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: Roadie</td></tr>
<tr><td>2</td><td>Team members names and netids: N/A</td></tr>
<tr><td>3</td><td>Overall project attempted, with sub-projects: Tracing NTM Behavior</td></tr>
<tr><td>4</td><td>Overall success of the project: From what I tested, my project was successful. I used the a+ and a*b*c* machines and tested several different input strings. They all accepted when they should accept and rejected when they should reject, besides the cases that reached past the max depth of 75 transitions.</td></tr>
<tr><td>5</td><td>Approximately total time (in hours) to complete: 7.5</td></tr>
<tr><td>6</td><td>Link to github repository: https://github.com/msorenso/TMProj</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>traceTM_roadie.py</td><td>Python code used for NTM tracing</td></tr>
<tr><td colspan="2">Test Files</td></tr>
<tr><td>a_plus.csv</td><td>CSV file with a+ TM description</td></tr>
<tr><td>abc_star.csv</td><td>CSV file with a*b*c* TM description</td></tr>
<tr><td colspan="2">Output Files</td></tr>
<tr><td>output roadie .pdf</td><td>PDF file containing input and output for test cases</td></tr>
<tr><td>Screenshot 2024-12-08 232757.png</td><td>This is a screenshot of</td></tr>
</table>
</td></tr>
</table>

	<table><tr><td></td><td>the output when ran in the terminal for reference.</td></tr></table>
8	Programming languages used, and associated libraries: **Language:** python **Libraries:** csv, sys, collections (defaultdict)
9	Key data structures (for each sub-project): Dictionaries and Lists I used a dictionary to store the machine, a dictionary of lists for the transitions, lists for the states, sigma, and gamma, and strings for the start, reject, and accept states and the name of the machine.
10	General operation of code (for each subproject): My code reads in the name of the machine and the input string. It reads in the information of the machine and stores it in a dictionary. It then uses this dictionary along with the input string to simulate all the possible paths the machine could go through, using a breadth first search. If at any point a branch reaches a reject state, the branch terminates but continues to run the other branches. If at any point a branch reaches an accept state, the simulation stops and the results are printed out. If the depth of the simulation tree reaches over 75, the program will stop.
11	What test cases you used/added, why you used them, what did they tell you about the correctness of your code: The test cases I used are all listed below in the Output Table. I used different strings that I knew would either accept or reject and then made sure my program outputted the correct solution. I also briefly went through the tree to see if the way they got to the solution was valid.
12	How you managed the code development: After understanding the algorithm I wanted to implement, I wrote all of the code by myself. I used examples of similar codes I created in my other classes such as Programming Challenges as inspiration and guidance.
13	Detailed discussion of results: My results were fairly straightforward. I would have my program print the name of the machine, the initial string, the depth of the tree, the total configurations, the tree of the configurations level by level, and then whether or not the strings were rejected or accepted and in how many transitions. For the results I tested, they are all accurate; if a string should be accepted it was and if it shouldn't it didn't. I tried three different strings each on two matches (a+ and a*b*c*).
14	How team was organized: I worked on the project individually
15	What you might do differently if you did the project again: When I first wrote my code, I dove head first into it and tried to write all of the functions and test it all afterwards. This led to a lot of time spent on debugging. If I were to do this project again, I think I would write my code in little steps and test them along the way.
16	Any additional material:

	I do not have any additional material to add. Please feel free to reach out to me if you have any questions about our answers.

Output table:

Machine	Input String	Result	Depth	Configurations	Average Non -Determinism
a+	aaa	accepted	5	8	1.6
a+	“”	rejected	1	1	1
a+	aaaabaaa	rejected	5	9	1.8
a*b*c*	aaabcc	accepted	8	28	3.5
a*b*c*	b	accepted	3	5	1.67
a*b*c*	aabbcca	rejected	7	29	4.14