**Project 2 & 3**

DUE: December 9th, 2019

Professor: Stephen Lucci

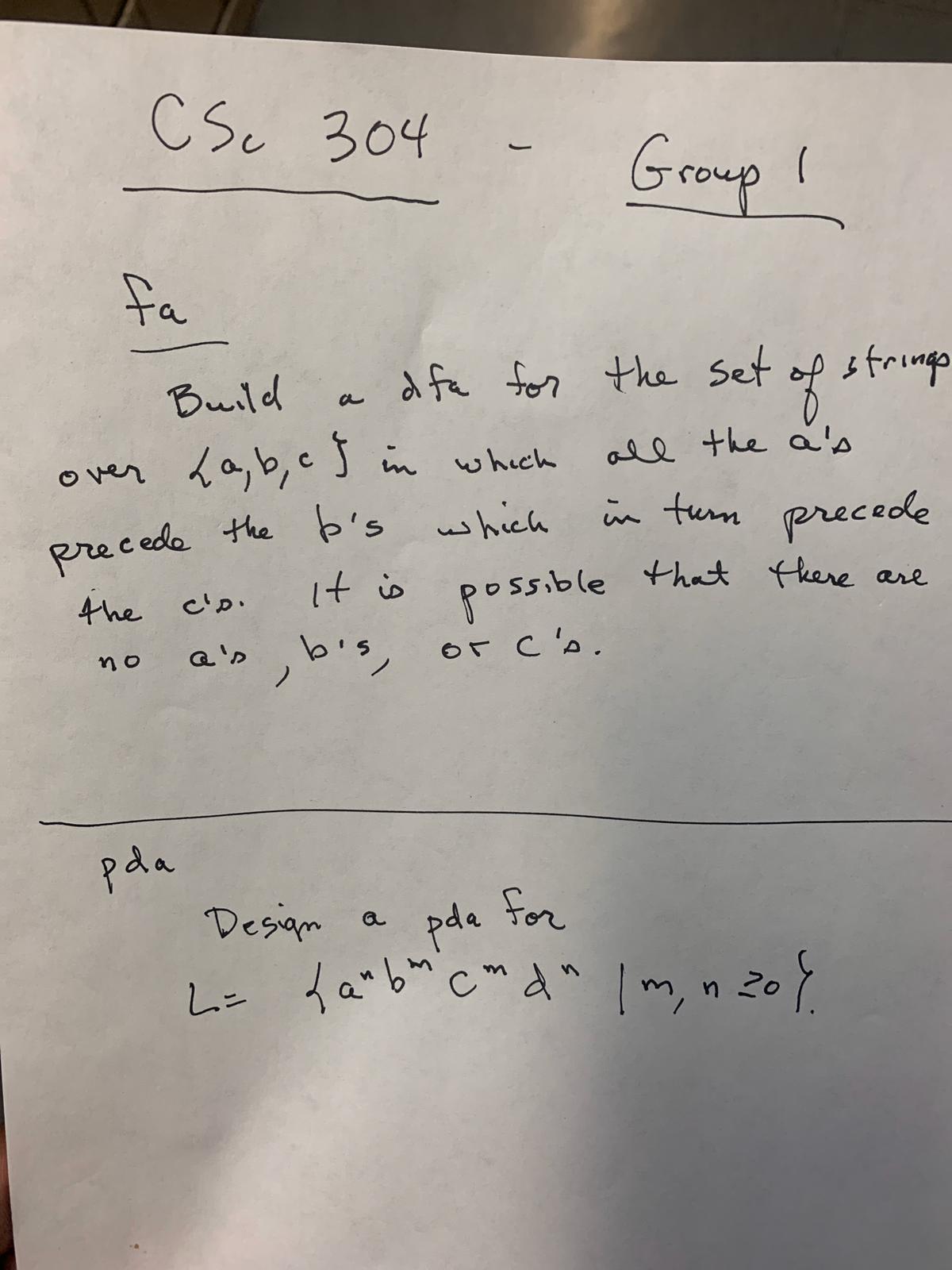
CSC 30400

**Cool Team**

Pavlo Aleksyeyev

Hasibul Islam

Warin Wohab



DFA:

