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: tparisi
2	Team members names and netids: tparisi
3	Overall project attempted, with sub-projects: Project 1 – Tracing NTM Behavior
4	Overall success of the project: Strong success, everything works as outlined by the project goals
5	Approximately total time (in hours) to complete: 5
6	Link to github repository: https://github.com/tim-parisi/theory-project2-ntm
7	

File/folder Name	File Contents and Use
Code Files	
traceTM_tparisi.py	Contains all code for the project. Usage: The program will prompt you for any required inputs (TM csv file, input string) but both can be specified with the flags -f [filename] and -t [tape], respectively. -d is a debug flag that shows the full path trace of the program. Flags -D and -T can be used to specify a maximum depth or number of transitions, respectively.
Test Files	
csv_files/	A collection of the provided test files, along with one I made based of a previous homework assignment (hw8_tm.csv)
Output Files	
output_files/	A collection of text files created by running traceTM_tparisi.py on all of the test files with the debug path trace enabled.
Plots (as needed)	

8	Programming languages used, and associated libraries: Python (csv and sys libraries)
9	Key data structures (for each sub-project): 3D array, trees
10	General operation of code (for each subproject): The program begins at the initial state with tape in front. The program then finds every possible transition it can take and creates a list of new states where they are taken. The program then goes through this list of states, finds all the possible transitions that can be taken from any of those states, and uses that to create a new list of states. This cycle repeats until either an accept state is reached, no transitions can be taken anymore, or a maximum depth/transition count is reached.

11	What test cases you used/added, why you used them, what did they tell you about the correctness of your code: I used multiple test cases to determine the accuracy of my program. There were many files I used (in the csv_files/ folder), but the standout characteristics I was looking for were an NTM with an average determinism greater than 1, a TM that looped constantly, and a TM that had no defined transitions for characters outside of the character list. Using these showed that my program could stop when it was supposed to, handle high levels of determinism correctly, and accept for the right strings.
12	How you managed the code development: I began by finding a good data structure for the transition tree and the transitions. Then I worked on generating a function to take a state and a transition and use those to “step” to a new state. Afterwards, I combined those into a function to process all transitions that can be taken from a list of states. Lastly, I put the finishing touches on the program such as printing all the relevant statistics and adding code flags.
13	Detailed discussion of results: The program passed every test case successfully, with the result, average determinism, and depth all returning what I expected. The program will return true for any successful string, understand NTMs, handle infinite loops without issue, and even handle strings with characters outside the language.
14	How team was organized: I wrote the whole program myself
15	What you might do differently if you did the project again: I could have improved cleanliness by splitting traceTM_tparisi.csv into multiple files.
16	Any additional material: