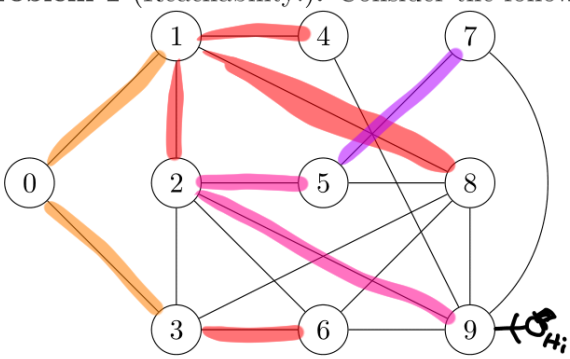
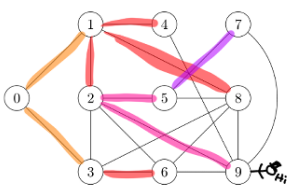
CSC 404 - Exam 2 - Name: Chris Glanzer

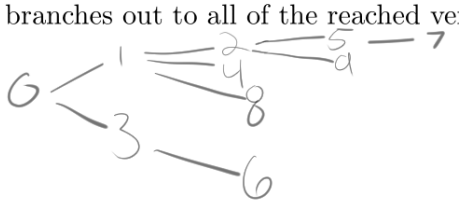
Solve 9 of the following 10 problems :-). Or you can do all 10 and I will throw a few bonus points your way.

Problem 1 (Reachability!)**.** Consider the following graph:



a. By running a breadth-first search, determine all of the vertices/nodes that can be reached starting at 0. Draw the resulting tree that stems from 0 and branches out to all of the reached vertices/nodes.

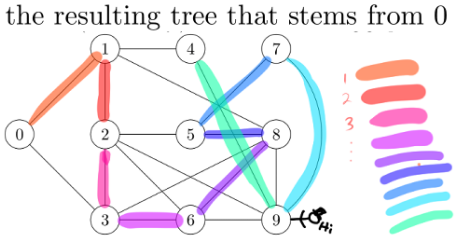




b. Which nodes can be reached within 3 edges/steps from 0?

All but 7

c. By running a depth-first search, determine all of the vertices/nodes that can be reached starting at 0. Draw the resulting tree that stems from 0 and branches out to all of the reached vertices/nodes.



0-1-2-6-8-5-7-9-4

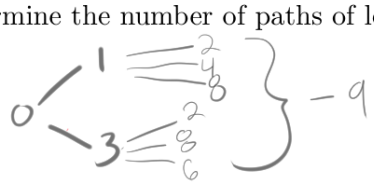
d. Construct the Adjacency Matrix, A.

blank = 0

$$A = \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \\ 9 \end{array} \begin{array}{c} \begin{array}{cccccccccc} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \end{array} \\ \begin{pmatrix} - & 1 & - & 1 & - & - & - & - & - & - \\ 1 & - & 1 & - & 1 & - & - & - & 1 & - \\ - & - & - & 1 & - & 1 & 1 & - & - & 1 \\ 1 & - & 1 & - & - & - & 1 & - & 1 & - \\ - & 1 & - & - & - & - & - & - & - & 1 \\ - & - & 1 & - & - & - & - & 1 & 1 & - \\ - & - & 1 & 1 & - & - & - & - & 1 & 1 \\ - & - & - & - & - & 1 & - & - & - & 1 \\ - & - & - & 1 & - & 1 & 1 & - & - & 1 \\ - & - & 1 & - & - & - & 1 & - & 1 & - \end{pmatrix} \end{array} \begin{array}{l} 1 \\ 1 \\ 1 \\ 1 \\ 0 \end{array}$$

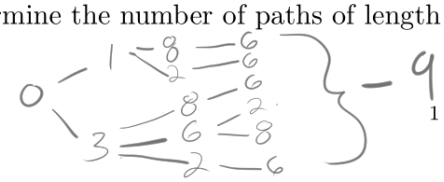
e. By using A^3, determine the number of paths of length 3 from 0 to 9. Then, list these paths.

6 paths

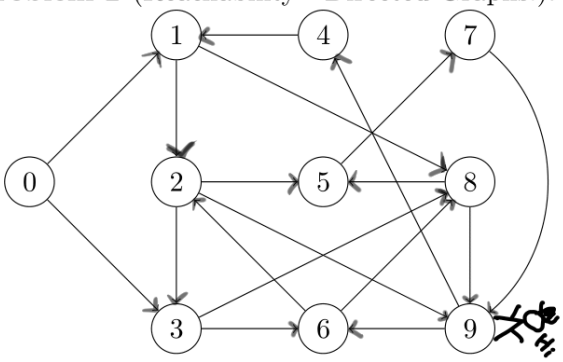


f. By using A^4, determine the number of paths of length 4 from 0 to 9. Then, list these paths :-).

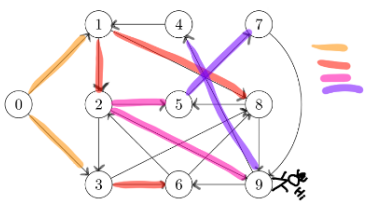
6 paths

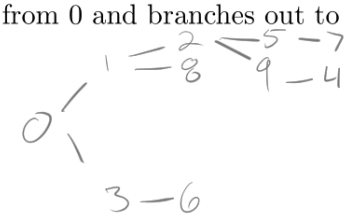


Problem 2 (Reachability - Directed Graphs!)**.** Consider the following directed graph:



a. By running a breadth-first search, determine all of the vertices/nodes that can be reached starting at 0. Draw the resulting tree that stems from 0 and branches out to all of the reached vertices/nodes.

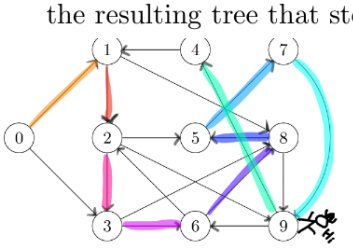




b. Which nodes can be reached within 3 edges/steps from 0?

all but (4) & (7)

c. By running a depth-first search, determine all of the vertices/nodes that can be reached starting at 0. Draw the resulting tree that stems from 0 and branches out to all of the reached vertices/nodes.



0-1-2-3-6-8-5-7-9

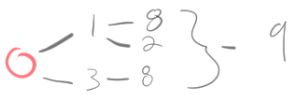
d. Similar to to non-directed graphs, we can construct an adjacency matrix A that records (directed) edges between two vertices. Since $0 \to 1$ we have a 1 in the $(0,1)$ spot. Since we do not have an arc from 1 to 0 we have a 0 in $(1,0)$ splot. Construct the matrix A.

blank = 0

$$A = \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \\ 9 \end{array} \begin{array}{c} \begin{array}{cccccccccc} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \end{array} \\ \begin{pmatrix} -- & 1 & -- & 1 & -- & -- & -- & -- & -- & -- \\ 0 & -- & 1 & -- & -- & -- & -- & -- & 1 & -- \\ -- & -- & -- & 1 & -- & 1 & -- & -- & -- & 1 \\ -- & -- & -- & -- & -- & -- & 1 & -- & 1 & -- \\ -- & 1 & -- & -- & -- & -- & -- & -- & -- & -- \\ -- & -- & -- & -- & -- & -- & -- & 1 & -- & -- \\ -- & -- & 1 & -- & -- & -- & -- & -- & 1 & -- \\ -- & -- & -- & -- & -- & -- & -- & -- & -- & 1 \\ -- & -- & -- & -- & -- & 1 & -- & -- & -- & 1 \\ -- & -- & -- & -- & 1 & -- & 1 & -- & -- & -- \end{pmatrix} \end{array}$$

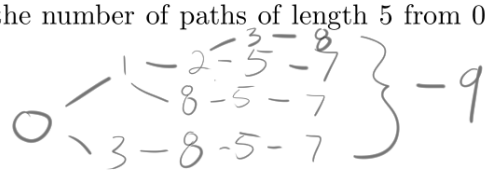
e. By using A^3, determine the number of paths of length 3 from 0 to 9. Then, list these paths.

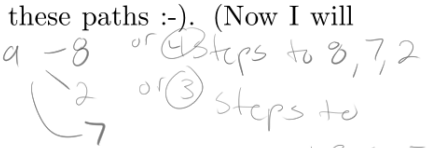
3 paths



f. By using A^5, determine the number of paths of length 5 from 0 to 9. Then, list these paths :-). (Now I will make you look at A^5)

4 paths





g. By using A^6, determine the number of paths of length 6 from 0 to 9. Then, list **two** such paths :-).

11 paths

0-1-8-9-6-8-9

0-1-2-9-6-8-9

5 steps to 8, 7, 2
4 steps to 1, 3, 6, 5
3 steps to 4

Problem 3 (More Swapping Pennies!)**.** In Homework 4 we looked into a simple game called matching pennies. Here two players Eve (Even) and Bob (Odd) start with two pennies and depending on the outcome of a secret revel pennies are passed between them – first person to 4 pennies wins.

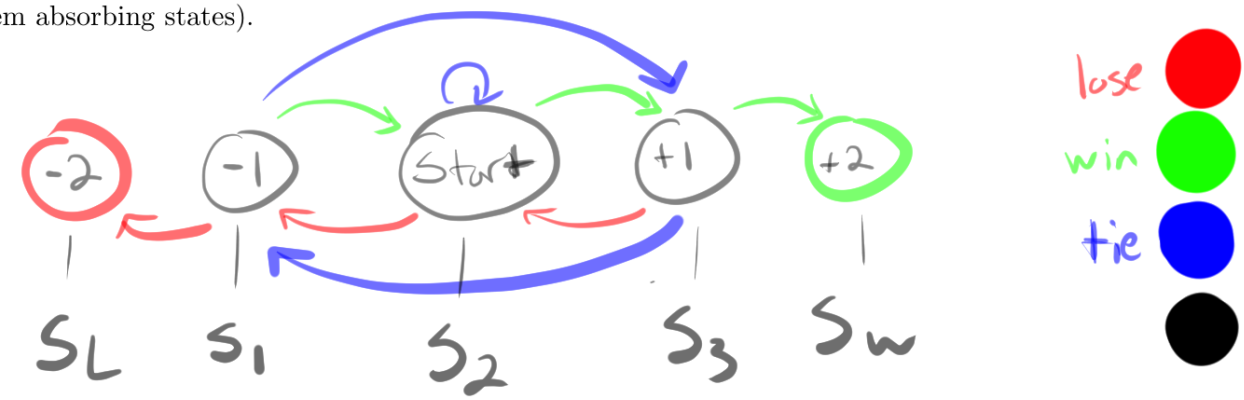
Let's change things up – here the secret revel will be a game of Rock-Paper-Scissors. If who ever wins receives an opponents penny. But, here is the twist – if there is a tie the two players swap penny piles (e.g., if Eve has 3 pennies and Bob has 1 penny following a tie Eve will have 1 penny and Bob will have 3 pennies).

Consider the following example, where Eve and Bob both start with two pennies. (Here the states `Si` denote how many pennies Eve has with `S0 = SL` and `S4 = SW`.

Round	Eve	Bob	Eve (#)	Bob (#)	State	Description
0	-	-	2	2	S2	Initially both have two pennies!
1	R	S	3	1	S3	Rock Beats Scissors - Eve wins a Penny!
2	P	P	1	3	S1	Paper Tie - Swap Penny Piles
4	S	P	2	2	S2	Scissors Beats Paper - Eve wins a Penny
5	S	R	1	3	S1	Rock Beats Scissors - Eve loses a penny
6	S	S	3	1	S3	Scissors Tie - Swap Penny Piles
7	S	P	4	0	S4 = SW	Scissors Beats Paper - Eve wins/wins overall

a. If both parties randomly throw R, P, or S, then each have a $1/3 = 0.3333$ chance of winning, losing, or tie.

i. Construct the state diagram for this game. Add loops on 1 at the two 'end game states' - `SW` and `SL` (i.e., make them absorbing states).



ii. Construct the state transition matrix, P, for this game. Label the rows SL, S1, S2, S3, SW. (You should see a 1 in the top left and bottom right)

$$P = \begin{array}{c} \\ SL \\ S1 \\ S2 \\ S3 \\ SW \end{array} \begin{array}{c} \begin{array}{ccccc} SL & S1 & S2 & S3 & SW \end{array} \\ \begin{pmatrix} 1 & -- & -- & -- & -- \\ 0.3 & -- & 0.3 & 0.3 & -- \\ -- & 0.3 & 0.3 & 0.3 & -- \\ -- & 0.3 & 0.3 & -- & 0.3 \\ -- & -- & -- & -- & 1 \end{pmatrix} \end{array}$$

iii. Compute P^{100} (or your favorite power of P). You should see a matrix with (essentially) zeros in the middle and all of the data on the left and right columns. From this, record the probability of Eve winning/losing. (Remember we start with 2 pennies, but you can also see the probabilities of starting with 1 or 3 pennies)

