

Project 2 Readme Team Andrew

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: Team Andrew																	
2	Team members names and netids: Andrew Cotaj - acotaj																	
3	Overall project attempted, with sub-projects: NTM Trace																	
4	Overall success of the project: Very Succesful																	
5	Approximately total time (in hours) to complete: ~9-10 hours																	
6	Link to github repository: https://github.com/AdrewC2026/Project2-TOC																	
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. <table border="1"><thead><tr><th>File/folder Name</th><th>File Contents and Use</th></tr></thead><tbody><tr><td align="center" colspan="2">Code Files</td></tr><tr><td><code>./src/ntm_tracer.py</code></td><td>File providing functionality for NTM Tracer utilized by main.py *All other files (enrtpoint.py, main.py, and helper files were not changed)</td></tr><tr><td align="center" colspan="2">Test Files</td></tr><tr><td><code>./src/input/a_plus.csv</code> <code>./src/input/composite.csv</code> <code>./src/input/equal_01s.csv</code> <code>./src/input/ktape.csv</code> <code>./src/input/ntm_n1n.csv</code></td><td>Provided file - accepts string of as Provided file - accepts composite numbers Provided file - accepts string of equal number of 0s & 1s Included in Forked Repo - Different Topic Included in Forked Repo - Different Topic</td></tr><tr><td align="center" colspan="2">Output Files</td></tr><tr><td><code>./aplus_aaa.</code></td><td>Photo showing output of running program with aplus.csv and input aaa</td></tr><tr><td><code>./composite_111111.</code></td><td>Photo showing output of running program with</td></tr></tbody></table>		File/folder Name	File Contents and Use	Code Files		<code>./src/ntm_tracer.py</code>	File providing functionality for NTM Tracer utilized by main.py *All other files (enrtpoint.py, main.py, and helper files were not changed)	Test Files		<code>./src/input/a_plus.csv</code> <code>./src/input/composite.csv</code> <code>./src/input/equal_01s.csv</code> <code>./src/input/ktape.csv</code> <code>./src/input/ntm_n1n.csv</code>	Provided file - accepts string of as Provided file - accepts composite numbers Provided file - accepts string of equal number of 0s & 1s Included in Forked Repo - Different Topic Included in Forked Repo - Different Topic	Output Files		<code>./aplus_aaa.</code>	Photo showing output of running program with aplus.csv and input aaa	<code>./composite_111111.</code>	Photo showing output of running program with
File/folder Name	File Contents and Use																	
Code Files																		
<code>./src/ntm_tracer.py</code>	File providing functionality for NTM Tracer utilized by main.py *All other files (enrtpoint.py, main.py, and helper files were not changed)																	
Test Files																		
<code>./src/input/a_plus.csv</code> <code>./src/input/composite.csv</code> <code>./src/input/equal_01s.csv</code> <code>./src/input/ktape.csv</code> <code>./src/input/ntm_n1n.csv</code>	Provided file - accepts string of as Provided file - accepts composite numbers Provided file - accepts string of equal number of 0s & 1s Included in Forked Repo - Different Topic Included in Forked Repo - Different Topic																	
Output Files																		
<code>./aplus_aaa.</code>	Photo showing output of running program with aplus.csv and input aaa																	
<code>./composite_111111.</code>	Photo showing output of running program with																	

	<p><code>./equal_01s_001011.</code></p> <p>composite.csv and input 111111</p> <p>Photo showing output of running program with equal_01s.csv and input 001011</p> <p>*these pictures were requested in Project 2 NTM Submission Assignment on canvas (photos will be submitted on that assignment and will also be in repository)</p> <p>Plots (as needed)</p> <table border="1"> <tr> <td>N/A</td><td>N/A</td></tr> </table>	N/A	N/A
N/A	N/A		
8	Programming languages used, and associated libraries: Python (including ‘typing’ library)		
9	<p>Key data structures (for each sub-project): My rendition of the NTM Tracer relies primarily on a hierarchical tree structure implemented as a list of lists, where each index i contains all active machine configurations at depth i. Each configuration is stored as a dictionary containing the left and right tape contents, the current state, the specific transition rule used (trans), and a parent tuple (depth, index) which is essential for backtracking to reconstruct the accepting path.</p> <p>Additionally, the TuringMachineSimulator (provided) parses the CSV input into a transition map - a dictionary where keys are tuples of (state, read_symbol) and values are lists of possible actions - which serves as the core mechanism for enabling and managing the nondeterministic branching behavior.</p>		
10	<p>General operation of code (for each subproject): The code operates using a BFS algorithm that initializes the execution tree with a single root node and iteratively expands it layer-by-layer until a maximum depth is reached or an accept state is found. During each iteration, the program examines every configuration in the current level, consulting the transition map to spawn new child configurations for every valid move; if a configuration reaches an accept state, the loop terminates immediately to guarantee the shortest path, whereas invalid or rejecting branches are simply dropped from the next level. Simultaneously, the system tracks the count of non-leaf nodes and outgoing transitions to calculate the average degree of nondeterminism, and upon successful acceptance, it utilizes the parent pointers to backtrack and print the precise sequence of transitions from the start.</p>		
11	<p>What test cases you used/added, why you used them, what did they tell you about the correctness of your code: I verified the correctness of the code using three distinct machines that targeted specific edge cases and complexity levels: a_plus.csv served as the baseline to confirm basic branching and rejection logic (e.g., accepting "aaaa" vs rejecting "b"); equal_01s.csv tested the robustness of tape management and head movement with its complex "zig-zag" matching patterns; composite.csv verified the loop termination logic by forcing the machine to hit the max_depth limit when processing prime numbers like 111.</p>		
12	<p>How you managed the code development: I utilized an incremental development strategy, beginning with the integration of the TuringMachineSimulator class into my program to ensure the CSV parsing correctly stored nondeterministic transitions as lists of tuples. Once parsing was stable, I</p>		

	<p>built the NTM_Tracer class, starting with a basic BFS loop to verify tree expansion.. Testing was conducted sequentially, ensuring the simple a_plus machine functioned perfectly before integrating and debugging the more memory-intensive logic required for the composite and equal_01s machines.</p>
13	<p>Detailed discussion of results: The results highlighted the distinct performance characteristics of NTMs, where the a_plus machine showed a high degree of nondeterminism (~1.8) due to constant branching, while equal_01s remained mostly deterministic (~1.1) with targeted guess points. The traces confirmed that the BFS approach was superior to DFS for this assignment, as it successfully found shallow solutions (like in composite.csv) without getting trapped in infinite recursion depth. Furthermore, the successful execution of the equal_01s traces demonstrated that the tape management logic correctly handled the "infinite blank tape" assumption by dynamically padding the string representation whenever the head moved beyond the existing input boundaries.</p>
14	<p>How team was organized: Individual Project - N/A</p>
15	<p>What you might do differently if you did the project again: If I were to redo this project, for cleanliness sake, I would optimize the memory management by replacing the "list of lists" history structure with a standard queue for BFS and a separate dictionary for parent pointers, as the current method retains every computed configuration and consumes exponential memory on deep trees. I would also implement a visited set to hash and prune redundant configurations (identical state and tape content) to prevent processing duplicate branches, which would significantly increase the depth the simulator could reach before hitting memory limits.</p>
16	<p>Any additional material: N/A</p>