Project 2 Readme Team JL

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"

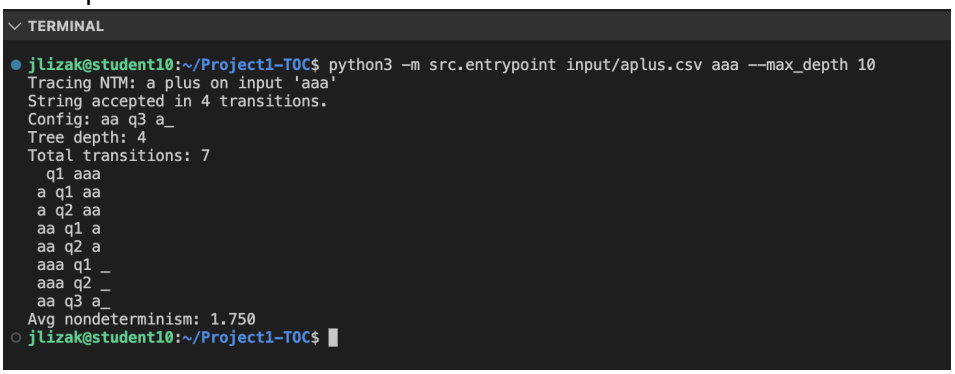
<table>
<tr><td>1</td><td>Team Name: JL</td></tr>
<tr><td>2</td><td>Team members names and netids

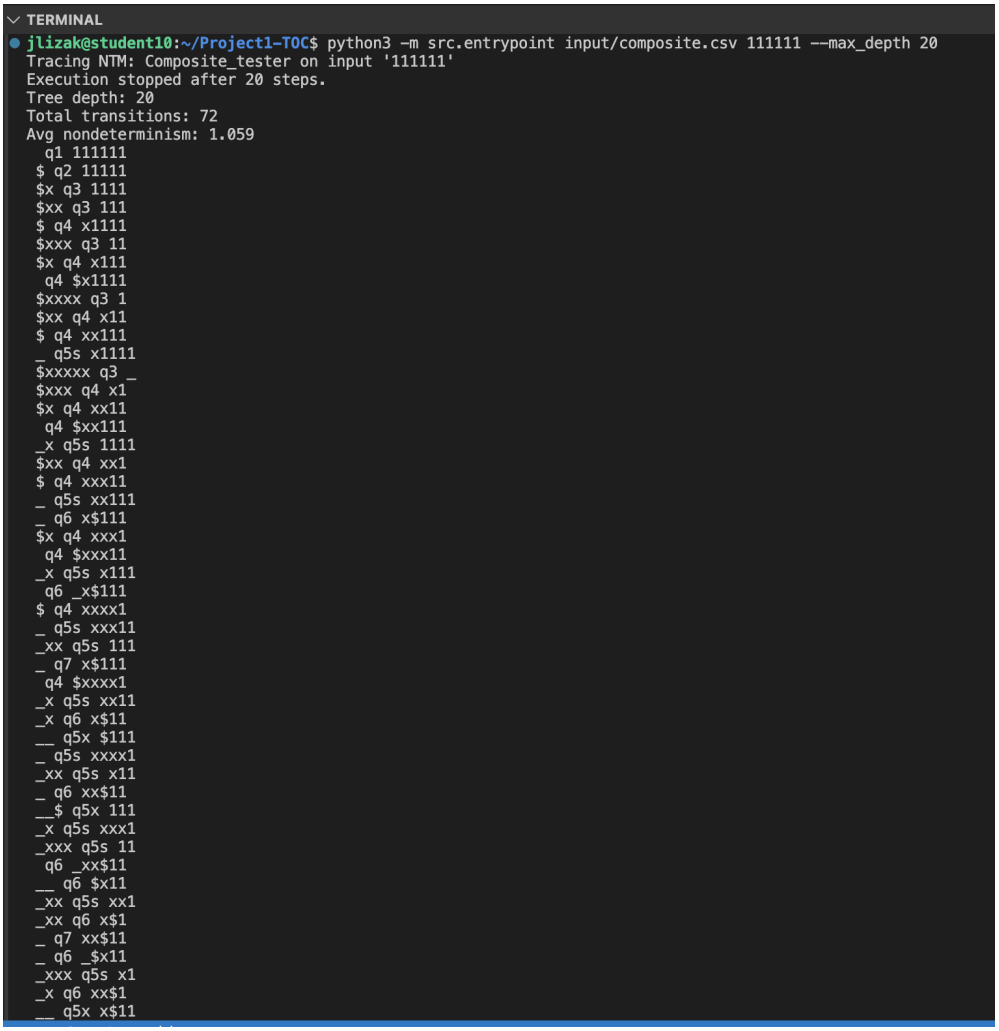
Julia Lizak
netID: jlizak</td></tr>
<tr><td>3</td><td>Overall project attempted, with sub-projects:
NTM Trace</td></tr>
<tr><td>4</td><td>Overall success of the project:
Good, it all works from what I can tell. I loaded the NTM definitions, tracked and printed the transitions and depth, and the full config. And I also accounted for the accep,t reject, and limit possibilities.</td></tr>
<tr><td>5</td><td>Approximately total time (in hours) to complete:
7-8</td></tr>
<tr><td>6</td><td>Link to github repository:
https://github.com/julializak/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>Main implementation, had the TODO statements, where I wrote majority of my code</td></tr>
<tr><td>entrypoint.py</td><td>Command line
Had to add the main function in there when I was calling to test it</td></tr>
<tr><td colspan="2">Test Files</td></tr>
<tr><td>aplus.csv</td><td>Given. NTM test for a language</td></tr>
</table>
</td></tr>
</table>

composite.csv	Given.NTM for composite checking
equal_01s.csv	Given. Strings with equal numbers of 0 and 1s

