



Problem 1 (River crossing with a middle island!). On one bank of a river are **four** missionaries and **four** cannibals. There is one boat available that can hold up to two people and that they would like to use to cross the river. The river contains a middle island that can be used to assist the movement of missionaries and cannibals across the river. (Here we are allowed to cross from bank-to-bank without stopping at the middle island. If this were not allowed, we could not transport two cannibals across if a missionary were on the middle island.)

If the cannibals ever outnumber the missionaries on either of the river's banks, the missionaries will get eaten. How can the boat be used to safely carry all the missionaries and cannibals across the river?



- a. Solve the riddle by drawing a diagram indicating the movement of missionaries and cannibals. Record the number of movements used. (M,C)

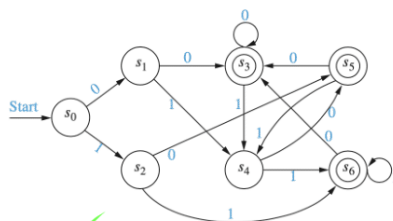
	Left Shore	Right Shore
	0	1
0	4,4	0,0
1	4,2	0,2
2	4,3	0,1
3	4,0	0,4
4	4,1	0,3
5	1,1	3,3
6	2,2	2,2
7	0,2	4,2
8	0,3	4,1
9	0,0	4,4

~ cannibals held 2 in the Boat

- b. (Bonus) Solve the original riddle with the assumption that shore-to-shore travel is not allowed (i.e., you must stop at the middle island). Does this change the number of movements used?

- c. (Bonus) Solve the riddle with $N = 5$ missionaries and $N = 5$ cannibals (without and with shore-to-shore movement)
- d. (Bonus) Attempts at implementing a search algorithm (in a language of your choice) to construct a path from initial state to end state in the Missionaries and Cannibals (+ Middle Island) Riddle?

Problem 2. (Why do you hate 01?) Consider the following deterministic finite-state automaton (DFA).



$S_3 = 00$
 $S_5 = 10$
 $S_6 = 11$

a. Determine which of the following are accepted by the DFA. What state do they end at?

$w_1 = 101010$ (42!)

S_5

$w_2 = 000111$

S_6

$w_3 = 011100$

S_3

$w_4 = 10100111001$ (1337!)

S_4

b. Identify all bit-strings of length 3 that are accepted by the DFA.

000

~~001~~

010

011

100

~~101~~

110

111

c. Identify all bit-strings of length 4 that are accepted by the DFA.

0000

~~0001~~

0010

0011

0100

~~0101~~

0110

0111

1000

~~1001~~

1010

1011

1100

~~1101~~

1110

1111

d. Identify all bit-strings of length 5 that are accepted by the DFA.

00000

~~00001~~

00010

00011

00100

~~00101~~

00110

00111

01000

01001

01010

01011

01100

01101

01110

01111

10000

10001

10010

10011

10100

10101

10110

10111

11000

~~11001~~

11010

11011

11100

~~11101~~

11110

11111

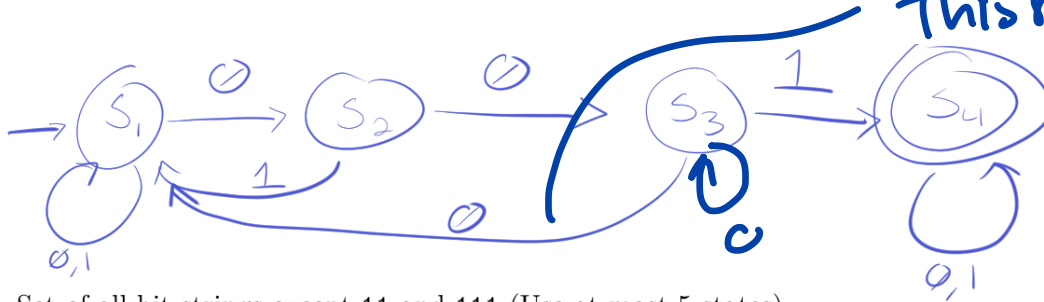
e. (Bonus) Describe (in words) the set of bit-strings recognized by this DFA. (Hopefully you see a pattern in the mess that is this machine.)

this machine only rejects strings ending in 01,
 & strings shorter than 2 bits

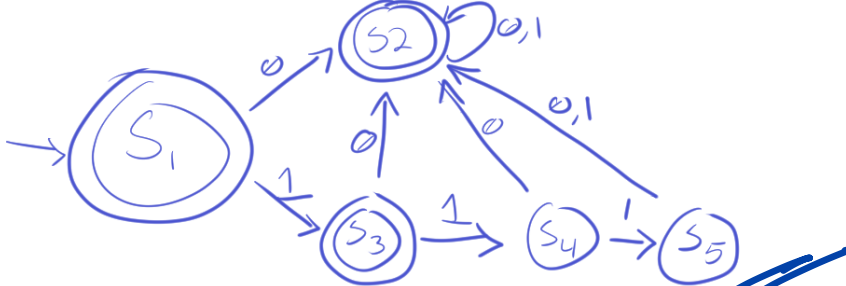
f. Blank space – add some sweet pictures – yay! Fun fact, binary numbers that end with 01 are congruent to 1 modulo 4.

