Readme_aarment2

Version 1 8/22/24

A copy of this file should be included in the github repository with your project. Change the teamname above to your own

1. Team name: **aarment2**
2. Names of all team members: **Alan Armenta**
3. Link to github repository: **https://github.com/ArmeAE22/ArmeAE22-aarment2_theory_proj2.git**
4. Which project options were attempted: **Project 01: Tracing NTM Behavior**
5. Approximately total time spent on project: **8-9 Hours**
6. The language you used, and a list of libraries you invoked: **Python, collections, csv, sys**
7. How would a TA run your program (did you provide a script to run a test case?): **yes, I provided input text files for each NTM csv file. The "usage" message that comes up when you enter incorrect input gives examples of how to run the program. If you want to use test strings not listed in the input test files, you can type your own into the program when it prompts you.**

```
python3 ./traceTM_aarment2.py abc_star.csv 20 < input_abc_star_aarment2.txt
```

8. A brief description of the key data structures you used, and how the program functioned.
    - **I used a deque for the BFS traversal of the possible paths the given NTM could take. I used a dictionary for easy lookup of the possible transitions from a certain state and input char. To store and traverse all the configurations, I used a list of lists as suggested in the project description.**
9. A discussion as to what test cases you added and why you decided to add them (what did they tell you about the correctness of your code). Where did the data come from? (course website, handcrafted, a data generator, other)
    - **I used 3 NTM csv files given by the instructor in an announcement and an extra one from the assignment description. For the test string files, I used AI to generate test strings. For each NTM, I used 4-5 test strings that tested acceptance, number of transitions, depth of tree of configurations, and how many steps it took to be rejected or accepted.**
10. An analysis of the results, such as if timings were called for, which plots showed what? What was the approximate complexity of your program?
    - **To analyze the results of each NTM, I included a table as suggested by the instructor. The table includes which machine was used, what input string, result (accept, reject, timed out), depth, # configurations explored, and average non-determinism (# configurations / depth).**
11. A description of how you managed the code development and testing.
    - **The project description gave lots of hints of how to handle code development. Refreshing on BFS and adjusting the function to handle**

traversing the possible configurations was probably the hardest part of the project. What took most of my time was handling the inputs and outputs. I followed along with the project description to get an idea of what kind of inputs I needed, but I needed a review on parsing csv files and using command line arguments. For output, I followed the same process as the inputs. For testing, I used the NTM files that the professor provided in an announcement. These were simple NTMs that I could easily find valid input strings for, such as a+, equal 0s and 1s, etc. I compared my own traversal results with those of the program.

12. Did you do any extra programs, or attempted any extra test cases
    - **I did not do any extra programs, but I did use several test cases. I used 4 different NTMs, and I had 4-5 test strings for each of them. The strings tested if the program would find an accepting state, the path to an accepting state, and the number of transitions simulated.**