfiable with satisfying assignments
Plots (as needed)	

	<table><tr><td>plots_alliegator.png</td><td>Contains image of plot of number of variables vs execution time</td></tr></table>
8	Programming languages used, and associated libraries: Python: time, random, csv, matplotlib, numpy
9	Key data structures: wff - represents a list of clauses, Assignment - list representing a binary assignment, sizes, times, satisfiable_vars, and unsatisfiable_vars - all stored results of my program
10	General operation of code: My code parsed the wffs from a file using parse_file, it has an increment function to iterate through binary assignments, it has a satisfiability check that takes a wff and attempts every possible assignment to see if it satisfies the formula, the test_wff function runs the satisfiability check and measures the execution time, build_wff generates random wffs to create random test cases, and lastly run_cases orchestrates multiple test cases, stores results, and writes output files. It also generate CSV files and plots a graph.
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 test cases in the original DumbSAT code and compared my results to those of the DumbSAT code. I made sure I used both large and small wffs to see the difference in how long they took. I used these because I knew what the correct output should be. These test cases showed me that even with my incremental search, larger problems still took an exponential amount of time. In my final code I added a function to parse an external file and created a test case of my own for my program to run through. This test case was long enough to show the exponential time on my graph but still completed in a reasonable amount of time.
12	How you managed the code development: I used past code I have written as reference such as creating the graphs and parsing through the file. I also asked the professor and TAs for help when needed. I have a function to handle each part of the problem (ie parse_file, test_wff, etc) and I tested at each step with print statements to ensure my code was working correctly. I tried to start small with what I was adding to the code before adding functionality such as parse_file.
13	Detailed discussion of results: My project outputs the execution time for each satisfiable and unsatisfiable wff. It also displays a graph plotting the variable vs the execution time to emphasize the exponential growth of the time needed as the clauses grow in length. My results ultimately show how inefficient the DumbSAT solver is and that the incremental search does not improve its efficiency. As the wffs grow in number of variables, the execution time grows exponentially.
14	How team was organized: N/A
15	What you might do differently if you did the project again: If I did the project again I would work with at least one more person instead of alone. I know in a larger team we would need to complete a larger project, but I think collaborating with others would have made this project easier because we could bounce ideas off of each other.

16	Any additional material: