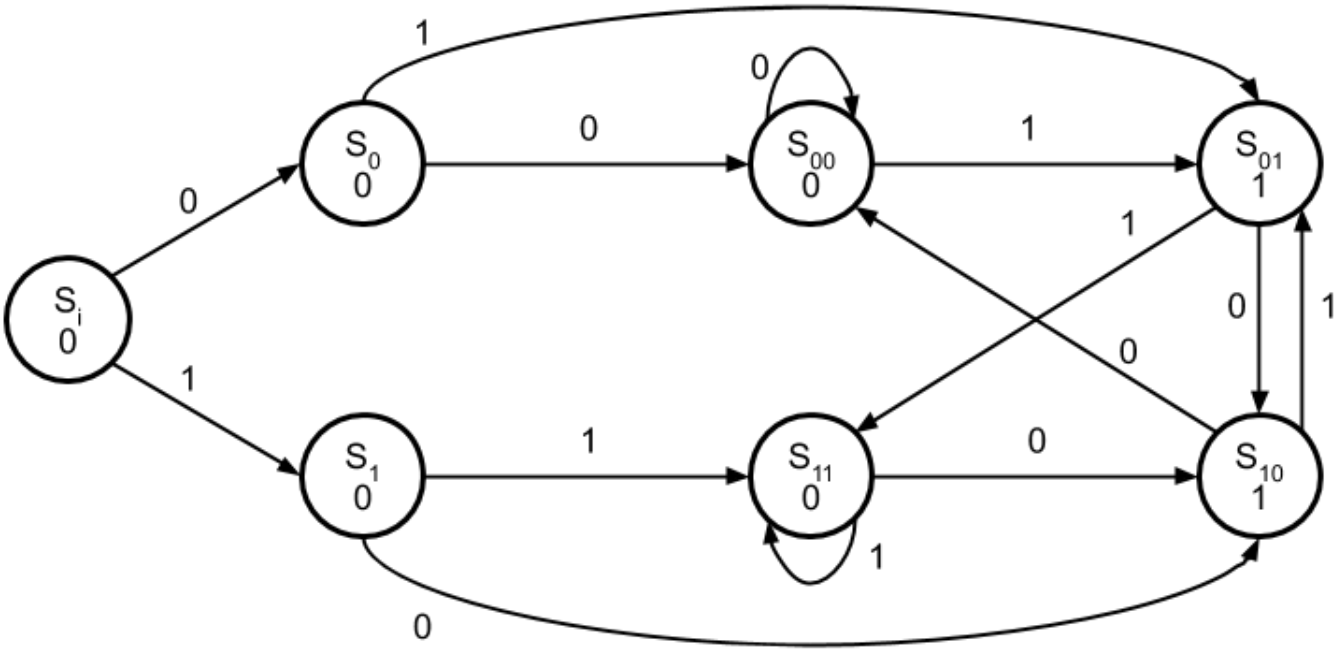
1. Recap of FSM design steps
    a. Turn a word description into a state diagram
    b. Turn the state diagram into a transition table
    c. Use transition table to make K-maps to simplify circuit
    d. Implement simplified circuit

2. Our word problem
    a. Want to create an edge detecting circuit
        i. Edge – position in a string of 0s and 1s where a 0 is adjacent to a 1
        ii. Essentially, XOR of adjacent bits
    b. Since this involves remembering the previous value, machine will have different states

3. State transition diagram
    a. Convert word description
        i. States based on patterns of inputs
        ii. Transitions between states based on individual input values
    b. For this problem
        i. S_i is our start state
        ii. Every other state is labeled S_{xy} where x is the previous bit, y is the current bit
            1. Each state will have a unique label
            2. This helps to minimizes the number of states
            3. Not perfect, though
                a. Will talk about method to further minimize number of states later
        iii. Values inside each state are the output of the circuit
            1. 0 indicates no edge
            2. 1 indicates an edge was detected



c. Mealy model differences
    i. Mealy model state diagrams look slightly different
        1. Outputs are on the edges/transitions between states, rather than the states themselves
        2. Usually, the number before the slash indicates the input
        3. Number after the slash indicates the output
        4. Will come back to this later

4. State table
    a. Table that lists all transitions from each *present state* to the *next state* for different values of inputs
        i. Output *z* is specified with respect to the present state
        ii. *x* is the next input
    b. Setting up the table
        i. Need a row for every single state
        ii. Need columns for each possible combination of inputs
        iii. Need a column for every output as well
    c. Filling in the table
        i. Use state transition diagram as reference
        ii. Given combination of present state and input x, what is the next state?
            1. Write name down
        iii. Given present state, what is the output of the present state?
            1. Bubble for present state has the output
            2. Write that value down

