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”

1	Team Name: CT
2	Team members names and netids: Chau Ta cta
3	Overall project attempted, with sub-projects: NTM Tracer
4	Overall success of the project: Successful
5	Approximately total time (in hours) to complete: 3 Hours
6	Link to github repository: https://github.com/cht-au/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	Main program that trace a NTM. Find accept and reject state and print a tree if accept
Test Files	
input/aplus.csv	NTM that accept a+
input/composite.csv	NTM that accept the string 1^n if n is a composite number
Input/equal_01s.csv	NTM that accepts {w | w has the same number of 0's and 1's}
Output Files	
output/output.txt	Txt file with outptuts from running all the test files above

	Plots (as needed)
8	Programming languages used, and associated libraries: Python
9	Key data structures (for each sub-project): Tree – list of list, where one inner list contains the left, the state, the right, the parent, and the transition number.
10	General operation of code (for each subproject) The machine does BFS to simulate possible branches for a NTM. We start with the initial configuration. At each step, we find valid transitions and add them onto our next level list. If we find an accepting state, we break out of the loop and print the tree. If it is in a reject state, then we skip it. If all paths are rejected, then we reject the string To print the depth and the number of transitions, we calculate the numbers of configuration we have at each level as well as how many levels we have in the tree.
11	What test cases you used/added, why you used them, what did they tell you about the correctness of your code. I tested my program for NTM aplus, composite, and equal_01s. The program accurately ran this code and give the expected outputs.
12	How you managed the code development I manage all the code since this is a one-person project. I first worked on finding the correct accepting node and implementing the search through the tree. After I finish the tree, I worked on outputting the results.
13	Detailed discussion of results: For aplus.csv, the machine was able to find the correct path for “aaa” and “aaaaaa”. This shows that the machine figured out the correct transition to take, whether to stay in state 1 or go to state 2 for the last ‘a’. Additionally, the machine also figures out all the possible cases that would lead to qreject. For the composite machine, it was able to confirm 6 to be a composite number. However, the test case for the composite machine is not accurate since it accept 5. For equal_01s machine, it passes all the tests by accepting if there are equals 0 and 1s and reject string that doesn’t have equal 0s and 1s. It passed the tests that I put in and manually verified
14	How team was organized: I did all the work since this is individual project
15	What you might do differently if you did the project again: I would add more test cases, especially one for testing infinite loops

16	Any additional material: