Project Tracing NTM Behavior Readme Team Alliecat

1	Alliecat
2	Allie Sproul asproul
3	Tracing NTM Behavior
4	Overall success of the project: Good - it took less time than the first project and I understand the material better
5	Approximately total time (in hours) to complete: 4
6	https://github.com/alliesproul/theory_alliecat
7	List of included files:

File/folder Name	File Contents and Use
Code Files	
traceTM_alliecat.ipynb	Contains Tracing NTM Behavior
Test Files	
data_containsb_alliecat.csv	Contains header and transitions for NTM that checks if there is at least one b
data_aplus_alliecat.csv	Contains header and transitions for NTM that checks if there is at least one a and only a's
data_abcstar_alliecat.csv	Contains header and transitions for NTM that checks for a least one a, b, or c in string in that order (a*b*c*)
Output Files	
output_aplus_alliecat.txt	Contains output for a+ machine in a table
output_containsb_alliecat.txt	Contains output for contains b machine in a table
output_abcstar_alliecat.txt	Contains output for a*b*c* machine in a table

8	Programming languages used, and associated libraries: Python: csv, collections, tabulate
9	Key data structures: transitions - dictionary (Stores the state transition table for the NTM, maps a pair (state, read_character) to a list of possible transitions) queue - deque (Stores configurations to be explored, maintaining the order of discovery)

	parent_map - dictionary (Tracks the parent of each configuration to reconstruct the path leading to an accept state) configurations_per_level - list (Tracks the number of configurations generated at each depth of the BFS tree) results - list (Contains results to be printed in table)
10	General operation of code: this code simulates an NTM with a BFS strategy. It parses the machine description file utilizing the csv module. The file contains the states, tape symbols, transitions, start, accept, and reject state. From this data in the header it creates a transition table where keys are state/input pairs and values are lists of possible transitions. The code simulates an NTM on the input string and creates the configuration, queue, parent map, and configurations per level. Within the simulation at each iteration it processes the current configuration, checks the halting conditions, retrieves possible transitions, simulates each transition, and tracks statistics. The simulation halts if it reaches accept and the full path of configurations is printed, the result is rejected, or the total transitions exceed the limit. It then prints out all the results.
11	What test cases you used/added, why you used them, what did they tell you about the correctness of your code: I used three different machine files: aplus, containsb, and abcstar. For each machine I test a few different strings to see different degrees of non-determinism. These test files showed my code is correct because I could mentally decide if the machine should accept or reject the input string. For example if I test “abbac” on my machine aplus, which is only supposed to accept strings with one or more a’s (and only a’s), my code should reject it. If I pass through the string “aaa”, the machine should accept it.
12	How you managed the code development: I managed the code development by referring to the first project to see how I parsed the input files. While developing the code I used print statements to make sure my code was working properly at each stage. I started small by parsing, then moving on to the tape movement logic. Next I worked on the simulation function, and ended by creating the path tracing function.
13	Detailed discussion of results: for each machine file I have different results depending on the input string. The general layout of the output is the name of the machine, the input string, the result (accept or reject), depth of tree, total transitions, degree of non-determinism, depth that string was accepted at (if it was), and the acceptance path (if accepted. The depth of the tree indicates the number of steps or levels explored during the breadth-first search (BFS) of the configuration space. I calculated the degree of non-determinism by dividing the sum of the configurations by the length of configurations (depth). The acceptance path follows this order: the string to the left of the head, the current state, the character being read under the head, then the string to the right of the head.
14	How team was organized: N/A
15	What you might do differently if you did the project again: If I did the project again I would work on understanding the material better first before diving head first into the project. I think I would have saved time if I understood NTMs better initially instead of learning more about them through the project.
16	Any additional material: