Problem Set 6

All parts are due on April 5, 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 6-1. [10 points] **Topological Tournament**

Eight soccer teams have qualified for the Global Goblet Soccer Tournament:

$\{$**A**ustralia, **B**razil, **G**ermany, **K**orea, **J**apan, **M**orocco, **S**outh Africa, **U**SA$\}$

Below is a list of games that were played between them during the season. A game where team x won against team y is represented by the ordered pair (x, y).

```
games = [
  (S, U), (K, J), (B, A), (S, J), (K, B), (B, U), (S, G), (K, U),
  (B, G), (G, J), (A, G), (J, U), (G, U), (A, U), (M, U), (G, B),
]
```

(a) [2 points] Draw a directed graph G on the teams where each game (x, y) corresponds to a directed edge from team x to team y.

(b) [5 points] Run Full DFS on G starting from Australia, and return the list of teams in reverse order of their DFS finishing times. Whenever there is ambiguity over which team to search next, break ties alphabetically. Is the returned list a topological sort?

(c) [3 points] A game (x, y) **goes against** an ordering of teams if y appears before x in the ordering. Use Full DFS to find an ordering of the teams which minimizes the number of games that goes against the ordering.

Problem 6-2. [10 points] **Limited Improvement**

A weighted directed graph $G = (V, E, w)$ is ***k*-limited from vertex *s*** $\in V$ if every vertex $v \in V$ has a minimum-weight path from s that traverses at most k edges. Given a weighted directed graph that is k-limited from s, describe an algorithm that solves weighted single-source shortest-paths from s in $O(|V| + k|E|)$ time.

Problem 6-3. [10 points] **Token Jumping**

Token Jumping is a token game played on a weighted directed graph $G = (V, E, w)$, where each edge weight is either a positive or negative integer. A token is placed on a starting vertex s, with an integer **score** that is initially zero. The token may **jump** around the graph along directed edges. Whenever the token jumps along a directed edge, the token adds the edge's weight to its score. Let $k > 0$ be the fewest jumps that the token needs to reach a target vertex t from s (assume that t is reachable from s). Describe an efficient algorithm to determine whether the token can reach t in exactly k jumps, arriving at t with a positive score.

Problem 6-4. [15 points] **Acute Fruit Pursuit**

MIT student Neil Millen is an avid avocado aficionado who already ate all his avocados! All he has in his dorm room is an apple leftover from a care package from his parents. Neil badly wants some avocado, but he is too lazy to leave his dorm. He emails his dorm mailing list asking if anyone has any fruit they would be willing to trade for another.[1] Neil receives many responses, where each response is a tuple of four integers (f_a, a, f_b, b) representing a student who is willing to trade 2^a fruit of type f_a for 2^b fruit of type f_b. For example, $(\text{orange}, 2, \text{pear}, 0)$ represents a student willing to give four oranges for each pear received (though they would not necessarily be willing to give one pear for four oranges). Assume that a student is willing (and able) to trade any amount of fruit at their posted rate, including partial fruits; so the aforementioned student would also be willing to give half an orange in exchange for an eighth of a pear. Given the list of n responses he received, describe an efficient algorithm to determine the maximum amount of avocado that Neil can obtain by trading his apple on the dorm fruit market.

Problem 6-5. [15 points] **Cross Country Criminal**

Tim the Beaver has built a fuel efficient vehicle called the WoodWagon for his engineering capstone project. Tim decides to drive it across country to test it out. The WoodWagon can travel up to m miles on a full tank of fuel before needing to refill. Not wanting to end up stranded, Tim checks a map to plan his route. Tim's map depicts all the roads and road intersections in the United States. Each of the n road intersections on the map is marked with its position $p = (x, y)$; assume the US is roughly flat so that the **straight-line distance** between two road intersections $p_1 = (x_1, y_1)$ and $p_2 = (x_2, y_2)$ is $d(p_2, p_1) = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$. Each road directly connects two road intersections and is marked on the map with its length; since roads are not always straight, the marked length of a road between p_1 and p_2 may be much longer than the straight-line distance $d(p_1, p_2)$ between its endpoints. Assume each road intersection connects at most five roads. The map marks k of the road intersections as having gas stations where Tim can refill. Tim is broke, so after filling up his gas tank at a station, he will leave without paying[2]! So whenever he leaves a gas station, Tim will only drive through road intersections that strictly increase his straight-line distance from the gas station he just left (until he fills up at the next gas station). Given Tim's map, describe an $O(nk)$-time algorithm to determine whether Tim can reach a destination intersection in California, starting with no fuel from a starting gas station in Massachusetts.

[1] guacamole green for bc-talk

[2] Tim is a very bad beaver. Please do not follow his example.

Problem 6-6. [40 points] **Awesome Alps**

Talent scouts are visiting the Hackson Jole ski resort, looking for new recruits! Vindsey Lonn is a burgeoning downhill skier who wants to show off her skills. She regularly trains at Hackson Jole so she knows the mountain well. On the mountain are many checkpoints. A **checkpoint** is represented by a pair (c, h) corresponding to its name c and its integer height h above sea-level. A **ski route** connects a pair of checkpoints having **different** heights, where Vindsey can ski **downhill** from the higher checkpoint to the lower one, but cannot ski the route in the other direction. Vindsey thinks some ski routes are more awesome than others, so she has assigned an integer **awesomeness** to each ski route: specifically triple (c_1, c_2, a) represents the awesomeness a assigned to a ski route connecting checkpoints with names c_1 and c_2 (when Vindsey skies that route downhill). A **downhill course** is any sequence of checkpoints of decreasing height, where each adjacent pair of checkpoints is connected by a ski route along the course. The **awesomeness** of any downhill course is the sum total of awesomeness of routes along the course.

(a) [5 points] Below is a list C of mountain checkpoints and a list R of awesomeness-assigned ski routes. There are nine downhill courses on the mountain. List the three most awesome downhill courses, and indicate how awesome they are.

```
C = [
    ("north_summit", 34),
    (  "west_bluff", 23),
    ( "eagle_ridge", 17),
    ("alpine_cliff", 24),
]
```

```
R = [
    ("alpine_cliff", "north_summit", 22),
    ( "eagle_ridge", "north_summit", 43),
    ("alpine_cliff",   "west_bluff", 12),
    (  "west_bluff", "north_summit", 37),
    ( "eagle_ridge",   "west_bluff", 38),
]
```

(b) [10 points] Given a list C of checkpoints on the mountain and a list R of awesomeness-assigned ski routes, describe an efficient algorithm to return any most awesome downhill course, so Vindsey can impress the talent scouts.

(c) [25 points] Write a Python function `most_awesome(C, R)` 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`.