Project x Readme Team PabloOlivaQuintana

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: PabloOlivaQuintana</td></tr>
<tr><td>2</td><td>Team members names and netids: Pablo Oliva Quintana (polivaqu)</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: Successful in my opinion</td></tr>
<tr><td>5</td><td>Approximately total time (in hours) to complete: 5 hours per day (2 days)</td></tr>
<tr><td>6</td><td>Link to github repository:

https://github.com/pablooliva04/Theory_Project_2_PabloOlivaQuintana</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>1. Tracing_PabloOlivaQuintana</td><td>1. The code simulates a Nondeterministic Turing Machine (NTM) using a Breadth-First Search algorithm. It checks if the input string is accepted, rejected, or if the computation times out, exploring all possible configurations.</td></tr>
<tr><td colspan="2">Test Files</td></tr>
<tr><td>2. a_plus_PabloOlivaQuintana
ends_with_bb_PabloOlivaQuintana</td><td>Specifies the machine's name States, Input Alphabet, Tape Alphabet, Start State, Accept</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><td></td><td>
<table>
<tr><td></td><td>State, Reject State, Transitions</td></tr>
<tr><td colspan="2">Output Files</td></tr>
<tr><td>1. simulation_output_a_plus_PabloOlivaQuintana.txt
2. simulation_output_ends_with_bb_PabloOlivaQuintanaPabloOlivaQuintana.txt</td><td>2. Results include the machine's name, computation depth, configurations explored, and degree of non-determinism.</td></tr>
</table>
</td></tr>
<tr><td>8</td><td>Programming languages used, and associated libraries:
Python libraries = csv, collections (deque)</td></tr>
<tr><td>9</td><td>Key data structures:
1. Tree: to build the machine
2. BFS: to compute the calculations
3. Dictionary: information from the turing machine stored in a machine</td></tr>
<tr><td>10</td><td>General operation of code

This code simulates a Nondeterministic Turing Machine (NTM) using a breadth-first search approach implemented with a queue to explore all possible paths in a tree-like structure. It begins by reading a CSV file that defines the machine’s configuration, including its name, states, alphabets, starting state, accepting and rejecting states, and transition rules. The simulate_ntm function processes an input string by exploring all configurations from the root (initial state) level by level, enqueuing new configurations based on valid transitions. The simulation terminates when the machine reaches an accept or reject state, or when the maximum depth is reached. Finally, the output_results function summarizes the simulation, displaying the machine’s name, input string, result (accept, reject, or timeout), depth, number of configurations explored, and the degree of nondeterminism, which then writes it to an output file.</td></tr>
<tr><td>11</td><td>What test cases you used/added, why you used them, what did they tell you about the correctness of your code.

I added the following test cases: “aaa” for a+ (deterministic) turing machine and “aaabb” for ends with bb (nondeterministic) turing machine.

The first test case involves the machine "ends with bb" and the input string aaabb. The machine correctly identifies that the string ends with the substring bb and produces an accepted result. The simulation reached a depth of 6, reflecting the number of state transitions required to process the input. A total of 7 configurations were explored, indicating that the machine branched out to check for</td></tr>
</table>

	potential matches to the substring bb while continuing to process the input. The average non-determinism of 1.17 highlights the machine's ability to handle multiple paths simultaneously, with minimal branching in this particular case. These results confirm the correctness of the machine's non deterministic behavior. The second test case tests the machine "a plus" with the input string aaa. The machine accurately recognizes this string as valid in the language L={a+}, producing an accepted result. The simulation required a depth of 4 transitions, which corresponds to processing each character sequentially and transitioning to the accept state. A total of 5 configurations were explored, which is consistent with the deterministic nature of the machine, where only one configuration is active at any time. The average non-determinism of 1.25 is a bit above 1 due to the machine handling transitions for blanks or the final acceptance step. These results demonstrate the correctness and efficiency of the deterministic "a plus" machine.
12	How you managed the code development: As in every project the most important thing I did was reading the document and the instructions carefully, setting mini milestones. First, I went on and ensured that I understood what NTMs and the problem that we were trying to solve were. It took me two days to develop the project. The first day, I coded the function that reads the description of the turing machine in the dictionary. The following day, I coded the actual tracing function, which was fairly complicated. Then I coded the output function and finally, I debugged and troubleshooted the code.
13	Detailed discussion of results: The results of the simulation have been validated to ensure that they align with the expected behavior of the Nondeterministic Turing Machine (NTM) as defined in the provided CSV files. Each result was cross-checked against known machine logic, ensuring the correctness of the program. For example, when using a_plus.csv, which is designed to accept strings of one or more a characters (a+), the program correctly accepts the input "aaa". It transitioned through the states, processed each character on the tape, and stopped in the accepting state (qacc). The same with the other one. The first test case for the nondeterministic machine "ends with bb" with the input aaabb correctly resulted in accept, as the string ends with the required bb. The simulation reached a depth of 6 and explored 7 configurations, demonstrating the machine's ability to branch nondeterministically when encountering potential matches for bb. The average non-determinism of 1.17 indicates minimal branching, as only a few paths were explored before confirming the result. In contrast, the input aaa resulted in a timed out outcome, as the machine was unable to find a match for bb and reached the maximum exploration depth. This shows the machine's proper handling of invalid inputs, where it explores all possible paths and stops gracefully when no valid sequence is found. These results confirm the machine's accuracy in identifying valid strings and its efficiency in managing nondeterministic

	paths.
	
	`--- Simulation Summary ---`
	`Machine: a plus`
	`Input String: aaa`
	`Result: accept`
	`Depth: 4`
	`Configurations Explored: 5`
	`Average Non-Determinism: 1.25`
	
	`Detailed Steps:`
	`Step 1: (, q1, aaa)`
	`Step 2: (a, q1, aa_)`
	`Step 3: (aa, q1, a__)`
	`Step 4: (aaa, q1, ___)`
	`Step 5: (aaa, qacc, ____)`
	
	`--- Simulation Summary ---`
	`Machine: ends with bb`
	`Input String: aaabb`
	`Result: accept`
	`Depth: 6`
	`Configurations Explored: 7`
	`Average Non-Determinism: 1.17`
	
	`Detailed Steps:`
	`Step 1: (, q1, aaabb)`
	`Step 2: (a, q1, aabb_)`
	`Step 3: (aa, q1, abb__)`
	`Step 4: (aaa, q1, bb___)`
	`Step 5: (aaab, q2, b____)`
	`Step 6: (aaabb, q3, _____)`
	`Step 7: (aaabb, qacc, ______)`

14	How team was organized: N/A (Only one person in the group)
15	What you might do differently if you did the project again: I would have tried starting it a bit earlier. We were given a lot of time to complete the project, and because of different reasons I ended up in a time crunch and I feel that it gave me a bit of anxiety.

	Therefore, I would start it again earlier to enjoy the process of coding it.
16	Any additional material: N/A