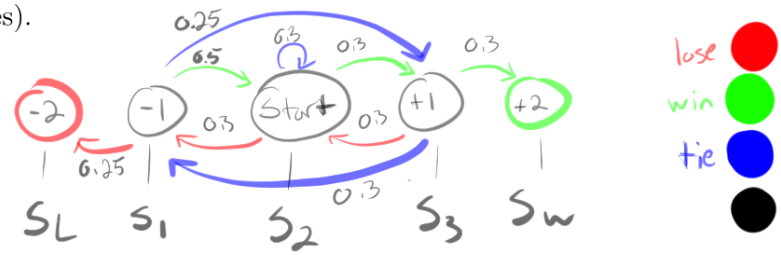
```
P =
[1.0, 0.0, 0.0, 0.0, 0.0]
[0.33, 0.0, 0.33, 0.33, 0.0]
[0.0, 0.33, 0.33, 0.33, 0.0]
[0.0, 0.33, 0.33, 0.0, 0.33]
[0.0, 0.0, 0.0, 0.0, 1.0]
n =100| P^n =
0: [0:1.0, 1:0.0, 2:0.0, 3:0.0, 4:0.0, ]
1: [0:0.6, 1:0.0, 2:0.0, 3:0.0, 4:0.35, ]
2: [0:0.47, 1:0.0, 2:0.0, 3:0.0, 4:0.47, ]
3: [0:0.35, 1:0.0, 2:0.0, 3:0.0, 4:0.6, ]
4: [0:0.0, 1:0.0, 2:0.0, 3:0.0, 4:1.0, ]
```

I printed rounded data... [illegible] probability seems like after 100 rounds, the missing 0.01% (1-3(0.33)) grew to a missing 6%

6.47 Chance to win/lose starting @ 2 pennies

b. Eve has been tipped off that if Bob has three pennies, i.e., we are at state `S1`, then he is more likely to throw Rock than Scissors or Paper (50% Rock to 25% Scissors and 25% Paper). With that, whenever Bob is up to three pennies Eve plans to always throw Paper.

i. Construct the state diagram for this game. Add loops on 1 at the two 'end game states' - `SW` and `SL` (i.e., make them absorbing states).



ii. Construct the state transition matrix, P, for this game.

$$P = \begin{array}{c} \\ SL \\ S1 \\ S2 \\ S3 \\ SW \end{array} \begin{array}{c} \begin{array}{ccccc} SL & S1 & S2 & S3 & SW \end{array} \\ \begin{pmatrix} 1 & -- & -- & -- & -- \\ 0.25 & -- & 0.5 & 0.25 & -- \\ -- & 0.3 & 0.3 & 0.3 & -- \\ -- & 0.3 & 0.3 & -- & 0.3 \\ -- & -- & -- & -- & 1 \end{pmatrix} \end{array}$$

iii. Compute P^{100} (or your favorite power of P). You should see a matrix with (essentially) zeros in the middle and all of the data on the left and right columns. From this, record the probability of Eve winning/losing.

