Problem Set 7

All parts are due on November 1, 2018 at 11PM. 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 7-1. [10 points] **cdkx**

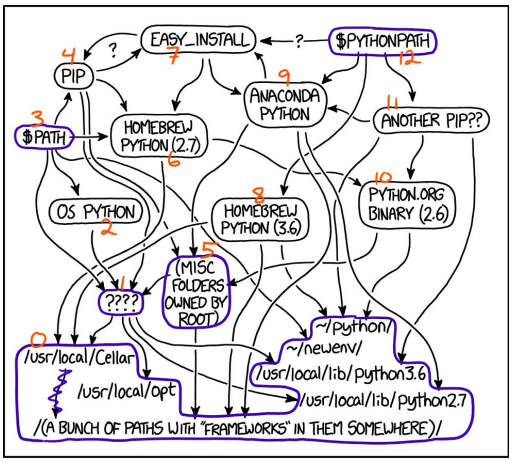
In this problem we'll attempt to topologically sort the directed, 13-vertex graph illustrated below.[1] For this problem we can ignore **parallel** or duplicate edges: for example, there are three separate edges from 8 to 0, but we only need to list this connection once.

```
Adj = {0:[], 1:[0], 2:[1], 3:[1,2,4,6], 4:[1,5,6,7], 5:[0], 6:[0,10],
       7:[4,6,9], 8:[0], 9:[0,5,7], 10:[0,5], 11:[0,9,10], 12:[7,8,9,11]}
```

(a) [2 points] Give a one-sentence reason why this graph can't be topologically sorted unless at least two edges are deleted.

(b) [5 points] Let's try anyway! Run a full DFS to search the entire graph, where an outer loop depth-first searches from unvisited vertices in order from 0 up to 12, and neighbors are always visited in increasing numerical order. Draw a graph of the parent pointers, and write down a list `F` of vertices arranged by finishing time.

(c) [3 points] The list `reversed(F)` is almost a topological ordering. Which edges would need to be removed from the graph to make it a valid topological ordering?



MY PYTHON ENVIRONMENT HAS BECOME SO DEGRADED THAT MY LAPTOP HAS BEEN DECLARED A SUPERFUND SITE.

Problem 7-2. [10 points] **Early Termination**

Suppose you are given a weighted directed graph $G = (V, E, w)$ with positive and negative edge weights, a vertex $s \in V$, and a number k. You are told that every vertex $v \in V$ has a minimum-weight path from s that uses at most k edges. Design an algorithm that solves weighted single-source shortest-paths from s in $O(|V| + k|E|)$ time.

[1] Adapted from `http://xkcd.com/1987`, ©2018 Randall Munroe.

Problem 7-3. [10 points] **Transfiguration**

Germione Hranger is taking a class in transfiguration at the Warthogs School of Wizardcraft and Witchery. She has learned many spells, where each spell transforms one material into another material in a $1 : r$ mass ratio for some $r > 0$. For example, she has learned a spell to transform x grams of salt into $3.2x$ grams of dragon scales, and spell to transform x grams of nightshade into $0.6x$ grams of gold. Given a list of all spells Germione has learned, describe an efficient[2] algorithm to determine the maximum amount of gold she can produce from s grams of salt.

Problem 7-4. [12 points] **Escape to the Surface**

After solving PS6, Lelda has successfully reached the underground cave where Zink was being held, and now they must escape via one of the above-ground cave entrances. As before, some caves are marked **dangerous**, where Zink and Lelda will need to fight enemies in order to pass. With their combined fighting abilities, Zink and Lelda can win any fight without losing a single heart, but Zink's sword will take damage during the scuffle.[3] If ever Zink's sword takes a total of k damage, it will break. Beetle's map indicates the strength of enemies at any cave, from which they can estimate how much damage will be dealt to Zink's sword in that cave. To maintain morale, they decide only to traverse tunnels that make **vertical upward progress** to the surface, i.e. they will only traverse a tunnel from cave a to cave b if cave b has strictly higher elevation than cave a (Beetle's map also indicates cave elevations). Describe an efficient algorithm to determine whether Lelda and Zink can escape to the surface always traveling up, without ever breaking Zink's sword.

Problem 7-5. [13 points] **Psychic Psweets**

It's Halloween, and Lee Evan, a special child with special powers, wants to bike from Jopper's shed to Whike Meeler's house along roads in her town, so together they can devour trick-or-treat candy. But she has to be careful: many enemy agents lie in wait to capture her and study her powers. Whenever she encounters an agent on a road, Lee must create a distraction using her psychic powers in order to escape, after which she must eat one piece of candy to replenish her energy. After Lee escapes, agents will not pursue her; they will remain at their assigned posts. In addition, Lee will trick-or-treat at any house on the **right side of any street**[4] she travels on, where she will gain one additional piece of candy. Once she is biking on a road in one direction, she will continue in that direction until reaching a road intersection, so biking along a road having a agents in a direction where h_R houses are on her right (and h_L on her left) will change her supply of candy by precisely $h_R - a$ pieces (or $h_L - a$ pieces if she travels the road in the opposite direction). Lee may trick-or-treat at a house more than once, but each time she does, she would need to bike along the **entire road** having that house on the right. Lee can start her journey with as much candy as she wants, and since candy is small, you may assume that there is no limit to the amount of candy Lee can carry. Lee knows where all the roads, houses, and agents are in the town (she is psychic after all). Describe an efficient algorithm to determine whether Lee can reach Whike's house with **strictly more candy** than when she started.

[2] By efficient, we mean that faster correct algorithms will receive more points than slower ones.

[3] Lelda uses a bow and arrow which does not take damage.

[4] Because left turns in the middle of a street are dangerous.

Problem 7-6. [45 points] **Bingle!**

Parry Lage is developing an amazing new search engine called Bingle! Every night, he needs to build source files S (from scratch) into a final executable file t via a collection of **code transformations**. Example code transformations include:

- converting a Python source file into a Python byte code file,
- converting a Java source file and its dependencies into a class interface file,
- converting a CoffeeScript source file into a JavaScript source file,
- converting a C source file into object code, etc.

Each code transformation (F, o, m) contains:

- a list F of input files (each either an original source file or an output from another code transformation) that need to exist before the transformation can be done,
- a unique output file o generated by the transformation,
- and the number of milliseconds m it takes to run the transformation.

Luckily, Parry has an extremely large number of machines, so he can **simultaneously** run as many code transformations as he wants in parallel. Help Parry determine the minimum amount of time needed to run code transformations to produce a final executable.

(a) [10 points] Given a weighted directed acyclic graph $G = (V, E, w)$, describe a linear-time algorithm to find a **maximum**-weight path from any vertex in a subset $S \subset V$ to a vertex t.

(b) [10 points] Given a list of S original source files, a list T of transformations, and a target file t, show how to use part (a) to compute the minimum number of milliseconds needed to build target file t from the source files S.

(c) [25 points] Implement method `build_time` based on the algorithm you described in part (b). You can download a code template containing some test cases from the website. Submit your code online at `alg.mit.edu`.

```
def build_time(source_files, transformations, target_file):
    """
    Return milliseconds needed to build the target file, assuming
    files not dependent on each other can be processed simultaneously.
    Input:      source | list of source file names
            transforms | list of transformations of form
                       | ([input_files], output_file, transform_time)
                target | name of target file to build
    """
    ##################
    # YOUR CODE HERE #
    ##################
    return 0
```