# Homework 6 Assigned: November 18, 2015

# Due: 11:59:59 PM, Wednesday, December 9, 2015

Task 0. Write a Java program named DFSASimulator.java that simulates a DFSA.

Given a machine description and input, determine if the input is accepted by the machine.

We will use this encoding of a DFSA:

Assume an alphabet sigma = {0,1}

Assume states are numbered starting at 1.

<at least one 0 that specifies the number of states>

<a single 1 acts as a delimiter>

<number of 0's specifies where the first sate transitions to on 0>

<a single 1 acts as a delimiter>

<number of 0's specifies where the first sate transitions to on 1>

<a single 1 acts as a delimiter>

<number of 0's specifies where the second sate transitions to on 0>

<a single 1 acts as a delimiter>

<number of 0's specifies where the second sate transitions to on 1>

:

:

<two 1's act as a delimiter>

<number of 0's specify an accepting state>

<a single 1 acts as a delimiter>

<number of 0's specify an accepting state>

:

:

<two 1's act as a delimiter>

The remaining string of 0's and 1's is a string that will be accepted or not.

The output of your program will be "Accept" if the string following the encoded machine is accepted by the machine and "Reject" otherwise.

Example execution:

Java DFSASimulator

0010010101001100111111101111

Accept

Java DFSASimulator

00100101010011001111111011110

Reject

Submit well documented DFSASimulator.java and DFSASimulator.class in a zipped directory named Project6Task0.

Task 1. Write a Java program that simulates a Turing Machine.

The Turing machine that we will simulate can be formally defined as M = (Q,,,,q0,B): where

Q, a finite set of states. For this program Q = {0,1,2,…n-1} and is selected

by the client programmer.

 = { 0,1,B } is the finite set of allowable *tape symbols*

B, a symbol of , is the *blank*

 = { 0,1}, a subset of  not including B, is the set of input symbols

: Q x   Q x  x {L, R} ( may, however, be undefined for some

arguments)

q0 = 0 is the initial state

Our tape will be infinite in both directions. To simulate this, define an array of size 40 and set the initial read/write head to 20 (the middle of the tape). Our simulation will only work for machines that stay within this range.

Suppose that we wanted our program to simulate the machine with the following values for delta:

(q0,0) = (q0,1,R)

(q0,1) = (q0,0,R)

(q0,B) = (q1,B,L)

This machine reads the tape from left to right and replaces any 1’s with 0’s and any 0’s with 1’s. It stops, by entering the halt state, when it encounters a B in the input. We will call this a one state machine (there is always an additional Halt state.)

Your task is to write a Java program (Turing1.java) that simulates this machine. The main routine of your solution will look like the following:

public static void main( String args[]) {

Turing machine = new Turing(1); // This machine will have one state.

// Note: There is an additional halt state.

// The values on the input tape are set to

// all B’s.

Transition one = new Transition('0',Transition.RIGHT, 0);

Transition two = new Transition('1',Transition.RIGHT, 0);

Transition three = new Transition('B', Transition.LEFT,1);

machine.addTransition(0, '0', two);

machine.addTransition(0, '1', one);

machine.addTransition(0, 'B', three);

String inTape = "11111100010101"; // The leftmost value of inTape will be

// placed under the read/write head.

System.out.println(inTape);

String outTape = machine.execute(inTape);

System.out.println(outTape);

}

And the output of this program is shown below:

C:\McCarthy\www\90-723\TuringMachine>java Turing

11111100010101

BBBBBBBBBBBBBBBBBBB00000011101010BBBBBBB

The transition Transition one =   new Transition('0',Transition.RIGHT, 0); along with the add method machine.addTransition(0, '1', one); assigns to state 0 a transition on '1'. This transition replaces the 1 with a 0, moves the read head right and returns to state 0.

Submit a zipped directory containing all of the Java code that you used to simulate the machine. For grading, we must be able to edit the main java file Turing1.java so that we can test your program against various input tapes. The main routine will be identical to the main routine above.

Task 2. Write a Java program (Turing2.java) so that it simulates the following machine:

Proper subtraction m – n is defined to be m – n for m >= n, and zero for m < n. The TM

M = ( {q0,q1,...,q6}, {0,1}, {0,1,B}, , q0, B, {} )

defined below, if started with 0m10n on its tape, halts with 0m-n on its tape. M repeatedly replaces its leading 0 by blank, then searches right for a 1 followed by a 0 and changes the 0 to a 1. Next, M moves left until it encounters a blank and then repeats the cycle. The repetition ends if

1. Searching right for a 0, M encounters a blank. Then, the n 0’s in 0m10n have all been changed to 1’s, and n+1 of the m 0’s have been changed to B. M replaces the n+1 1’s by a 0 and n B’s, leaving m-n 0’s on its tape.
2. Beginning the cycle, M cannot find a 0 to change to a blank, because the first m 0’s already have been changed. Then n >= m, so m – n = 0. M replaces all remaing 1’s and 0’s by B.

The function  is described below.

(q0,0) = (q1,B,R) Begin. Replace the leading 0 by B.

(q1,0) = (q1,0,R) Search right looking for the first 1.

(q1,1) = (q2,1,R)

(q2,1) = (q2,1,R) Search right past 1’s until encountering a 0. Change that 0 to 1.

(q2,0) = (q3,1,L)

(q3,0) = (q3,0,L) Move left to a blank. Enter state q0 to repeat the cycle.

(q3,1) = (q3,1,L)

(q3,B) = (q0,B,R)

If in state q2 a B is encountered before a 0, we have situation i

described above. Enter state q4 and move left, changing all 1’s

to B’s until encountering a B. This B is changed back to a 0,

state q6 is entered and M halts.

(q2,B) = (q4,B,L)

(q4,1) = (q4,B,L)

(q4,0) = (q4,0,L)

(q4,B) = (q6,0,R)

If in state q0 a 1 is encountered instead of a 0, the first block

of 0’s has been exhausted, as in situation (ii) above. M enters

state q5 to erase the rest of the tape, then enters q6 and halts.

(q0,1) = (q5,B,R)

(q5,0) = (q5,B,R)

(q5,1) = (q5,B,R)

(q5,B) = (q6,B,R)

Submit a zipped directory containing all of the Java code that you used to simulate the machine. For grading, we must be able to edit the main java file Turing2.java so that we can test your program against various input tapes. The main routine should look almost identical to the main routine above. The only difference should be the size of the machine and the assignments to delta via addTransition().

Task 3. Design a Turing machine that reads a series of one’s and makes a copy of those one’s to the same tape.

There will be a single B separating the two lists on the final output. For example, if the input is 11111 then the output will be B11111B11111B. If the input is 1 then the output will be B1B1B. If the input is the empty string then the output will be a blank tape (all B’s). If the input is 1111111111 then the output will be B1111111111B11111111B. We are using the ‘B’ symbol to represent a blank tape symbol. So, we are assuming that there are an infinity of B’s on each side of the input. Do not worry about the number of blanks appearing before or after the input. Don’t worry about the number of B’s appearing before or after your string on output. The only blank we care about is the one between the two strings of 1’s. You may assume that the read head is positioned on top of the first 1 in the input (or, on a blank if the input is empty.)

Write a Java program (Turing3.java) so that it simulates your new machine. Of course, you should reuse the work you did in Task 1.

Submit a zipped directory containing all of the Java code that you used to simulate the machine. For grading, we must be able to edit the main java file Turing3.java so that we can test your program against various input tapes. The main routine should look almost identical to the main routine above. The only difference should be the size of the machine and the assignments to delta via addTransition().