

Project 2 Readme Team CD

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: CD																			
2	Team members names and netids - Andre Mayard, amayard																			
3	Overall project attempted, with sub-projects: NTM Trace																			
4	Overall success of the project: Success																			
5	Approximately total time (in hours) to complete: 4 hrs																			
6	Link to github repository:																			
7	<p>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.</p> <table border="1"><thead><tr><th>File/folder Name</th><th>File Contents and Use</th></tr></thead><tbody><tr><td colspan="2">Code Files</td></tr><tr><td>src</td><td>ntm_tracer_CD.py</td></tr><tr><td colspan="2">Test Files</td></tr><tr><td>input</td><td>ntm_n1n.csv</td></tr><tr><td colspan="2">Output Files</td></tr><tr><td>.</td><td>output.txt</td></tr><tr><td colspan="2">Plots (as needed)</td></tr><tr><td></td><td></td></tr></tbody></table>		File/folder Name	File Contents and Use	Code Files		src	ntm_tracer_CD.py	Test Files		input	ntm_n1n.csv	Output Files		.	output.txt	Plots (as needed)			
File/folder Name	File Contents and Use																			
Code Files																				
src	ntm_tracer_CD.py																			
Test Files																				
input	ntm_n1n.csv																			
Output Files																				
.	output.txt																			
Plots (as needed)																				
8	Programming languages used, and associated libraries: python																			
9	<p>Key data structures (for each sub-project):</p> <p>The program uses configurations stored as lists of four elements: the left tape, the current state, the right tape, and the parent index. These configurations are organized into a tree where each level is a list of configurations representing that BFS depth.</p>																			

	Transitions are kept in a dictionary keyed by (state, symbol) and the values are lists of transition descriptions. A path is reconstructed by following parent indices level by level back to the root.
10	<p>General operation of code (for each subproject)</p> <p>The program runs a BFS over all possible computation paths of the NTM. It starts from the initial configuration and expands level by level, checking each configuration for accept or reject status. If it is neither, it looks up valid transitions and generates new configurations with updated tape content and a parent pointer. The search continues until an accept state is found, all branches reject, or the maximum depth is reached. Once it ends, it reports acceptance or rejection and reconstructs the path if accepted.</p>
11	<p>What test cases you used/added, why you used them, what did they tell you about the correctness of your code.</p> <p>I ran my code using various 0 and 1 combinations and determined that it returned the right answer each time - I edited and used the ntm_n1n.csv file to test this. I verified these answers by tracing through it myself, and I verified it was accurate.</p>
12	<p>How you managed the code development</p> <p>I wrote my own pseudocode for both the print function and the actual tracing function, and then I implemented that with some hurdles to eventually it working. I did it all in one sitting</p>
13	<p>Detailed discussion of results:</p> <p>The results showed each time that the code was correct, indicating that there were no issues with the code. I therefore implemented an NTM tracer.</p>
14	How team was organized - I did all the work
15	What you might do differently if you did the project again - I wouldn't do anything differently besides starting earlier.
16	Any additional material: None