1. The team name
    - jtran4

2. Who were the other team members.
    - There are no other team members just me, myself, and I.

3. Under whose netid is the readme-team.pdf, code, and other material saved.
    - jtran4

4. How much time did you personally spend on the project, and what did you do?
    - I spent 8.5 hours on this project in total.
        - 4 hours were spent creating the code from scratch.
        - 2 hours were spent in office hours getting help with the code.
        - 1.5 hours were spent on creating the test cases for the code
        - 1 hour was spent on writing the reflection for the “readme-jtran4” and “teamwork-jtran4” files.

5. What did you personally learn from the project, both about the topic, above programming and code development techniques, and about algorithms.
    - About Programming:
        - Understanding Turing Machines:
            - The project involves implementing a Non-deterministic Turing Machine (NTM). This would deepen one's understanding of theoretical computer science concepts, specifically Turing machines.
        - File Parsing:
            - The code includes parsing a file that represents the configuration of the Turing machine. This experience could improve skills in file reading and parsing.
        - Data Structures:
            - The use of dictionaries and lists for storing and managing transitions and states showcases practical application of data structures in Python.
        - Dynamic Tape Representation:
            - The representation of the tape using a custom Tape class with dynamic left and right portions, and a head, is a practical application of object-oriented programming.
        - Interactive User Input:
            - The program interacts with the user, taking input for the Turing machine file, input strings, and maximum steps. This adds a user-friendly dimension to the program.
        - Output File Handling:
            - The code creates and writes to an output file, enhancing the user experience by providing a record of the machine's execution.
        - Application of Graph Theory:

        - The BFS traversal for tracing machine execution can be seen as an application of graph theory concepts, as the configurations form a tree-like structure.
    - Documentation:
        - The code includes comments and docstrings, contributing to its overall readability and making it easier for others (and the author) to understand the implementation.
- About Code Development Techniques:
    - Algorithm Design:
        - The implementation of the breadth-first search (BFS) algorithm for tracing the Turing machine execution demonstrates algorithmic thinking in solving computational problems.
    - Verbose Output and Logging:
        - The code provides detailed output, including the depth of the tree of configurations, total transitions, and the step-by-step execution of the machine. This practice is useful for debugging and understanding the machine's behavior.
    - Library Usage:
        - The code utilizes external libraries such as tqdm for progress bars. This demonstrates an awareness of available libraries and the ability to integrate them for improved functionality.
    - Error Handling:
        - The code handles cases where the input file is not formatted as expected, providing a basic form of error handling.
    - Modularization:
        - The code is reasonably modular, with distinct classes and functions for tape representation, NTM, and the main program. This promotes code readability and maintainability.
    - Learning from Results:
        - The code prints and writes results for each input string, facilitating understanding and analysis of the machine's behavior.
- About Algorithms:
    - Limits and Execution Control:
        - The program includes control mechanisms for stopping execution after a specified number of steps, adding a practical aspect related to runtime control.

6. In your own words, how did the team dynamics work? What could be improved? (e.g. did you use github and if so did it help, did you meet frequently enough, etc.) For single person teams, reflect how you would have worked if there was a teammate.

- I finally used a github to organize all of my files. It was very helpful to organize all of my code into one place as well as my readme and team files. I found it especially convenient when I submitted all of my files to the Canvas dropbox because all of my code is in one place, and this will be helpful when I need to refer to this code in the future.

- I worked independently, but if I were in a team, I would have regular team meetings which would have been essential to discuss progress, share insights, and ensure that everyone was on the same page. Having a teammate would have allowed for a division of tasks, potentially making the project more efficient. Additionally, peer code reviews and feedback would have been valuable to improve the overall code quality. Overall, while I managed on my own, team communication and collaboration could have enhanced the project.

7. From your own perspective, what was the role of each team member, and did any member greatly exceed expectations (and if so how/why), or vice versa.
This teamwork file should be submitted even for single student projects, although clearly in that case no comments on how the teaming efforts are needed.

- I derived a sense of personal fulfillment from investing considerable effort in completing this assignment. I conscientiously fulfilled all stipulated requirements, ensuring the assignment's timely submission. This endeavor spanned several days, encompassing coding sessions, the creation of output files, and the development of graphical representations to visually convey the outcomes of the code.
- Surprisingly, I exceeded my initial expectations in the course of this project. It represented my inaugural venture into independently tackling a computer science assignment, despite the availability of collaborative options within a group setting. This experience proved to be a significant opportunity for my personal development and learning.