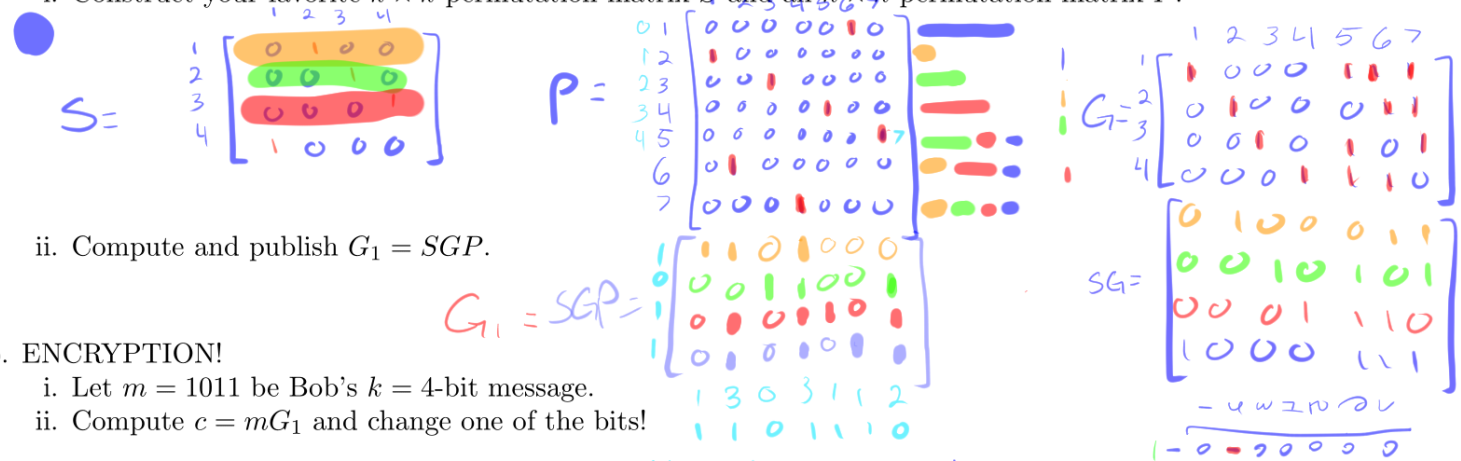
@ 100 W: 53% L: 42%

c. (Bonus) If you have more ideas for variations to this game let me know (the above example was from a student's submission in HW4)

Problem 4 (McEliece Cryptosystem – one more time, because it is really cool)**.** Alice wants to use the McEliece Public Key Cryptosystem with a $[n, k]$ Hamming Error Correcting Code, C. To keep things 'small', let's use $[n, k] = [7, 4]$. i.e., $r = 3$.

a. KEY GEN!

i. Construct your favorite $k \times k$ permutation matrix S and an $n \times n$ permutation matrix P.



ii. Compute and publish $G_1 = SGP$.

$G_1 = SGP =$

b. ENCRYPTION!

i. Let $m = 1011$ be Bob's $k = 4$-bit message.

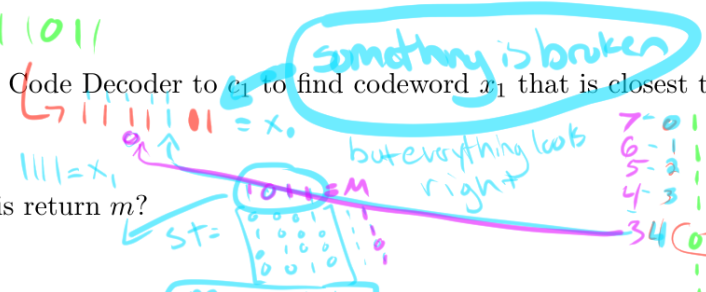
ii. Compute $c = mG_1$ and change one of the bits!

c= 110 1110 → 11 1110

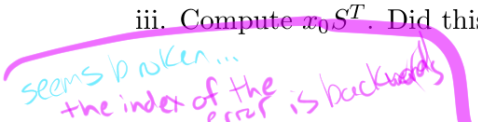
c. DECRYPTION!

i. Compute $c_1 = cP^T$

ii. Apply Error Correcting Code Decoder to c_1 to find codeword x_1 that is closest to c_1. Then, let x_0 be the first k bits of x_1.



iii. Compute $x_0 S^T$. Did this return m?



Problem 5 (A 'bit' of bad hashing)**.** In our chat about Hashing Algorithm we constructed a 'Bad Hash' by XORing all of the entries together (all of the collisions). In our new version of 'Bad Hash' we initialize the digest to be 0, XOR the next character, then XOR that result with its bit reversal. (Here, $rev(1101) = 1011$)

```
badHashV2(str2hash)
    d = 0
    for x in str2hash_unicode:
      d = d xor x
      d = d xor bit_reversal(d)
    return(d)
```

For example, let's play with the string 'Hash'. Notice that

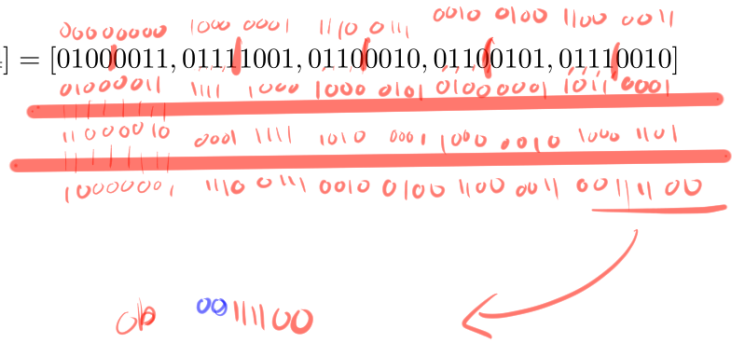
$$Hash \sim [H, a, s, h] \sim [72, 97, 115, 104] \sim [01001000, 01100001, 01110011, 01101000]$$

