Data Structures and Algorithms, Course Project 2019

Very important – Read before starting

- The deadline for completing and submitting your project is strictly Friday 31st May 2019 at 18:00.
- VLE will be set up to not accept late submissions meaning that you will get zero marks if late.
    - Please plan ahead (it is recommended that you try and upload and verify your work a day before).
    - Technical problems, internet connectivity issues, lost backups, cats eating laptops, etc… are not valid excuses.
- You must complete the project completion form (shown later) and include it in your report. Submissions without the statement of completion will not be considered.
- You must complete a plagiarism declaration form and include it in your report. Submissions without the form will not be considered.
- Projects must be submitted using VLE only. Physical copies or projects (including parts of) sent by email will not be considered.
- For your convenience, a draft and final submission area will be set up in VLE. Only projects submitted in the final submission area will be graded. Projects submitted to the draft area are not considered.
- It is suggested that after submitting your project, you re-download it and check it just in case. It is your responsibility to ensure that your upload is complete, valid, and not corrupted. You can re-upload the project as many times as you wish within the deadline.
- Your project must be submitted in ZIP format without passwords or encryption. Project submitted in any other archiving format will not be considered.
- The total size of your ZIP file should not exceed 38 megabytes.
- Your submission should include your report in PDF format, your source code, and executable file(s).
- It is expected that you submit a quality report with a proper introduction, discussion, evaluation of your work, and conclusions. Also, make sure you properly cite other people's work that you include in yours (e.g. diagrams, algorithms, etc…).
- In general, I am not concerned with which programming language you use to implement this project. However, unless you develop your artifact in BASIC, C, C++, Objective C, Swift, Go, Pascal, Java, C#, Matlab, or Python, please consult with me to make sure that I can correct it properly.
- This is not a group project.
- Plagiarism will not be tolerated.

Project

Implementation:

1. Construct a deterministic finite state automaton A according to the following recipe:
    a. Create n states, where n is a random number between 16 and 64 inclusive.
    b. Randomly label every state as either accepting or rejecting.
    c. Every one of the n states has two outgoing transitions leading to two other random states; one transition is labelled with the symbol a, and the other with the symbol b. Transitions from a state to itself are allowed.
    d. Choose any random state as the starting state.
2. Compute the depth d_A of A. The depth of an automaton is defined as the maximum over all states of the length of the shortest string which leads to that state (hint: you can use breadth-first search to get this). Output:

    Print the number of states in A.
    Print the depth d_A of A.

3. Minimise the automaton A using either the Moore or Hopcroft minimization algorithm to obtain a new automaton M.
4. Compute the depth d_M of M. Output:

    Print the number of states in M.
    Print the depth d_M of M.

5. Generate 100 random strings over the alphabet $\{a, b\}$. Each string must have a random length between $[0 \dots 128]$ – note that the empty string may be included. Use the automaton to classify each of the 100 random strings as either accepting or rejecting. Output:

    String_1 :: Accepting/Rejecting
    String_2 :: Accepting/Rejecting
    …
    String_{100} :: Accepting/Rejecting

6. Learn what strongly connected components (SCCs) in graphs are and implement Tarjan's algorithm, for finding the strongly connected components in M. Output:

    Print the number of strongly connected components in M.
    Print the size of (number of states in) the largest SCC in M.
    Print the size of (number of states in) the smallest SCC in M.

Report:

- When implementing the data structure to represent a DFA you'll probably choose between an adjacency list or an adjacency matrix. Properly justify why you chose the implementation you did.
- Discuss (don't just state it) the time complexities of (a) the minimization algorithm you chose, (b) Tarjan's algorithm, and (c) the string parser (classifier).
- Design and present an evaluation to determine whether your implementation of the required algorithms works properly.

Marking Breakdown

Item	**Percentage Weight**
Constructed the basis automaton A	5%
Minimised A to obtain M	20%
Computed the depth of A and M	10%
Random String Classification	20%
Tarjan's algorithm	20%
Report Quality	15%
Evaluation	10%

Statement of completion – MUST be included in your report

Item	**Completed (Yes/No/Partial)**
Constructed the basis automaton A	
Computed the depth of A	
Minimised A to obtain M	
Computed the depth of M	
Implemented Random String Classification	
Implemented Tarjan's algorithm	
Evaluation	