State Table			
Present State	**Next State**		**Output**
	***x* = 0**	***x* = 1**	***z***
I	0	1	0
0	00	01	0
1	10	11	0
00	00	01	0
01	10	11	1
10	00	01	1
11	10	11	0

d. Choice of flip-flops
    i. Here, will keep it simple and use DFFs
        1. Could use more complicated FFs, like J-K
        2. Using these typically means more logic in front of FFs
    ii. Occasionally, different types of flip flops can result in simpler circuitry
        1. You can assume DFFs unless specified otherwise
        2. Don't need to worry about getting it perfect in this class
            a. However, does need to be minimized with respect to DFFs

5. State transition table
    a. Will continue with the Moore model for this example
    b. Since we have 7 states, will need $\lceil \log_2 7 \rceil = 3$ DFFs to represent all possible states
        i. A, B, and C will represent the present state of the corresponding flip-flops
            1. This is the *output* of the FFs
        ii. A', B', and C' will represent the next state of the corresponding flip-flops
            1. This is the *input* into the FFs
        iii. x is the current input into the FSM (already stated above)
        iv. z is the current output out of the FSM (already stated above)
    c. Need to assign binary codes to each state
        i. When FFs contain this value, you know you're currently in that state
        ii. Initial state we start out in *i* must always be assigned to binary code of all 0s
            1. Flip flops are assumed to be 0 when we first start circuit
        iii. Have assigned binary codes below simply by going down the list of possible codes
            1. Will talk about better way to do this later
    d. Setting up the table
        i. Use state table as reference (or state transition diagram once you know what you're doing)
        ii. *Present State A*, *B*, and *C* columns along with *Input x* column form input
            1. Concatenation of A, B, C, and x form 4-bit binary number
            2. Looks very similar to a truth table for a 4-input Boolean function
                a. That's because we are making one for A', B', C', and z
    e. Filling in the table
        i. Use state table and binary codes to fill in columns for A', B', C'
            1. For given present state and binary code, what is the binary code of the next state?
            2. Example: eighth row, state 00 (011) with input of 1
                a. Next state is state 01 from state table
                b. Binary code of state 01 is 100, so place that inside columns for A', B', and C'
        ii. Output z is the same as in the state table

State Transition Table									
Present State	**Binary Code**	**Present State**			**Input**	**Next State**			**Output**
		A	***B***	***C***	***x***	***A'***	***B'***	***C'***	***z***
i	000	0	0	0	0	0	0	1	0
i	000	0	0	0	1	0	1	0	0
0	001	0	0	1	0	0	1	1	0
0	001	0	0	1	1	1	0	0	0
1	010	0	1	0	0	1	0	1	0
1	010	0	1	0	1	1	1	0	0
00	011	0	1	1	0	0	1	1	0
00	011	0	1	1	1	1	0	0	0
01	100	1	0	0	0	1	0	1	1
01	100	1	0	0	1	1	1	0	1
10	101	1	0	1	0	0	1	1	1
10	101	1	0	1	1	1	0	0	1
11	110	1	1	0	0	1	0	1	0
11	110	1	1	0	1	1	1	0	0
	111	1	1	1	0	d	d	d	d
	111	1	1	1	1	d	d	d	d

6. Derivation of minimized next-state and output expressions
    a. Use state transition table to create K-maps for FF input combinational circuits
        i. Inputs are the concatenation of the current state variables A, B, C, and the input x
        ii. Outputs of each K-map are the next state variables A', B', C' and the output z
            1. Need to create 4 different K-maps, one for each variable
        iii. Fill in K-maps using appropriate column from state transition table
            1. Since state 111 wasn't assigned, combinations of that with x are don't cares

A'

Cx \ AB	**00**	**01**	**11**	**10**
00	0	1	1	1
01	0	1	1	1
11	1	1	d	1
10	0	0	d	0

$$A' = B\bar{C} + A\bar{C} + Cx$$

B'

Cx \ AB	**00**	**01**	**11**	**10**
00	0	0	0	0
01	1	1	1	1
11	0	0	d	0
10	1	1	d	1

$$B' = \bar{C}x + C\bar{x}$$

C'

Cx \ AB	**00**	**01**	**11**	**10**
00	1	1	1	1
01	0	0	0	0
11	0	0	d	0
10	1	1	d	1

$$C' = \bar{x}$$

    b. Create K-map to determine the output combinational circuit
        i. Remember, Moore models don't use the input to determine the current output
            1. Therefore, K-map for z only uses A, B, and C
            2. Could place x in as well but result will be identical
        ii. Mealy model differences
            1. Mealy models use the current input to determine the current output
            2. Final K-map for output z would incorporate current flip-flop values as well as input x
            3. K-map below for Mealy model would have 4 inputs

z

C \ AB	**00**	**01**	**11**	**10**
0	0	0	0	1
1	0	0	d	1

$$z = A\bar{B}$$

7. Finishing steps
    a. After K-maps done, implement these circuits
    b. Mealy model differences
        i. In general, Mealy model usually involves less circuitry
            1. In this case, it would
            2. Only need two DFFs to remember previous bit
            3. XOR those values with input to give final output
        ii. Not always the case, though