CS161 Homework 4 **Due:** 13 May 2016
Submit on Gradescope **Handed out:** 6 May 2016

Instructions: Please answer the following questions to the best of your ability. If you are asked to design an algorithm, please describe it clearly and concisely, prove its correctness, and analyze its running time. If you are asked to show your work, please include relevant calculations for deriving your answer. If you are asked to explain your answer, give a short ($\sim$ 1 sentence) intuitive description of your answer. If you are asked to prove a result, please write a complete proof at the level of detail and rigor expected in prior CS Theory classes (i.e. 103). When writing proofs, please strive for clarity and brevity (in that order). Cite any sources you reference.

1 (14 points) The streets of Bearville

Barr the bear is the mayor of the newly established city of Bearville near the North Pole. He learns that Po the Penguin is a master civil engineer from the South Pole and hires Po to design the Bearville city streets. Po comes back and in his design all the streets are one-way! Barr demands that Po proves to him very quickly that one can reach any street intersection from any other street intersection. (Note that multiple streets can meet at an intersection.) In this question we will help Po:

(a) Give a graph formulation of the above problem: what are the vertices and edges of the graph? What computational problem does Po need to solve?

(b) Let m be the number of edges and n be the number of vertices in the graph. Give an algorithm to solve the intersection reachability problem in time $O(m+n)$. Prove why your algorithm is correct and why it runs in $O(m+n)$ time.

2 (12 points) Touring Parallel Bearville

In a parallel universe, Po-Prime has designed the streets to be one-way and moreover, *for every* intersection i there is *no way* to start at i and get back to i by following streets. This is very disconcerting to Barr-Prime. Po-Prime, however claims that there is an intersection s and a way to tour the city starting from s, traversing every intersection *exactly once*. We call this tour an *exact traversal*. (If we believe Po-Prime, then we can fix his design as follows: find the starting and ending intersections s and t of the exact traversal and add a street going from t to s – now every intersection is reachable from any other intersection.)

In this problem we will check whether Po-Prime is right.

Let n be the number of intersections and m be the number of streets as in the previous problem. Design an $O(m+n)$ time algorithm that determines whether an exact traversal exists, and if so, returns the order in which the intersections are traversed. Prove why your algorithm is correct and why it runs in $O(m+n)$ time.

3 (12 points) Negative Edges

In lecture, we discussed how Dijkstra's Algorithm does not handle graphs with negative edge weights. In this question, we'll explore why this is the case. For this question assume that all edge weights are integers.

(a) Draw a graph G, which contains both positive and negative edges but does not contain negative cycles, and specify some source $s \in V$ where $\mathsf{Dijkstra}(G, s)$ does not correctly compute the shortest paths from s. Your graph should have the minimum number of vertices possible. Show which paths are computed incorrectly and explain why Dijkstra fails.

(b) Consider the algorithm Negative-Dijkstra for computing shortest paths through graphs with negative edge weights (but without negative cycles).

Algorithm 1: Negative-Dijkstra(G, s)

$w^* \leftarrow$ minimum edge weight in G;
for $e \in E(G)$ **do**
  $w'(e) \leftarrow w(e) - w^*$;
$T \leftarrow$ Dijkstra(G', s) `// Run Dijkstra with updated edge weights to get a SSSP Tree`
return weights of T in the original G;

Note that Negative-Dijkstra shifts all edge weights to be non-negative (by shifting all edge weights by the smallest original value) and runs in $O(m + n \log n)$ time.

Prove or Disprove: Negative-Dijkstra computes single-source shortest paths correctly in graphs with negative edge weights. To prove the algorithm correct, show that for all $u \in V$ the shortest $s - u$ path in the original graph is in T. To disprove, exhibit a graph with negative edges, with no negative cycles where Negative-Dijkstra outputs the wrong "shortest" paths, and explain why the algorithm fails.

4 (14 points) Po and Barr party

Po the penguin and Barr the bear have the same set of n friends, $F = \{f_1, f_2, \ldots, f_n\}$, but Po and Barr don't really like each other. Po decides to have a party. After learning this, Barr decides to have a party at the same time as Po. Among their common friends various pairs don't like each other so would prefer not to go to the same party. You are given a set of enemy pairs $E = \{(f_{i_1}, f_{j_1}), \ldots, (f_{i_m}, f_{j_m})\}$ and you need to decide whether the friends in F can be partitioned into two parts B and P (i.e. each friend is in either P or B but not both) so that no pair from E appears together in B or P (and hence the friends in B can go to Barr's party and the friends in P can go to Po's party and both parties will be without conflict).
That is, give an $O(m+n)$ time algorithm that given $F = \{f_1, f_2, \ldots, f_n\}$ and $E = \{(f_{i_1}, f_{j_1}), \ldots, (f_{i_m}, f_{j_m})\}$

- either finds a partition of F into B and P so that for every $(f_{i_r}, f_{j_r}) \in E$, $|B \cap (f_{i_r}, f_{j_r})| = 1$ and $|P \cap (f_{i_r}, f_{j_r})| = 1$,
- or determines that no such partition exists.

Prove the correctness and runtime of your algorithm.