Output Files

No output files but here are some screenshots





```
_ q6 xx$11
__$ q5x 111
_x q5s xxx1
_xxx q5s 11
 q6 _xx$11
__ q6 $x11
_xx q5s xx1
_xx q6 x$1
_ q7 xx$11
_ q6 _$x11
_xxx q5s x1
_x q6 xx$1
__ q5x x$11
__ q7 $x11
_xxxx q5s 1
_ q6 xxx$1
___ q5x $11
___ q5s x11
_xxx q6 x$
 q6 _xxx$1
___$ q5x 11
___x q5s 11
_xx q6 xx$
_ q7 xxx$1
___ q6 $x1
___ q6 x$1
_x q6 xxx$
__ q5x xx$1
__ q6 _$x1
__ q6 _x$1
_ q6 xxxx$
___ q5x x$1
___ q7 $x1
___ q7 x$1
 q6 _xxxx$
____ q5x $1
____ q5s x1
____ q5x $1
jlizak@student10:~/Project1-TOC$
```

```
TERMINAL
jlizak@student10:~/Project1-TOC$ python3 -m src.entrypoint input/equal_01s.csv 0101 --max_depth 20
Tracing NTM: {w | w has the same number of 0's and 1's} Nondeterministic on input '0101'
String accepted in 13 transitions.
Config: _xxx_ qacc
Tree depth: 13
Total transitions: 21
  q0 0101
 _ q1 101
  q3 _x01
 _ q4 x01
 _ q5 x01
 _x q4 01
 _x q5 01
 _xx q1 1
 _x q3 xx
 _ q3 xxx
  q3 _xxx
 _ q4 xxx
 _ q5 xxx
 _x q4 xx
 _x q5 xx
 _xx q4 x
 _xx q5 x
 _xxx q4 _
 _xxx q5 _
 _xx q3 x_
 _xxx_ qacc _
Avg nondeterminism: 1.105
jlizak@student10:~/Project1-TOC$
```

Plots (as needed)

8	Programming languages used, and associated libraries:
	Python3 - no external libraries but included the given helper functions and everything
	
	`from src.helpers.turing_machine import (`
	`  TuringMachineSimulator,`
	`  BLANK,`
	`  DIR_L,`
	`  DIR_R,`
	`  DIR_S,`
	`)`
	
	`from src.helpers.argument_input import parse_inputs`
	`from src.helpers.turing_machine import TuringMachineSimulator`
	`from src.ktape_dtm import KTape_DTM`
	`from src.ntm_tracer import NTM_Tracer`

9	Key data structures (for each sub-project):
	
	Mainly representing the head, current state, and tape under or after the head. Also for the tree, used for the BFS traversal. And finally, dictionaries were mainly used and provided from the helper (write, move, next)
	
	`initial_config = ["", self.start_state, input_string]`
	`tree = [[initial_config]]`

10	General operation of code (for each subproject) Read machine and build initial configuration Use BFS to explore the nondeterministic branches Go level by level In the loop (For each config) Check if it is accept or reject Look up valid transitions Write Symbol, move head Regenerate child configs Count transitions and nondeterminism Stop when accept, or all branches reject, or max depth is reached
11	What test cases you used/added, why you used them, what did they tell you about the correctness of your code. **aplus.csv** Given to us. Gives a good idea of nondeterminism and an accept possibility. And it doesn't accept just after 1, it goes through 4 transitions so we can see that process and check it. Also, works with and shows branch expansions.

	composite.csv This was a deep tree. And tests that max depth component. And makes sure that it can handle larger transitions amounts without crashing **equal_01s.csv** This is more of a complex nondeterministic branches as well as sequence
12	How you managed the code development Used VSCode When I write code, I always test and test, step by step, so I avoid having a bunch of errors at the end and getting overwhelmed. First, just got an idea of the required config. Format from the instructions. (i.e. left, state, right). Then I kind of just started with a short and pretty straight forward code structure and adding to the run(), so loading the initial config, creating the tree, and loop through the level. I was really just getting an idea of what the code given to us was doing and then what I would need to implement (the commented steps were very helpful!!) Then, I just went step by step like how you had in the comments and tried to add print statements through to see what I was getting and if it aligns with the goal or expected outcome. It actually took me numerous print and debug statements at one point to try to find an error…I did not find it. So I had to veer to the AI route to help me find my error of why I was getting nothing printed out…just to find out it was a singular indentation that I missed…But after that, I went back to my normal process and really just finished everything up and make sure the print statements aligned with what the project wants.
13	Detailed discussion of results: It works! It fully reconstructs that search tree for each of the nondeterministic programs. And when I printed from the inside of the loop, it correctly matched the Turning machine behaviors. The transitions and nondeterminism values printed out as expected. And as we went into more test files (i.e. the composite one), the performance remained well performing and stable.
14	How team was organized Just me :)
15	What you might do differently if you did the project again Maybe just add some additional test files and also maybe print the actual accepting path rather than the whole tree. Also, maybe split the run() process into more functions and everything. But I was just trying to follow the TODO comments you laid out.
16	Any additional material: