CS Problem Set #4

This assignment is due on Sunday, July 29th by 11:59 pm. Questions #1 - 6 are required. Challenge problems are optional for CS2, though encouraged. Submit your answers to the CS Problem Set #2 Gradescope assignment:

- Encapsulate all the code you used for this assignment into stand-alone functions, then submit a single python file
- A single .pdf document should contain all your written responses. Hint: Make a copy of this Doc on your own Drive.

Required Problems

1. Marina is attempting to find a solution to a cryptarithmetic puzzle by hand, with brute force.

   Because each letter in a cryptarithmetic puzzle represents a unique digit (0 - 9), Marina's approach is to try all possible combinations of digits for each letter (excluding leading 0's). If Marina finds the first solution after considering 10% of all possible combinations, approximately how many hours will she spend trying to solve the puzzle, assuming that it takes Marina 30 seconds to try each new combination? Justify your answer.

```
          CS
+         IS
+        FUN
--------
=       TRUE
```

2. In the context of building Word Ladders, Donald Knuth uses *"aloof"* to describe words that are not connected to any other words with a hamming distance of 1. Note that "aloof" is itself such a word.[3] How many words in **valid_ladders_word_list** are *"aloof"*? Describe the algorithm you used to arrive at your answer.

3. A different way of representing a graph is with a matrix:

Matrix: 36 elements	**Adjacency List:** 18 elements

`[[0, 1, 1, 1, 1, 0],`
` [1, 0, 0, 1, 1, 0],`
` [1, 0, 0, 0, 0, 1],`
` [1, 1, 0, 0, 0, 1],`
` [1, 1, 0, 0, 0, 1],`
` [0, 0, 1, 1, 1, 0]]`
	`{'A':['B','C','D','E'],`
	` 'B':['A','D','E'],`
	` 'C':['A','F'],`
	` 'D':['A','B','F'],`
	` 'E':['A','B','F'],`
	` 'F':['C','D','E']}`

Considering only the 4-letter words in **valid_ladders_word_list**:

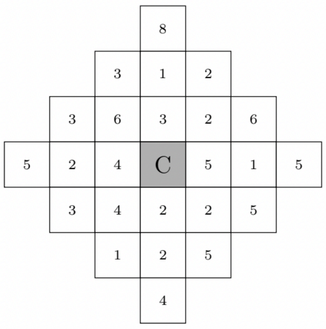
a) How many elements would exist in the **matrix** representation of the graph?

b) How many elements would exist in the sparse, **adjacency list** version of the graph?

c) How many elements would be in the **pruned matrix**, which has all *"aloof"* words removed?

4. The number of coconuts that have fallen from a coconut tree can be documented on this map:

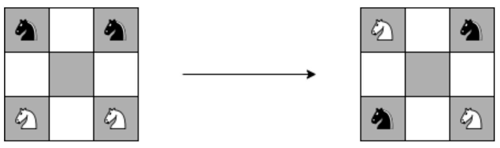
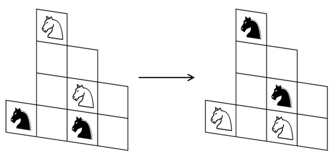
   Starting at the coconut tree, and only assuming a horizontal or vertical movement away from the tree, what is the maximum number of coconuts you could collect after three movements from the tree? Describe how you approached solving this problem.



5. If you wanted to solve Project Euler #249, you'd start by exploring subsets of the set of prime numbers less than 5000. If **S** = {2, 3, 5, ..., 4999} is the set of prime numbers less than 5000, how many digits are in the total number of subsets of **S**? How do you know?

   Note: You do not have to solve Project Euler #249 to answer this question. Project Euler #249 is an optional challenge problem.

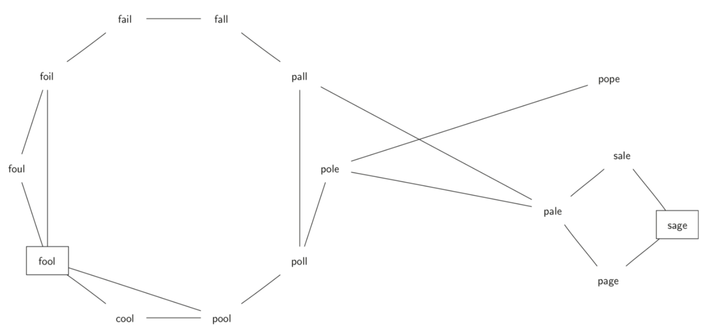
6. Solve ***one*** of these alternate versions of Guarini's Puzzle. Represent your solution as a linear sequence of steps. If your reasoning can be more easily explained with a photo/diagram you may upload a photo of your justification to Gradescope..

Alternating Guarini Knights	**Jagged Guarini Knights**
	

Challenge Problems

Challenge problems are optional for CS2.

1. It takes roughly $\mathbf{n^2}$ comparisons to generate our word ladder adjacency lists, which results in slow run times for large word lists (hence the slow Gradescope grading times). One way to potentially reduce the amount of time required to generate our adjacency list is to use a letter mask as the key structure lookup in our dictionary. This would essentially group words together into "fully-connected neighborhoods":



```
{
    ...
   '*ool': {'fool', 'cool', 'pool'},
   'fo*l': {'fool', 'foil', 'foul'},
   '*ope': {'pope', 'rope', 'nope', 'hope', 'lope',
             'mope','cope'},
   'p*pe': {'pope', 'pipe', 'pape'},
   'po*e': {'pope','pole','pore','pose','poke'},
   'pop*': {'pope', 'pops'},
    ...
 }
```

Thus, a single four-letter word would fit into 4 letter mask neighborhoods: *___, _*__, __*_, ___* . This should take far fewer comparisons (n words * m letters per word), theoretically resulting in faster generation of an adjacency list.

a) Compare the original adjacency list generated from **valid_ladders_word_list** with the word mask adjacency list generated by the same word list.
   - Which version of the adjacency list takes longer to generate for 3, 4, 5, 6, 7 letter word families?
   - Which version of the adjacency list has more keys? Which version of the adjacency list has more values? Does your answer change for the pruned version of the adjacency list (ie. "aloof" words removed)?

b) Write a new version of **LadderGraph** which generates and uses the masked letter mask adjacency list.
   - What needs to change in the bfs pathfinding algorithm in order to accommodate this new data model?
   - What other parts of your **LadderGraph** class would need to change?

2. Solve Project Euler #259:

   *Let **S** = {2, 3, 5, ..., 4999} be the set of prime numbers less than 5000.*
   *Find the number of subsets of **S**, the sum of whose elements is a prime number.*