Project 2 Readme Team JPMachines

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: JPMachines</td></tr>
<tr><td>2</td><td>Team members names and netids: Juan Pablo Rubero, jrubero</td></tr>
<tr><td>3</td><td>Overall project attempted, with sub-projects:
Program 1: Tracing NTM Behavior</td></tr>
<tr><td>4</td><td>Overall success of the project:
I would definitely say that this project was a success, especially since I was able to meet the project goals and create a correct simulation of an NTM. Furthermore, the program offers detailed output and successfully handles multiple scenarios, such as acceptance and rejection. Also, the code of the program is well structured and easy to understand. Finally, the project was a success because it helped me understand nondeterminism and NTMs, deepening my understanding of the topics taught in class.</td></tr>
<tr><td>5</td><td>Approximately total time (in hours) to complete:
Around 8-10 hours</td></tr>
<tr><td>6</td><td>Link to github repository: juanpirubero/TraceTM_JPMachines: Theory of Computing Project 2, Juan Pablo Rubero</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_JPMachines.py</td><td>Python code for simulating the NTM</td></tr>
<tr><td colspan="2">Test Files</td></tr>
<tr><td>a_plus.csv</td><td>CSV file for testing a plus NTM</td></tr>
</table>
</td></tr>
</table>

	abc_star.csv — CSV file for testing abc star NTM Output Files output_aplus_JPMachines.txt — Output file for the test case of the a plus NTM output_abc_star_JPMachines.txt — Output file for the test case of the abc star NTM Plots (as needed)

Machine	Input	Result	Depth	# of configurations	Average Nondeterminism
a_plus.csv	aaa	accepted	5	8	1.60
	aaaba	rejected	4	7	1.75
	aaaaaaaaaa	accepted	12	22	1.83
	aaaaaaaabbaaaaaaaaa	rejected	9	17	1.89
abc_star.csv	aaabbbc	accepted	9	36	4.11
	aaacc	accepted	7	22	3.29
	aaaccbb	rejected	6	21	3.50
	aaaaabbbbbbbccccc	accepted	19	84	4.47

8	Programming languages used, and associated libraries: Programming languages: Python Associated libraries: csv, sys, defaultdict
9	Key data structures (for each sub-project): In this program, I used defaultdict, a list of lists, breadth first exploration, tuple, and a file object (used for opening the csv file)

10	General operation of code (for each subproject) This program simulates a Nondeterministic Turing Machine based on a configuration described in a CSV file (test cases). The way the program works is that it first loads the NTM configuration and here it reads the CSV file to extract the states, transition rules, etc. Then, the program runs the NTM using breadth-first search and tracks the configurations and it stops if the machine reaches an accept, reject state or it passes the max depth. Then, the program prints the output results.
11	What test cases you used/added, why you used them, what did they tell you about the correctness of your code. I used the NTM csv files that the professor provided, specifically the a_plus.csv and abc_star.csv files. For the test cases, I decided to use four different test cases per NTM. I first did a small, easy input and then I transitioned to an input that I knew was incorrect, and then I did the same thing again but with two larger test cases, to test the depth and total configurations of the NTM. I repeated my same process for the abc_star machine, but for this one I made sure to have 3 correct test cases to evaluate every outcome.
12	How you managed the code development I thought that the best course of action was to work in parts. I first worked on making sure I could read the csv file correctly. Then, I worked on being able to trace the NTM entirely and be able to simulate the NTM. After that, I worked on formatting the output results and being able to print all the necessary output correctly. Finally, I combined all of these parts together using the main function and making sure that all functions worked together well.
13	Detailed discussion of results: The results give us an in-depth overview of the Nondeterministic Turing Machine. The depth of the tree represents the number of levels explored in the search and the greater depth implies that the NTM took more steps to accept, reject, or end the program. The total transitions means the total number of transitions ran across all the configurations and if the machine is very nondeterministic, then the value can be much larger than the depth. The average nondeterminism tells us how many configurations the NTM explores at each depth, and a closer value to 1 represents a more deterministic behavior. Finally, the configuration tree displays all the configurations explored at each depth.
14	How team was organized I worked alone so I worked on everything of the program, the github repository, and the readme file.
15	What you might do differently if you did the project again If I would do the project again, I would better allocate my time on the project and focus more time on testing and debugging, since I heavily underestimated the difficulty of the test cases. I also would have spent more time understanding the csv files and nondeterminism, since I started the project without fully understanding. This caused problems in my implementation of the code and I had to sort of start from scratch again. Finally, I believe that my code is a little bit larger than what it's supposed to be, so for the future I would try, if possible, to use more efficient data structures in order to enhance the program.
16	Any additional material:

	N/A