Project 2 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: fdougher
2	Team members names and netids: Frazier Dougherty, fdougher
3	Overall project attempted, with sub-projects: Program 1: Tracing NTM Behavior
4	Overall success of the project: The project went well, I really enjoyed designing my own Turing machines but struggled to make them nondeterministic. It was very interesting to visualize how a turing machine works. For example, when I made my 2x0 and palindrome TM's it felt so simple on paper, but could effectively solve challenging inputs, which was neat to see as the head zigzagged across the tape.
5	Approximately total time (in hours) to complete: ● 12 ● ● (12/1/24) - 3.5 hours ● 12 2 - 1 hour ● 12/4 - 1:15 - 3 (2 hours) ● 12/6 - 11:30→ 2:45 (3.25 hours) ● 12/7 - 1-2:30 (1.5)
6	Link to github repository: https://github.com/fDoughertyND/TOCproject2
7	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.

File/folder Name	File Contents and Use
Code Files	
traceTM_fdougher.py	**Description**: My python script simulates a turing machine based on user specified csv files. It creates a visualization of the simulation based on an input string and tracks key metrics like depth, transitions, and average non-determinism. **Functions**: Read_ntm_file: this function converts the turing

machine from the CSV format into machine_info and transitions dictionaries which I can then process throughout the program.

Run_ntm: this is the heart of the program, given the ™ and string this function utilizes BFS to explore different configurations at different depths of my computation tree. It tracks current tape content, state, and head position, ensuring that no configuration is revisited (to avoid infinite loops). It can then print the results and path found toward the solution along with key metrics described above.

Print_path: This function prints the tape contents by showing the current state and head position at each step. Because we loop over the different configs, we essentially create timeline showing machines process.

Calculate_average_value_count: is a function made to calculate the non-determinism of a specified ™ . It divides the number of values with the number of keys to do so.

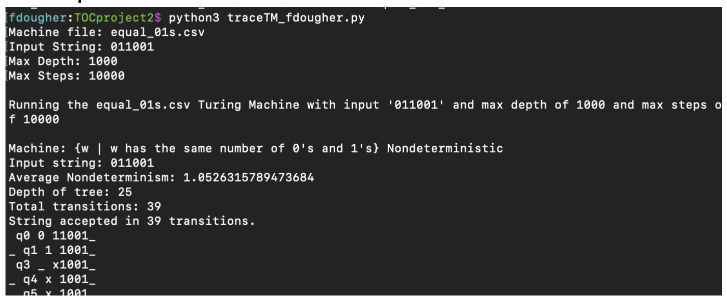
Main: The main function parses command-line arguments or user input.

Use:

python3 traceTM_fdougher.py TM.CSV input_String Max_depth max_steps

Or simply run: python3 traceTM_fdougher.py and the program will prompt you for the following information: Machine file, Input String, Max Depth, Max Steps

Examples:

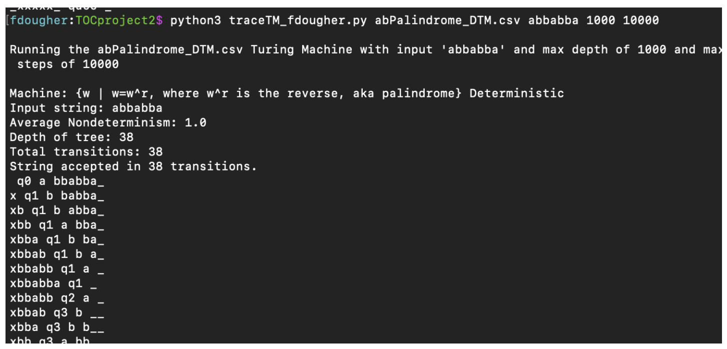


```
fdougher:TOCproject2$ python3 traceTM_fdougher.py
Machine file: equal_01s.csv
Input String: 011001
Max Depth: 1000
Max Steps: 10000

Running the equal_01s.csv Turing Machine with input '011001' and max depth of 1000 and max steps o
f 10000

Machine: {w | w has the same number of 0's and 1's} Nondeterministic
Input string: 011001
Average Nondeterminism: 1.0526315789473684
Depth of tree: 25
Total transitions: 39
String accepted in 39 transitions.
 q0 0 11001_
_ q1 1 1001_
 q3 _ x1001_
_ q4 x 1001_
  q5 x 1001
```

…



```
[fdougher:TOCproject2$ python3 traceTM_fdougher.py abPalindrome_DTM.csv abbabba 1000 10000

Running the abPalindrome_DTM.csv Turing Machine with input 'abbabba' and max depth of 1000 and max
 steps of 10000

Machine: {w | w=w^r, where w^r is the reverse, aka palindrome} Deterministic
Input string: abbabba
Average Nondeterminism: 1.0
Depth of tree: 38
Total transitions: 38
String accepted in 38 transitions.
 q0 a bbabba_
x q1 b babba_
xb q1 b abba_
xbb q1 a bba_
xbba q1 b ba_
xbbab q1 b a_
xbbabb q1 a _
xbbabba q1 _
xbbabb q2 a _
xbbab q3 b __
xbba q3 b b__
xbb q3 a bb
```