Design a DFA for the set of string over {a,b,c} and it is possible that there are no a’s or no b’s or no c’s (meaning an empty input. The a’s must come before b’s and b’s must come before c’s. We were able to successfully design the DFA with four states. Initial state is q0. If q0 reads an a it will stay in state q0 until it reads a b or a c. If it reads a b then the machine goes to state q1 and it will stay in state q1 until it either encounters a c or an a. If it encounters an then the machine goes to state q3, the state to collect all the incorrect languages. If the machine encounters a c then it moves to state q2 and will stay there or move to incorrect collection state if it encounter a or b.

Machine Complexity:

At every step, the machine is reading the input by each character. To read all the characters of an accepted word over the language (a,b)\*, the machine runs twice for each step:One step to read the input and another step to move to the next state. So, the total time needed is 2\*|w|, |w| = length of the word.

Runtime: Theta(|w|)

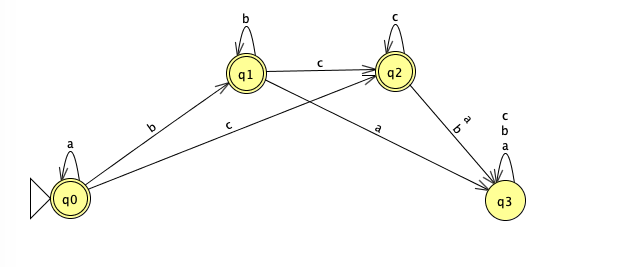
While the machine doesn’t write anything, but it requires the memory for the input string (word) with length |w| exactly that amount of space.

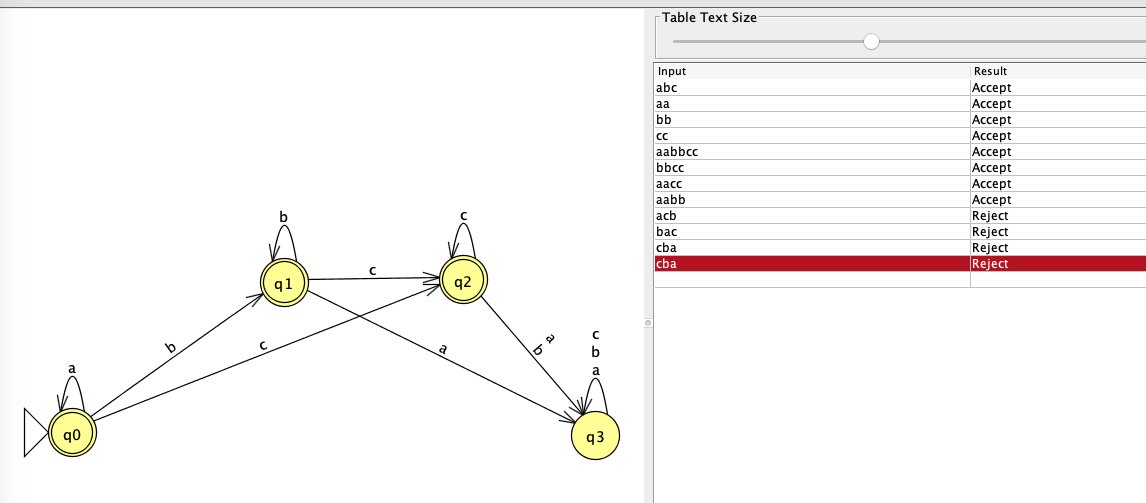
Space: Theta(|w|) Where w is the length of input tape

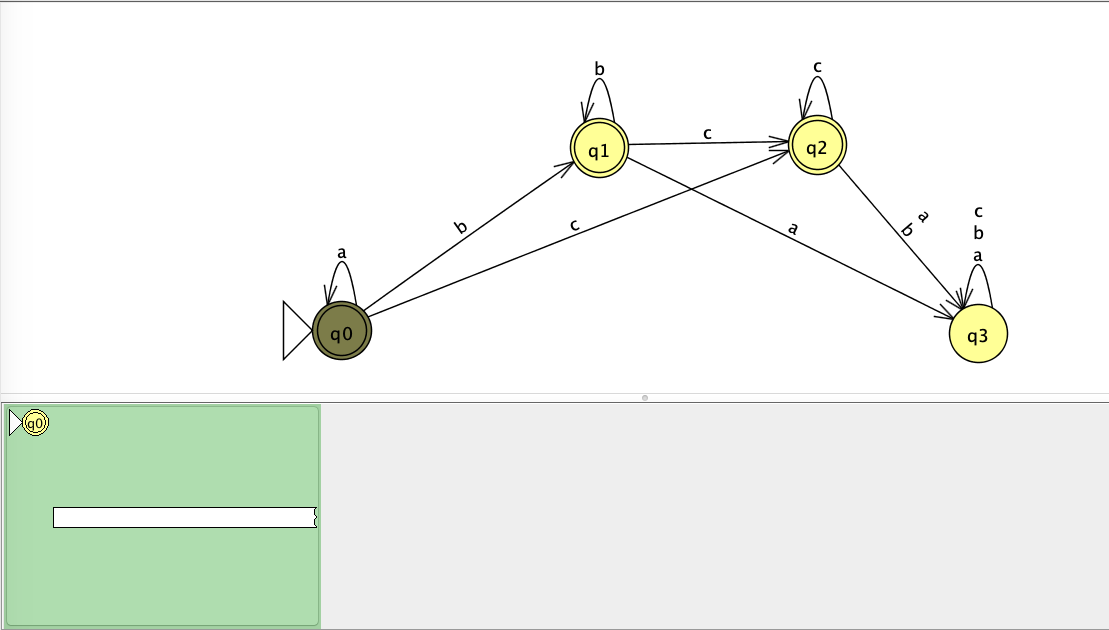
Input tape State transition for DFA:

The machine reads the words. If the word is empty, the final state is q0 as accepted.

When the word read three consecutive a’s at the beginning, it halts at state q3. Else, it will be at state q4 as the word is being accepted.







PDA: Design a PDA for the language L = {a^n b^m c^m d^n | m, n >= 0}

For this Automata we needed to make sure that an equal number of a’s and d’s and an equal number of b’s and c’s were being accepted, in the order abcd. To begin we first had to set an initial marker in the bottom of the stack. Next, we needed to make sure that an input with any amount of a’s will add the letter a to the stack. Then we move on to b’s and add b’s to the stack. Next, to make sure that the number of b’s and c’s are equal, we pop one b for every c in the input. If they were not equal, the input gets rejected. If it is equal, we then do the same for the d’s and make sure to pop off one a per d from the stack. If they were equal, the last step would be to empty the stack of the initial marker and return the accepting state. This was our first design. However, we had to add what would happen if the input started with b’s and ended with c. To make this accepted, we added the last step to empty the stack of the initial marker and return the accepting state. Also, we had to add an arc that would allow there to be 0 b’s and c’s.

Machine Complexity of PDA:

Runtime:

From the input we can see that the machine reads

a = n b = 2n c = m d = m

At each input, the machine reads in one step and writes into the stack in another step. So, for n input length, there are 2n steps required. Total steps for this machine for a and b = 2n + 2\*2n = 6n and for c and d = 2m + 2m = 4m. Assuming n and m to, the runtime for this machine = O(6n+4m) = O(n+m)

Space:

The machine writes into the stack for each a and c. The total space required for this machine in stack is max(n,m). So, the space complexity of the PDA is O(max(m,n)) = O(m + n).