Round 1: $d = 0$ and $x = 72 = 01001000$
$d = d \oplus x = 00000000 \oplus 01001000 = 01001000$
$d = d \oplus rev(d) = 01001000 \oplus 00010010 = 01011010 = 90$
Round 2: $d = 90 = 01011010$ and $x = 97 = 01100001$
$d = d \oplus x = 01011010 \oplus 01100001 = 00111011$
$d = d \oplus rev(d) = 00111011 \oplus 11011100 = 11100111 = 231$
Round 3: $d = 231 = 11100111$ and $x = 115 = 01110011$
$d = d \oplus x = 11100111 \oplus 01110011 = 10010100$
$d = d \oplus rev(d) = 10010100 \oplus 00101001 = 10111101 = 189$
Round 4: $d = 189 = 10111101$ and $x = 104 = 01101000$
$d = d \oplus x = 10111101 \oplus 01101000 = 11010101$
$d = d \oplus rev(d) = 11010101 \oplus 10101011 = 01111110 = 126$

Thus, `badHashV2('Hash') = 126 = 0b01111110`.

a. Run the badHashV2 on the string 'Cyber'.

$$[C, y, b, e, r] = [67, 121, 98, 101, 114] = [01000011, 01111001, 01100010, 01100101, 01110010]$$



b. On D2L I have a link to an implementation of this 'Bad Hash'. Verify your work in part a. and determine another string with the same hash. (i.e., find a collision!).

```
13 OCCe\C`vXLd]yYr 60
```

c. Let's find some more collisions! Based on the birthday paradox, for a collection of about 16 strings we have a pretty decent chance of finding a collision. (Randomly) generate a list of 16 strings and compute the corresponding 'Bad Hash' and record any collision(s) that exist. (If not collisions rerun until you find a collision).

```
15 c[]OUNNCuQc 165
```

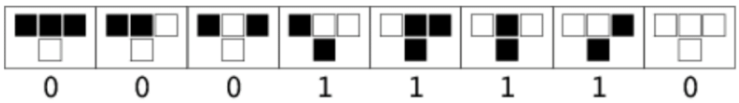
```
7 VyXfH_c^gKh`{\Mp{ 165
```

d. (Bonus) You know you want to build this into a Turing Machine.

```
========== Bad Hash - Example 4 ==========
String: c[]OUNNCuQc
bad_hash(string) = 165
bad_hash(string) = 0b10100101
bad_hash(string) = 0xa5
```

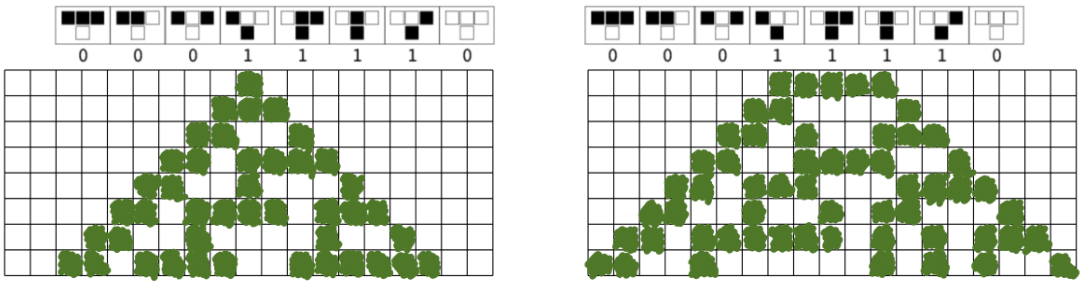
Problem 6 (1D Cellular Automata!)**.** In our last section we played around with Conway's Game of Life – a 2D cellular automata. Here we constructed new generations and stored them in a new 2D array and then 'animated' them by displaying individual frames on top of each other (think old school flip-books). We can back things up and look at 1D cellular automata. Here, each generation will be a 1D array and we 'animate' these by printing future generations below the current generation (though you can also do traditional flip-book animations).

As with the Game of Life (2D cellular automata), in 1D cellular automata future states will be determined by the current state of a cell (dead/alive) and the states of its two neighbors. Since there are $2 \cdot 2 \cdot 2 = 2^3 = 8$ possible binary states for the three cells neighboring a given cell, there are a total of $2^8 = 256$ one-dimensional cellular automata. These can be indexed with an 8-bit binary number, for example, the following table gives the evolution of rule 30 (30 = 00011110).

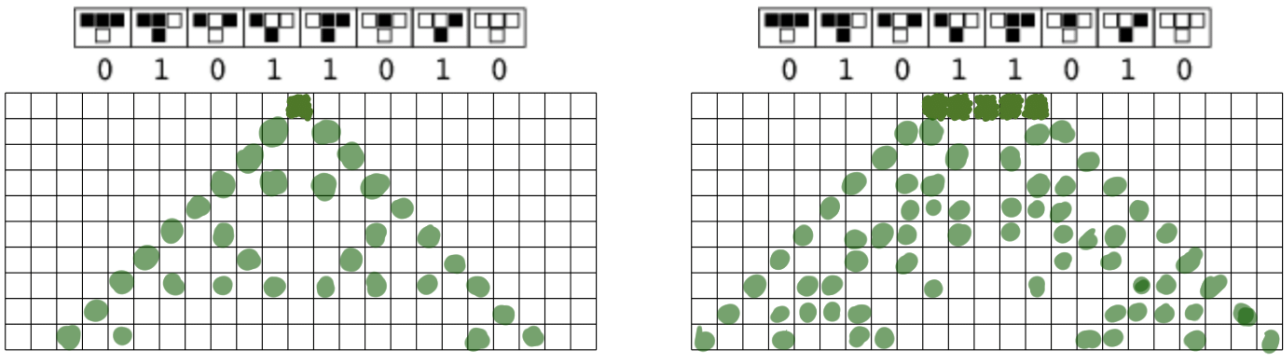


The first says if a cell is currently alive and its two neighbors are alive it will die. The second says if a cell is alive and the neighbor to the left is alive and the neighbor to the right is dead it will die....

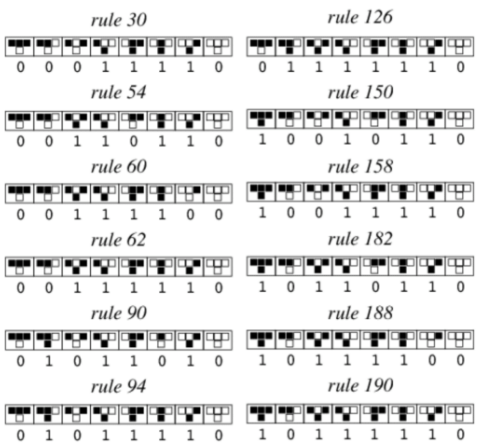
With this, we can construct future generations (rows) based off of different starting generations (first row).



a. Construct future generations using rule 90 (90 = 01011010)



b. (Bonus) Implement the 1D cellular automata in a language of your choice. Some other interesting 'Rules' to try it on are the following. Note - for some of these you will see a very structure emerge (e.g., symmetries). Whereas rules like Rule 30 are chaotic. In fact, Rule 30 is used in the pseudo random number generator for lager integers in the Wolfram Language. For more information see https://mathworld.wolfram.com/ElementaryCellularAutomaton.html

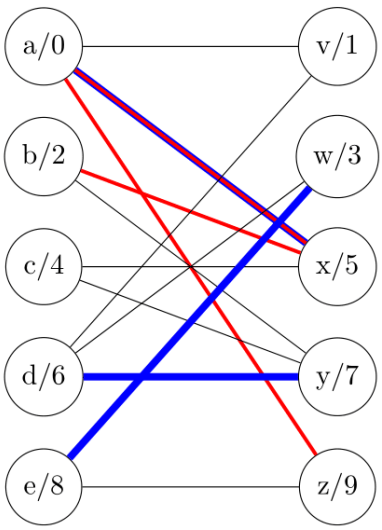


Problem 7 (Matching – Hopcroft and Karp)**.** Given a matching M in a bipartite graph G, we say that a simple path P in G is an 'augmenting path' with respect to M if it

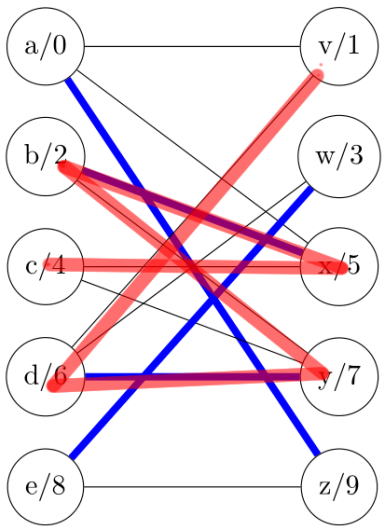
i. Starts at an unmatched vertex on one side and ends at an unmatched vertex on the other side.
ii. Its edges alternate between a non-matched edges and matched edges.

By identifying augmenting paths we can apply a symmetric difference/XOR to increase the size of the matching!

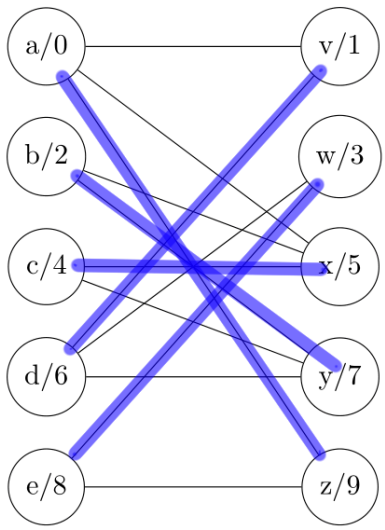
a. Consider the matching $M = \{05, 67, 83\}$ with size $|M| = 3$ (given in blue below). An augmenting path is $P = \{25, 50, 09\}$ (given in red). Thus, we can compute $M \oplus P$ to improve or original matching! Here $M' = M \oplus P = \{09, 25, 67, 83\}$ (All but the double counted edge 05). Using M' identify another augmenting path P' and use this to construct a maximal matching $M'' = M' \oplus P'$.



Matching: $M = \{05, 67, 83\}$
Path: $P = \{25, 50, 09\}$

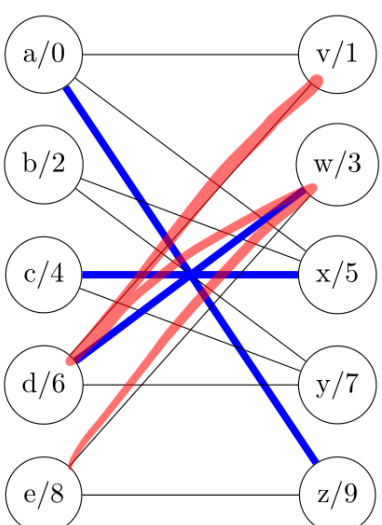


Matching $M' = M \oplus P$:
$M' = \{09, 25, 67, 83\}$
$P' = \{45, 52, 27, 76, 61\}$

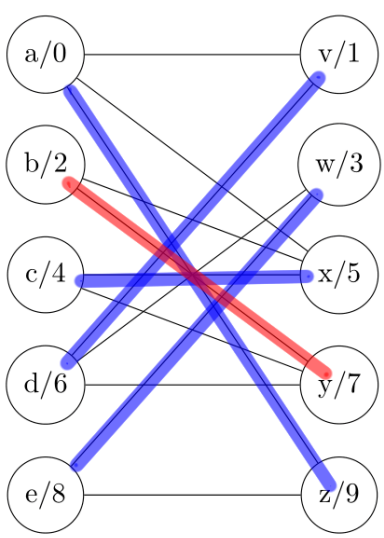


Matching $M'' = M' \oplus P'$:
$M'' = \{09, 27, 45, 61, 83\}$

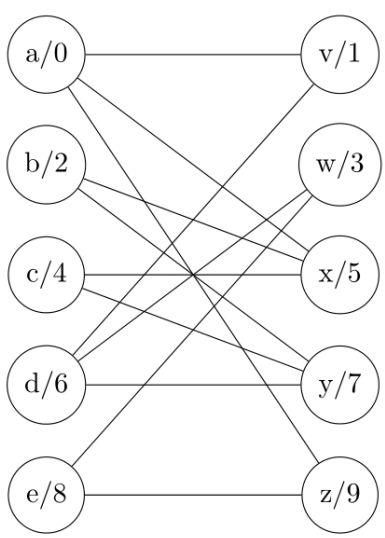
b. Consider the matching $M = \{09, 45, 63\}$ with size $|M| = 3$. Identify an augmenting path P and construct $M' = M \oplus P$. Then identify an augmenting path P' to M' and construct $M'' = M' \oplus P'$.



Matching: $M = \{09, 45, 63\}$
Path: $P = 83, 36, 61$



Matching $M' = M \oplus P$:
$M' = \{09, 45, 61, 83\}$
$P' = \{2\ 7\}$



Matching $M'' = M' \oplus P'$:
$M'' = \{09, 45, 61, 83, 27\}$