…

Test Files	
aPlus.csv	™ description which only accepts strings which are a sequence of a’s, nondeterministically
detAPlus.csv	™ description which only accepts strings which are a sequence of a’s, deterministically
abPalindrome_DTM.csv	™ description which accepts language {w | w=w^r, where w^r is the reverse, aka palindrome} deterministic
abPalindrome.csv	™ description which accepts language {w | w=w^r, where w^r is the reverse, aka palindrome} nondeterministic I can't figure out why but it will reject strings that are too short. I'm not super confident in my ability to make NTMs because of this.
2x0_DTM.csv	™ description which accepts language {w | w has twice the number of 0's as it does 1's} Deterministic
The following Test files are from NTM test files (shared class folder)	
Equal_01s.csv	™ description which accepts language {w | w has the same number of 0's and 1's} Nondeterministic
equal_01s_DTM.csv	™ description which accepts language {w | w has the same number of 0's and 1's} deterministic
abc_star.csv	™ description which accepts language a*b*c* Nondeterministic
Output Files	
Project 2 Analytics-fdougher (google doc)	I ran my program many times and tracked the results. I tracked the following metrics and compiled it into a table:

	● Machine ● String used ● Result (Accepted/rejected, ran too long) ● Depth of tree ● Number of configurations explored ● Average nondeterminism ● Comments
Plots (as needed)	
	Na

8	Programming languages used, and associated libraries: python
9	Key data structures (for each sub-project): machine_info (dictionary): this stores the metadata of the specified turing machine in a way I can access throughout the program. It tracks the name, states, input symbols, tape symbols, start state, accept state, and reject state. Transitions (dictionary): this dictionary maps a tuple of (state, input_symbol) to a list of possible transitions in the form of (next_state, write_symbol, direction). This allows for non determinism by letting the program transition to different configurations from one. Tape (list): this represents the content of the tape with an appended '_' representing the blank space. Configs_at_depth (list of lists): stores the configurations at each depth in the BFS search. The outer list is the level of depth, and each inner list represents the configurations at that level. visited_configs (Set): this set prevents the program from revisiting configurations thereby preventing infinite loops. next_level (list): tores the configurations that are generated in the current step and will be explored in the next step on the next depth. left_of_head, head_char, right_of_head (Strings) to help print the tape
10	General operation of code (for each subproject): traceTM_fdougher.py ● My python script simulates a turing machine based on user specified csv files. It creates a visualization of the simulation based on an input string and tracks key metrics like depth, transitions, and average non-determinism. ● It does so by first parsing CSV of the specified Turing Machines, then running the NTM and printing the results. To run the NTM, it employs BFS to search configurations at increasing depths. If a configuration leads to an accept state,

	the simulation halts and prints the path.
11	What test cases you used/added, why you used them, what did they tell you about the correctness of your code. For each different Turing Machine, I ran several input strings to verify correct output. Many of these turing machines are simple enough to verify by hand. For instance, equal_01 you can count the number of 0s and ensure it matches the number of 1s. I checked each input to ensure my algorithm ran properly. You can see specific inputs used in my project 2 analytics table.
12	How you managed the code development: I started by parsing a ™ csv and after ensuring that worked I started writing the Run_ntm function. This proved tricky, but once that was working I allowed for user specified input and prompting at the command line. I then developed several of my own turing machines by paper and then into csv format.
13	Detailed discussion of results: Nondeterminism increases complexity due to their ability to branch into multiple computation paths. This results in higher depths and configuration counts compared to their deterministic counterparts. Max Depth Exceeded was encountered with particularly large input strings and was especially prominent in NTM's, where the number of branches grows exponentially requiring large depths and steps to compute. Deterministic machines, being less complex in terms of transitions, are faster and more predictable, with a depth of tree and configurations explored directly corresponding to the length of the input and the machine's transition function. To exceed depth or step limits you would need an input string longer than the max_depth/steps which would be an instance of user error since you can clearly change these values. Average nondeterminism serves as an indicator of computational branching in nondeterministic machines, and higher values correlate with more computationally intensive tasks. For instance, I created the column Average Configs explored / depth of tree for machine and we can observe correlation between this and average nondeterminism.
14	How team was organized: I worked alone
15	What you might do differently if you did the project again: This project was vastly more successful than the first. I made sure to read, extremely carefully, the instructions and follow them closely so I wouldn't have to redo any of my work. I did get stuck figuring out the BFS algorithm for a long time, but that's a natural part of programming.
16	Any additional material: NA "These files may simply be machines from the book (particularly for simple DTMs for basic testing). However, for any nontrivial submission either converted from some

	non- Sipser-related source or created from scratch, extra credit will be considered for the creators, for a maximum of 6 points divided by the number of creators, and a maximum of 12 total extra credit points per student.” I would like my palindrome (both DTM and NTM) and 2x0_DTM turing machines to be considered for extra credit but am not sure where to submit them.