CS F351: Theory of Computation

Assignment- 1

Marks: 20 (weightage 10%) **Due date: 15 Oct 2022 11:59 PM**

Consider an integer grid G of size $n \times n$ (where n is a positive integer) such that the coordinates of each point are non-negative integers (here G is formed by the intersection of lines $x = 0, x = 1, \ldots, x = n$ and $y = 0, y = 1, \ldots, y = n$), see Figure 1 for an illustration. Each point (i, j), $0 \leq i \leq n$ and $0 \leq j \leq n$, is called a *grid point*.

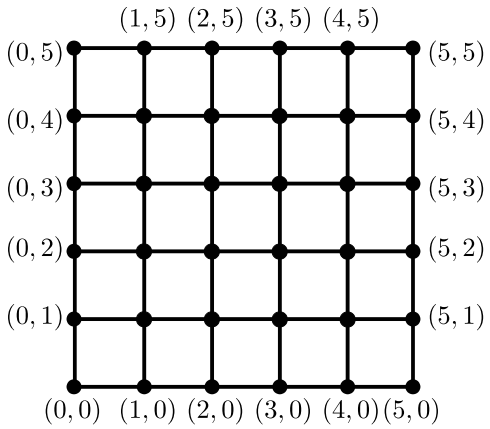

Figure 1: Grid of size 5×5

A robot is placed at some grid point on G, and the robot's movement depends on the instructions given to it. We can give only one of the two instructions, 0 (*horizontal move*) or 1 (*vertical move*), to the robot. Suppose the robot is standing at (i, j). If it reads the instruction 0 then it will move to either $(i-1, j)$ or $(i+1, j)$ (horizontal move) and if it read the instruction 1 then it will move to either $(i, j-1)$ or $(i, j+1)$ (vertical move). However, it cannot move to a point outside the grid G, i.e., suppose the robot is standing at $(0, 0)$ then it can move to one of the points $(1, 0)$ or $(0, 1)$ depending on the instruction it receives.

Suppose that the robot current position is (i, j) where i and j are non-negative integers. For a given *instruction* string $w \in \{0, 1\}^*$ (which can be seen as a sequence of instructions to the robot), the robot moves to the other points in the grid G by using the sequence of instructions given in the instruction string w. For example, for $w = 01100$, from the point (i, j), the robot first makes a horizontal move, a vertical move, a vertical move, a horizontal move, and finally, a horizontal move (in the same order).

Define a language L as the set of all instruction strings $w \in \{0, 1\}^*$ such that the robot starts at grid point $(0, 0)$ and reaches to grid point (n, n) by following the sequence of instructions given in the string w.

One can design a non-deterministic finite automata (NFA) M for the language L by considering states corresponding to the grid points on G, the start state corresponding to grid point $(0, 0)$, the final state corresponding to gird point (n, n) and possible moves can be defined as transitions. Further, $\Sigma = \{0, 1\}$.

1 Tasks

Write a C program that performs the following tasks:

Task 1 (10 marks): Given an instruction string $w \in \{0,1\}^*$, verify that w is accepted by NFA M or not? i.e., Is $w \in L$ or not?

Use **multiprocessing** and shared memory to create a simulator for the same. i.e., suppose you are at a state p, and the next input instruction is a (here a is either 0 or 1); if there are more than one transition exists for this case, you have to create sub-processes such each takes a possible transition and proceeds independently. If any process reaches the final state after reading the last instruction in w, you announce that M accepts w, and the remaining processes may stop their execution.

Note: if you perform the task with simple recursion you will not fetch any marks.

Task 2 (10 marks): For the NFA M (defined above), give an equivalent minimal deterministic finite automata (DFA). Here, minimal DFA means all the states in the DFA are distinguishable.

2 Input and Output formats

Input: You will be given a text file "input.txt" which contains two rows; the first row contains the value of n (it will be a positive integer), and the second row contains a string, w, of 0's and 1's.

Output:

Your program must generate two text files namely $yourBITSid\_t1.txt$ (output of task 1) and $yourBITSid\_t2.txt$ (output of task 2).

Format of output file $yourBITSid\_t1.txt$: (will be uploaded soon).

Format of output file $yourBITSid\_t2.txt$: Assume that your resultant DFA for task 2 has t states and assume the labeling of the states are $0, 1, 2, \ldots, t-1$ such that 0 represents the start state. The file $yourBITSid\_t2.txt$ contains t columns (one for each state) and $2t+1$ rows. One can see this as a 2-dimensional array A of size $(2t+1) \times t$ (assume that the row and column numbering start with 0).

1. 0-th represents the set of final states.

   $$A[0][j] = \begin{cases} 1 & \text{if state } j \text{ is one of the final states} \\ 0 & \text{otherwise} \end{cases}$$

2. Rows from 1 to t represents the transition function corresponding symbol 0.

   $$\text{For } i = 1, 2, \ldots, t, \qquad A[i][j] = \begin{cases} 1 & \text{if } \delta(i-1, 0) = j \\ 0 & \text{otherwise} \end{cases}$$

   where δ is the transition function of your DFA.

3. Rows from $t+1$ to $2t$ represents the transition function corresponding symbol 1.

   $$\text{For } i = t+1, t+2, \ldots, 2t, \qquad A[i][j] = \begin{cases} 1 & \text{if } \delta(i-1-t, 1) = j \\ 0 & \text{otherwise} \end{cases}$$

   where δ is the transition function of your DFA.

3 Other Instructions

1. The program must be written in C, and should be able to compile with gcc-11 on Ubuntu 22.04.
2. For your benefit, you are provided a starter template for writing your code. The same can be available at Assignment 1 Starter.
3. The instructions for using the template is available in the repositories README.md file.

4 Submission guidelines

1. You need to submit a single C program which performs the tasks mentioned above.
2. The name of the submission file must be of the form $yourBITSid.c$
3. Late submissions will fetch penalty of 20% per day up to two days. No late submissions will be considered after two days from the due date.