ckotsen_teamname

Version 1 8/22/24

A copy of this file should be included in the github repository with your project. Change the teamname above to your own

1. Team name: ckotsen
2. Names of all team members: Caz Kotsen
3. Link to github repository: https://github.com/cjkot11/cjkot11-TheoryProj/blob/main
4. Which project options were attempted: I attempted to rewrite the DumbSAT with an incremental search.
5. Approximately total time spent on project: 10 hours
6. The language you used, and a list of libraries you invoked: I used Python and the following libraries: csv, time, matplotlib, numpy, and scipy
7. How would a TA run your program (did you provide a script to run a test case?). Simply running the code in any compiler should work. The source code includes all functions and a main which is called.
8. A brief description of the key data structures you used, and how the program functioned: This program reads logic problems from a CSV file, each consisting of multiple clauses, and solves them using the DPLL (Davis-Putnam-Logemann-Loveland) algorithm. The problems are processed to either determine a satisfiable assignment of variables or confirm that the problem is unsatisfiable. The program tracks the size of each problem (number of clauses) and the time it takes to solve it. The unit propagation, pure literal elimination, and recursive backtracking steps of the DPLL algorithm are applied to find solutions. After solving, the program plots the results in a scatter plot, where satisfiable and unsatisfiable problems are shown.
9. A discussion as to what test cases you added and why you decided to add them (what did they tell you about the correctness of your code). Where did the data come from? (course website, handcrafted, a data generator, other). All data I used came from the course website and I added them while writing my code in order to ensure accuracy.
10. An analysis of the results, such as if timings were called for, which plots showed what? What was the approximate complexity of your program? The results were great as shown by the graph which displays the size of the problem as X axis and the time as the Y axis. As shown, the time increases with the increase of problem size. I would say the timings make sense and showed a much better version of the brute force code from the course website. The DPLL algorithm generally has a worst-case time complexity of O(2^n), but the plot shows how execution time scales with problem size in practice.
11. A description of how you managed the code development and testing. I tried to space out the code development so that I could figure out the algorithm with new ideas each time. In terms of testing, I struggled cleaning the data how I wanted, but figured it out. I also faced the difficulty of a very slow computer.
12. Did you do any extra programs, or attempted any extra test cases: Yes I attempted some test cases on my own in order to troubleshoot.