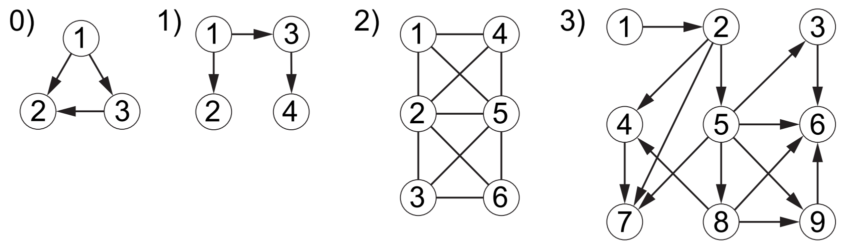
Introduction to Algorithms: 6.006
Massachusetts Institute of Technology — October 24, 2017
Instructors: Jason Ku, Muriel Medard, and Silvio Micali — Problem Set 6

Problem Set 6

All parts are due on November 2, 2017 at 11:59PM. 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.csail.mit.edu .

Problem 6-1. [25 points] **DFS Practice**

A depth first search (DFS) of a graph explores vertices along an unsearched path until reaching a vertex that has already been explored, at which point the search backtracks and continues along the next unsearched path. Which unsearched path is *next* is arbitrary and different choices can lead to different DFS trees. For graphs embedded in the plane, it is common to fix the order to search outgoing edges, for example in counterclockwise order from the edge from which you entered the node. We will refer to DFS using this ordering as a **CCW DFS**. Answer the following questions by performing depth first search on the following graphs, starting your search along the edge from vertex 1 to vertex 2, using CCW DFS.



(a) [12 points] During DFS, *tree* edges are edges traversed to a previously unexplored node and together form a DFS tree. Edges traversed to previously explored nodes have different labels: a *back* edge extends from a node to one of its ancestors in the DFS tree; a *forward* edge extends from a node to one of its descendants; and a *cross* edge is any other edge. Label edges in graphs G_1, G_2, and G_3 as either tree (T), back (B), forward (F), or cross (C), using a CCW DFS. For example, CCW DFS on G_0 has a single cross edge, $(3, 2)$, with all other edges being tree edges.

(b) [6 points] For graphs G_1, G_2, and G_3, list vertices in a topological sort based on the finishing times of a CCW DFS, or argue that a topological sort does not exist. The topological sort of G_0 based on CCW DFS is $[1, 3, 2]$.

(c) [7 points] Graph G_1 is actually the CCW DFS tree of a larger **simple** directed graph G. In fact, among all graphs admitting a CCW DFS tree equivalent to G_1, G is the graph with the most edges. Provide an argument for any edge (i, j) that does not exist in G, and draw G with vertices labeled.

Problem 6-2. [30 points] **Spy Networks**

Harriet works for the CIA and is monitoring terrorists with the help of an inside informant. Help Harriet bring down the terrorists. Time is of the essence, so you will be judged based on the speed of your solutions.

(a) [10 points] **Who's the Boss:** Harriet's informant has given her details on various terrorists and who they take orders from. A terrorist's *influence* is the set of terrorists he can give orders to, directly or indirectly; a terrorist will never take orders from someone within their influence, but may take orders directly from multiple people. A *terrorist cell leader* is a terrorist who takes orders from no one; a leader's influence is called a *terrorist cell*. Based on her informant's information, Harriet needs to generate a list of terrorist cells, and for each cell, construct a list of the cell's members so that a terrorist within the influence of another appears lower down on the list. Describe an algorithm to help Harriet generate these lists as quickly as you can.

(b) [10 points] **Informant Rescue:** Harriet's informant has been discovered by a terrorist cell. He's been captured and is being held, chained in an elaborate system of underground tunnels. Fortunately, the guards do not patrol the tunnels at night, so Harriet has been flown in to rescue him. Unfortunately, Harriet has no information about the tunnels or where in the tunnels her informant is being held, except that she will be able to reach him from the tunnel's only entrance. Inside the tunnels, there is a lantern hanging at each intersection to light the way. Time is short, but Harriet thinks there will be enough time to scratch a single arrow on the ground at each intersection. Describe an algorithm to help Harriet rescue her informant as quickly as you can.

(c) [10 points] **Targeted Capture:** Now that the informant is safe, the CIA is ready to take action. Harriet needs to recommend to her superiors a list of candidate terrorists for capture to help break up the cells. Cell leaders are very difficult to capture, so she has been asked to prioritize non-leader terrorists whose individual capture would decrease the size of a cell leader's influence by more than one. Assume that only one terrorist from the list will be pursued at this time. Describe an algorithm to help Harriet identify a list of candidate targets for capture as quickly as you can.

Problem 6-3. [45 points] **Linear Equations:** A linear equation on n variables $\{x_1, \dots, x_n\}$ specified by sequence of coefficients $C_i = (d_i, c_{i1}, \dots, c_{in})$ has the following form:

$$0 = d_i + \sum_{j=1}^{n} c_{ij} x_j$$

Given a set of linear equations, an assignment of the variables to real numbers is said to *solve* the set of equations if the assignment applied to each equation makes the equation true. You may assume that sets of linear equations encountered in this problem will have a single unique solution, with the number of equations equal to the number of variables.

(a) [9 points] An algorithm called Gaussian Elimination can solve a set of linear equations, but requires $\Omega(n^3)$ time in the worst case. Some sets of linear equations can be solved faster. Call a set of linear equations *delegated* if $c_{ii} = -1$ for all $i \in [1, n]$. Equation C_i from a delegated set of linear equations can be rewritten as:

$$x_i = d_i + \sum_{\substack{j=1 \\ i \neq j}}^{n} c_{ij} x_j$$

We say that x_i *depends on* x_j in a delegated set if $c_{ij} \neq 0$. A delegated set is *special* if there exists a permutation $\pi : [1, n] \to [1, n]$ such that x_i depends on x_j only when $\pi(i) < \pi(j)$, for all $i, j \in [1, n]$. Describe a simple directed graph on n vertices based on a delegated set of linear equations that will be acyclic if and only if the set of equations is special.

(b) [8 points] Describe an algorithm to determine whether a delegated set of linear equations is special in $O(n^2)$ time. Note that the input to the problem consists of $\Omega(n^2)$ coefficients, so $O(n^2)$ running time is linear in the size of the input.

(c) [8 points] Describe an algorithm that returns a solution to a special set of linear equations in $O(n^2)$ time. You may assume that the size of coefficient values in the problem are small, i.e. have $O(1)$ bit complexity so that any single arithmetic operation takes $O(1)$ time.

(d) [20 points] Implement function `solve_special` that takes as input a list of lists representing a set of linear equations and returns a list representing a solving assignment for the variables from x_1 to x_n if the equations are delegated and special, and `None` otherwise. Please submit code for this problem in a single file to alg.csail.mit.edu. ALG will only import your implementation of `solve_special`.

```
def solve_special(C):
    """
    Given a set of linear equations C, returns a solving assignment to variables
    if the set of equations is delegated and special, and None otherwise
    Input:
        C is a list of n lists, each with length n + 1
        C = [C_0, ... C_n]
        C_i = [d_i, c_i1, c_i2, ... c_in]
        C_i represents equation:  0 = d_i + \sum_j c_ij * x_j
    Output:
        x = [x_1, ... x_n]
        solving assignment of variables
    """
    ############################
    #  Implement me! Part (d)  #
    ############################
    x = None
    return x
```