Problem 3. (Yay designing DFAs - Enjoy!) For each of the following, construct a deterministic finite-state automaton (DFA) that recognizes the given set.

- a. Set of all bit-strings that contain the substring 001. (Use at most 4 states)



- b. Set of all bit-strings except 11 and 111 (Use at most 5 states)

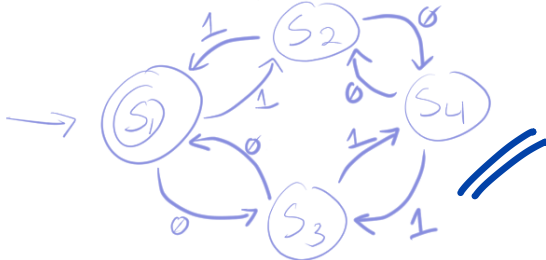


- c. Set of all bit-strings where every 1 is followed by at least two 0s. (e.g., 0, 100, 01000, 1001000, ...)



~ This accepts 11111100
No 0s after these

- d. Set of all bit-strings that contain an even number of 0s and an even number of 1s. (Use 4 states)

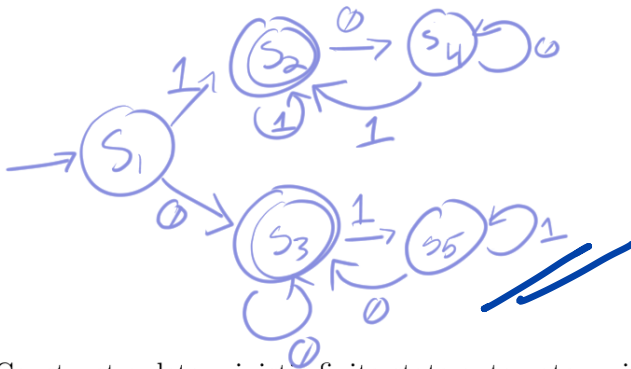


is 0 even
 $0 \% 2 = 0$
so yes?
Yup!

- e. (Bonus) Set of bit-strings that contain an equal number of occurrences of the substrings 01 and 10. Note 101 is accepted because 101 contains a single 10 and a single 01, but 1010 is not because it contains two 10s and one 01.

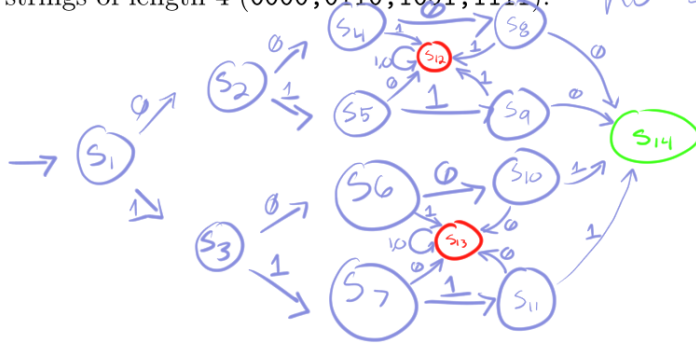
Problem 4. (Palindromes and DFA are bad news :-/) A string w is palindromic if it is the same read forwards and backwards - e.g., 10101, 111, racecar, neveroddoreven (Never odd or even).

- a. Construct a deterministic finite-state automaton with fixed input length $\ell = 3$ (all words/strings are of length 3) that recognizes the palindromic bit-strings of length 3 (000, 010, 101, 111). (Solve with 6 states or fewer.)



✓ 000 = s_4
 ✓ 001 = s_6
 ✓ 110 = s_4
 ✓ 011 = s_6

- b. Construct a deterministic finite-state automaton with fixed input length $\ell = 4$ that recognizes the palindromic bit-strings of length 4 (0000, 0110, 1001, 1111).



● = Fail dump
 ● = success dump
 ●

- c. (Bonus) Construct a deterministic finite-state automaton with fixed input length $\ell = 5$ that recognizes the palindromic bit-strings of length 5 (00000, 00100, 01010, 01110, 10001, 10101, 11011, 11111).

- d. (Bonus) What issues emerge when you attempt to construct a DFA that recognizes all bit string palindromes? (This is one of the examples the prompts us to look at additional models of computation)

we don't have any means of storing data or referring to elements/variables that have already been addressed in the string. this makes it difficult to tackle variable-length strings, since we have to plan for all possible cases.

also complicating things is a lack of any comparator method → which reinforces the requirement that we anticipate all cases in advance, and prepare all

Problem: the longer the string, the further back we must recall, without any means of storing, indexing, or referring to data, we have to prepare a unique path for each new palindrome. which is real bad :)