Project TeamworkTemplate

Version 1 9/11/24

A ***separate copy*** of this template should be filled out and submitted by each student, regardless of the number of students on the team. Also change the title of this template to “Project x Teamwork <team> - <netid>”

1	Team Name: **ariter**
2	Individual name: **Alyssa Riter**
3	Individual netid: **ariter**
4	Other team members names and netids: **N/A**
5	Link to github repository: https://github.com/AlyssaRiter23/Theory-DPLLSolver/tree/main
6	Overall project attempted, with sub-projects: **Implementing a polynomial-time 2SAT solver (DPLL algorithm)**
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)

File/folder Name	File Contents and Use
Code Files	
dpll_ariter.py	**This code implements the DPLL algorithm. It also generates CNFs to test the 2SAT problem as well as code to graph the results of the program.**
Test Files	
dpll_ariter.py **check_ariter.txt**	**Randomly generated CNFs in the dpll_ariter.py program using the *generate_random_CNFs* function. The randomly generated CNFs can be found in the check_ariter.txt file.**
Output Files	
check_ariter.txt **output_ariter.txt**	**The check_ariter.txt file contains the CNFs generated from the *generate_random_CNFs* function in the dpll_ariter.py program which can be considered both a test file and an output file. output_ariter.txt contains the literal assignments that were created and whether or not each CNF was satisfiable or unsatisfiable, it also records the execution time for each trial.**

<table>
<tr><td></td><td>
<table>
<tr><td colspan="2">Plots (as needed)</td></tr>
<tr><td>execution_time_plot.png</td><td>This image shows the result of the DPLL algorithm and the polynomial time execution. This plot was created in a function called plot_execution_times in the dpll_ariter.py program.</td></tr>
</table>
</td></tr>
<tr><td>8</td><td>Individual Student time (in hours) to complete: 6-7 hours</td></tr>
<tr><td>9</td><td>Your specific activities and responsibilities: I researched the DPLL algorithm and implemented it in python code through numerous functions. I Implemented backtracking, unit propagation, and pure literal elimination all of which are characteristics of the DPLL algorithm. Additionally, I created a function to generate random 2SAT test cases to verify the correctness of my program and graph the results. I also kept track of the execution time for each case and whether or not the CNF was satisfiable. I used lists to store these metrics and then plotted it using matplotlib and the plot_execution_times function I created.</td></tr>
<tr><td>10</td><td>What was personally learned (topic, programming, algorithms). Overall, I learned about the Davis–Putnam–Logemann–Loveland (DPLL) algorithm and how it is used to solve SAT problems which are NP-complete. The DPLL algorithm expands on the basic backtracking (recursive) algorithm by incorporating unit propagation and pure literal elimination. Unit propagation asserts that if a clause is a unit clause (only has a single literal) then that literal must be assigned to true in order for the entire CNF to be satisfiable. Unit propagation simplifies the problem by removing every clause containing a unit clause’s literal and discards the complement of the unit clause’s literal from every clause containing it. DPLL and my program also incorporated pure literal elimination which is when a propositional variable occurs with only one polarity in the formula and so it can be assigned in a way that makes all the clauses it is in true. If a clause becomes empty then we know that the partial assignment is unsatisfiable. Furthermore, I learned a lot about the various libraries in python, specifically the copy library, and I gained more experience with plotting in python. It was incredible to see that as the number of variables increased the execution time began to display exponential worst-case time complexity, giving me deep insight into the world of NP-complete problems. This revealed to me that although a solution to an NP-complete problem can be verified fairly quickly (i.e me checking if my program is operating correctly and producing accurate results), there is no way to actually find that solution quickly. Furthermore, the time required to solve the CNFs using the DPLL algorithm increases rapidly as the size of the problem grows. As observed by my graph, NP-complete problems, like SAT can be solved by algorithms like DPLL, but have an exponential worst-case time complexity.</td></tr>
<tr><td>11</td><td>How the team was organized, and what might be improved: There was no team as I completed this project individually. However, I think that overall the use of Github was greatly beneficial for both an individual and team because it provides a way to organize all the programs in a clear and concise manner. Additionally, members can contribute and view each other's work, allowing for good</td></tr>
</table>

	collaboration. As an individual, Github made it easier to organize and layout all the necessary programs and files that I need for submission. To improve I would have myself set more strict deadlines for when to complete each portion of code so that the project was implemented with better time-management.
12	Any additional material: **N/A**