Project Readme Template

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"

<table>
<tr><td>1</td><td>Team Name: cmassman</td></tr>
<tr><td>2</td><td>Team members names and netids: Katie Massman (netid: cmassman)</td></tr>
<tr><td>3</td><td>Overall project attempted, with sub-projects:
Main Project: SAT
Sub-projects: rewriting DumbSAT to use an incremental search through possible solutions (#2). This involved parsing CNF files to extract k-SAT clauses, implementing an incremental backtracking algorithm for solving k-SAT problems, validation and comparison of solver results with expected satisfiability from benchmark datasets, and performance analysis and timing for different problem sizes and clause structures.</td></tr>
<tr><td>4</td><td>Overall success of the project:
The project was completed successfully. The backtracking-based solver was able to handle k-SAT problems of varying complexity. The validation system flagged incorrect results for deeper analysis, and the overall performance was reasonable given the nature of k-SAT being NP-complete.</td></tr>
<tr><td>5</td><td>Approximately total time (in hours) to complete: Around 5 hours spent working on the algorithm implementation, testing, and performance analysis.</td></tr>
<tr><td>6</td><td>Link to github repository: https://github.com/cmassman1/theory_proj1</td></tr>
<tr><td>7</td><td>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.
<table>
<tr><td>File/folder Name</td><td>File Contents and Use</td></tr>
<tr><td colspan="2">Code Files</td></tr>
<tr><td>incremental_cmassman.py</td><td>Main implementation of the backtracking k-SAT solver.</td></tr>
<tr><td colspan="2">Test Files</td></tr>
<tr><td>kSAT_cmassman.csv</td><td>CNF test cases used</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><td></td><td>
<table>
<tr><td></td><td>for solving k-SAT problems. Contains expected results for validation (satisfiable / unsatisfiable)</td></tr>
<tr><td colspan="2">Output Files</td></tr>
<tr><td>outputfile_cmassman</td><td>Output results of the solver with timing data.</td></tr>
<tr><td colspan="2">Plots (as needed)</td></tr>
<tr><td>plot_cmassman</td><td>Plots of performance results (time vs. variables/clauses).</td></tr>
</table>
</td></tr>
<tr><td>8</td><td>Programming languages used, and associated libraries:
Python was the primary language, with csv, matplotlib, sys and time libraries used</td></tr>
<tr><td>9</td><td>Key data structures (for each sub-project):
List of Clauses: Each clause in the CNF formula is stored as a list of literals (variables and their negations).
Assignment List: A list used during the incremental search to track truth assignments to variables.</td></tr>
<tr><td>10</td><td>General operation of code (for each subproject):

Incremental k-SAT Solver:

● The solver reads CNF files representing k-SAT problems, assigns truth values to variables incrementally, and checks if the clauses are satisfied. If a clause is unsatisfied, the solver backtracks and tries a different assignment.
● Validation: Compares the solver's results with expected values from the results file. If a mismatch is found, the program halts and reports the error.</td></tr>
<tr><td>11</td><td>What test cases you used/added, why you used them, what did they tell you about the correctness of your code:
Test cases include CNF files with varying numbers of clauses and variables to evaluate the solver's efficiency and correctness. I used both satisfiable and unsatisfiable test cases to ensure the solver handles both outcomes. The tests confirmed the correctness of the solver by comparing its results with known outcomes of the CNF files. I used the kSAT solver which was provided in Canvas.</td></tr>
<tr><td>12</td><td>How you managed the code development:</td></tr>
</table>

	The development process was managed using GitHub for version control. Frequent commits ensured that progress was tracked, and debugging was handled in a collaborative manner with detailed commit messages.
13	Detailed discussion of results: **The solver performed well on all provided test cases, correctly identifying satisfiability or unsatisfiability. The runtime plots indicate that the solver handles larger inputs efficiently, and the solver consistently returns accurate results when compared to expected outputs for both simple and complex CNF datasets.**
14	How team was organized: **This was an individual project. All tasks, including implementation, testing, and debugging, were handled independently.**
15	What you might do differently if you did the project again: **Introduce heuristics to select variables in a smarter way during backtracking (e.g., selecting the most constrained variable first).**
16	Any additional material: **Plots showing the time complexity of the k-SAT solver are included in the repository.**