Project 2 Readme Team philipRivers

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:
philipRivers</td></tr>
<tr><td>2</td><td>Team members names and netids
Tim Vyverberg, tvyverbe</td></tr>
<tr><td>3</td><td>Overall project attempted, with sub-projects:
NTM trace main project</td></tr>
<tr><td>4</td><td>Overall success of the project:
I was able to successfully implement all parts of the project.</td></tr>
<tr><td>5</td><td>Approximately total time (in hours) to complete:
3</td></tr>
<tr><td>6</td><td>Link to github repository:
https://github.com/timvyve/Project2-TOC</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>ntm_tracer.py</td><td>The fundamental logic code for implementing the NTM trace, called in entrypoint.py.</td></tr>
<tr><td>entrypoint.py</td><td>Entrypoint for the project's package. Used to read in arguments and call our NTM trace code</td></tr>
<tr><td>argument_input.py</td><td>Used to parse through the arguments to the code, such as file name, input, etc.</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><td></td><td>
<table>
<tr><td>turing_machine.py</td><td>Reads in a csv file to “transform” (format) it into a turing machine we can work with.</td></tr>
<tr><td colspan="2">Test Files</td></tr>
<tr><td>aplus.csv</td><td>Used as test input for NTM trace</td></tr>
<tr><td>composite.csv</td><td>Used as test input for NTM trace</td></tr>
<tr><td>ntm_n1n.csv</td><td>Used as test input for NTM trace</td></tr>
<tr><td colspan="2">Output Files</td></tr>
<tr><td>Results.md</td><td>Stores the output from our three test files using various input strings, is in markdown format</td></tr>
<tr><td colspan="2">Plots (as needed)</td></tr>
<tr><td>N/A</td><td>N/A</td></tr>
</table>
</td></tr>
<tr><td>8</td><td>Programming languages used, and associated libraries:
Python is used, only libraries used are those in the starter repository.</td></tr>
<tr><td>9</td><td>Key data structures (for each sub-project):
- Configuration – this is a list that represents the Turing machine at one ‘snapshot’. It also includes values about the parents of each child, allowing us to backtrack.
- Tree represents the configurations at each depth. It is a list of lists, with the ‘further’ lists representing larger depths.
- Transitions is a dictionary that represents the transitions in our TM, it is found with TuringMachineSimulator</td></tr>
<tr><td>10</td><td>General operation of code (for each subproject)
- To operate the code, use uv run main.py <filename> <input_string>, where filename is the name of the csv file with the TM information. Input_string is the string being tested. The code will output (print) all of the relevant information from the trace.</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.
- As mentioned before, I used aplus.csv, composite.csv, and ntm_n1n.csv. I used</td></tr>
</table>

	these because I was sure I would be able to come up with what the output should be for each one. This way, I could easily check my output vs the expected output. By the end, my code's output matched the output I came up with own my own, so I was confident with the correctness of my code.
12	How you managed the code development - First, I planned out most of the program on paper. Once I had a confident idea of the data structures and code format I wanted to use, I was able to pretty much write the entirety of ntm_trace. I had to debug and refactor a few times, but planning out beforehand helped a lot of headaches.
13	Detailed discussion of results: - As I mentioned earlier, the output I got in my code matched what I expected. The composite test TM had the lowest degree of nondeterminism, and a-plus had the highest. When I tested the string '0011' with ntm_n1n, it rejected, which is expected. As expected, longer string took much longer to trace, with exponentially more transitions traced.
14	How team was organized The team was just me, so all work was done by me.
15	What you might do differently if you did the project again I think I would have created another test file so I could understand the topics a bit better. I grasped the project from the code's standpoint, but I think creating a csv file could have familiarized me with the format even more
16	Any additional material: N/A