

Project 2 Readme Team perchance

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: perchance	
2	Team members names and netids: Colin Nguyen (cnguyen8)	
3	Overall project attempted, with sub-projects: NTM Tracer	
4	Overall success of the project: Successful	
5	Approximately total time (in hours) to complete: 5 hours	
6	Link to github repository: https://github.com/colinqnguyen/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.	
File/folder Name		File Contents and Use
Code Files		
src/ntm_tracer.py		NTM tracer program. Traces all possible paths the NTM might take and stops at accept state or when all paths end at reject state.
Test Files		
input/aplus.csv		NTM that accepts a string of a's.
input/composite.csv		NTM that accepts {1^n n is composite}.
input/equal_01s.csv		NTM that accepts {w w has the same number of 0's and 1's}.
Output Files		

	<table border="1"> <tr> <td>results/NTM_output.txt</td><td>A text file containing the output of each test file.</td></tr> <tr> <td colspan="2">Plots (as needed)</td></tr> <tr> <td></td><td></td></tr> </table>	results/NTM_output.txt	A text file containing the output of each test file.	Plots (as needed)			
results/NTM_output.txt	A text file containing the output of each test file.						
Plots (as needed)							
8	Programming languages used, and associated libraries: Python						
9	<p>Key data structures (for each sub-project):</p> <p>The key data structure is a list of lists (tree). Each inner list consists of the tape contents, the state, and parent information.</p>						
10	<p>General operation of code (for each subproject):</p> <p>The program uses breadth first search in order to simulate an NTM. It goes through each transition and stops at an accepting configuration. The accepting path is also traced. If every path leads to the reject state, then the input is rejected.</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>The test files I used were aplus, composite, and equal_01s. These test files are all from the class's NTM Test Files folder. My program was able to successfully run with these test files and output as expected, therefore validating my code.</p>						
12	<p>How you managed the code development</p> <p>I worked on one section of the NTM tracer at a time. Firstly, I focused on figuring out how to program breadth first search in order to simulate an NTM. Then once I felt the breadth first search functioned correctly, I then implemented parent tracking and formatting the output.</p>						
13	<p>Detailed discussion of results:</p> <p>My NTM tracer can successfully trace each path of an NTM and accurately show both the accept and reject conditions.</p> <p>For the aplus test case with input “aaaa”, the output displayed a small computation tree with one path reaching the accept state, thus demonstrating nondeterminism ad early acceptance.</p> <p>For the composite.csv on input “1111111”, the output displays heavy nondeterminism with a big tree. There are numerous rejected branches before an accepted path is found, proving that the NTM tracer correctly explores all possibilities.</p>						
14	<p>How team was organized</p> <p>The project was done individually.</p>						

15	What you might do differently if you did the project again If I did the project again, I would spend more time working on the output before trying to finalize my breadth first search code. I had trouble debugging it because I didn't have a definite way of knowing what it outputted. Also, I would spend more time implementing test cases.
16	Any additional material: