Problem Set 5

All parts are due on March 22, 2019 at 6PM. Please write your solutions in the LaTeX and Python templates provided. Aim for concise solutions; convoluted and obtuse descriptions might receive low marks, even when they are correct. Solutions should be submitted on the course website, and any code should be submitted for automated checking on `alg.mit.edu`.

Problem 5-1. [16 points] **Graph Mechanics**

(a) [4 points] Draw the **directed** graph associated with each of the following graph representations.

```
G1 = {"StringMander": ["MITsaur"],
           "KoiQueen": ["StringMander", "GeoBro"],
             "GeoBro": ["StringMander", "PikaBoo"],
            "MITsaur": ["KoiQueen", "GeoBro"],
            "PikaBoo": ["KoiQueen"]}

G2 = [[1,6], [5,6], [4], None, [2], [0], []]
```

(b) [4 points] Let the ***k*-prime-difference graph** be the undirected graph on vertices $\{1, \ldots, k\}$ having an undirected edge between distinct vertices u and v if and only if $|u - v|$ is a prime number. For example, the 5-prime-difference graph contains edges $\{5, 3\}$ and $\{2, 5\}$ (and others), but not edges $\{4, 5\}$ or $\{5, 1\}$. Draw a picture of the 8-prime-difference graph, and write its graph representation as a direct access array of adjacency lists, as in `G2` above. Hint: your graph should have 15 edges.

(c) [8 points] Run breadth-first search and depth-first search on the 8-prime-difference graph, starting from node 7. While performing each search, visit the neighbors of each vertex in increasing order. For each search, draw the parent tree constructed by the search, and list the order in which nodes are first visited.

Problem 5-2. [10 points] **Graph Diameter**

In any undirected graph $G = (V, E)$, the **eccentricity** $e(u)$ of a vertex $u \in V$ is the shortest distance to its farthest vertex v, i.e., $e(u) = \max\{\delta(u, v) \mid v \in V\}$. The **diameter** $D(G)$ of an undirected graph $G = (V, E)$ is the largest eccentricity of any vertex, i.e., $D(G) = \max\{e(u) \mid u \in V\}$. The diameter of a graph is then the largest distance between two nodes.

(a) [5 points] Given connected undirected graph G, describe an $O(|V||E|)$-time algorithm to determine the diameter of G.

(b) [5 points] Given connected undirected graph G, describe an $O(|E|)$-time algorithm to determine an upper bound D^* on the diameter of G, such that $D(G) \leq D^* \leq 2D(G)$.

Problem 5-3. [8 points] **Social Optimization**

LinkedOUT is a new social network. Every pair of LinkedOUT users is either **acquainted** or not acquainted with each other. Two users u_1 and u_2 are **approachable** if there exists a sequence of users starting with u_1 and ending with u_2 where each adjacent pair of users in the sequence are acquainted. New user Weff Jeiner has just signed up for LinkedOUT and is not acquainted with anyone. Given a list of all pairs of users that are acquainted on LinkedOUT, describe an efficient algorithm to determine the minimum number of users with whom Weff needs to become acquainted in order to be approachable by every user on LinkedOUT.

Problem 5-4. [21 points] **Beaver Break**

The snow and ice have **dam**pened the spirits of Tim the Beaver. He's super excited to spend spring break on a beach in Miami, but he forgot to book flights which are now very expensive. Tim wonders whether he can save money traveling by bus. The Silverdog bus company offers bus routes around the country, where each bus route is a pair (c_1, c_2) representing a non-stop trip from city c_1 to city c_2. Assume it is possible to reach Miami from Boston on the Silverdog bus network.

(a) [4 points] Silverdog offers big discounts to students! A student ticket on any bus route costs exactly one dollar. Given a list of the n bus routes serviced by Silverdog, describe an $O(n)$-time algorithm to determine the minimum number of dollars Tim needs to spend on bus tickets to get from Boston to Miami.

(b) [7 points] Tim is despondent. When he goes to buy bus tickets, Tim discovers that the discount only applies to human students, so he will sadly have to pay full price. The full price of each bus ticket varies, but is always a positive integer number of dollars. Describe an $O(kn)$-time algorithm to determine whether there exists a sequence of bus routes that Tim can take from Boston to Miami that costs less than k dollars.

(c) [10 points] Tim complains on social media that Silverdog's business practices are discriminatory. The resulting public outrage prompts Silverdog to lift the restriction, allowing Tim to buy dollar student tickets! But in order to compensate for the loss in revenue, Silverdog places the following restriction on all student tickets: if a passenger enters a city on a bus using a student ticket, they may not use a student ticket to leave that city (so must pay full price). Describe an $O(kn)$-time algorithm to determine whether there exists a sequence of bus routes that Tim can take from Boston to Miami that costs less than k dollars subject to these new restrictions.

Problem 5-5. [45 points] **Peg Solitaire**

Peg solitaire is a single player game played on a **game board**: a square grid having R rows and C columns. The board begins in a given **initial configuration** where each grid square either contains a peg, or is empty and does not contain a peg. Then given any **straight line** of three adjacent grid squares (x, y, z) where squares x and y contain pegs and square z is empty, a player can make a **move** by moving the peg at square x into square z and removing the peg at square y from the board, resulting in a new board configuration containing one fewer peg. Specifically, a move can be specified by triple $m = (r, c, d)$ corresponding to moving the peg in row r and column c in the cardinal direction $d \in \{N, E, S, W\}$, if the move is possible. A board configuration is **solved** if exactly one of its grid squares contains a peg. An initial configuration is **solvable** if the player can make a sequence of moves to transform the board into a solved configuration.

In this problem, we represent a board configuration B as a length R tuple of length C tuples, where $B[r][c] = 1$ if the grid square in row r and column c contains a peg, and $B[r][c] = 0$ otherwise. Given a board configuration B and triple $m = (r, c, d)$, let `make_move(`B, m`,"forward")` be the board configuration B' resulting from making move m on B (or `None` if move m is not a valid move). Further, if $B' =$ `make_move(`B, m`,"forward")`, we call the transformation of B' into B the **inverse** application of move m, where `make_move(`B', m`,"backward")` $= B$. We have provided an implementation of `make_move` in your code template.

(a) [3 points] Apply moves $((0, 1, S), (1, 3, W), (1, 1, S))$ in sequence to the board configuration below, and draw the resulting board after each move.

```
    c  0  1  2  3  4
                          r
B = ((0, 1, 0, 0, 0),     0
     (0, 1, 1, 1, 0),     1
     (0, 0, 0, 0, 0),     2
     (0, 0, 1, 0, 0))     3
```

```
        |
        N
    --W   E--
        S
        |
```

(b) [3 points] Inversely apply moves $((3, 4, W), (3, 1, E), (3, 3, N))$ in sequence to the board configuration above, and draw the resulting board after each move inverse.

(c) [4 points] The **accessibility** of an initial board configuration B is the number of board configurations that can be reached via a sequence of moves from B. If N is the accessibility of an initial board configuration having R rows and C columns containing exactly k pegs, prove that $\log N = O(\min(RC, k \log k))$.

(d) [10 points] Given an initial board configuration B with R rows and C columns and accessibility N, describe an $O(kRCN)$-time algorithm to return a sequence of moves that solves B, or returns `None` if no such sequence exists.

(e) [25 points] Write a Python function `peg_solve(B)` that implements your algorithm. You can download a code template containing some test cases from the website. Submit your code online at `alg.mit.edu`.