ASSIGNMENT 2 CMSC 451 (Section 0101, Fall 2020)

Due by 11:59 pm on Wednesday, September 30. Submit your solutions in PDF via Gradescope.

1. *Gears.* [20 points]

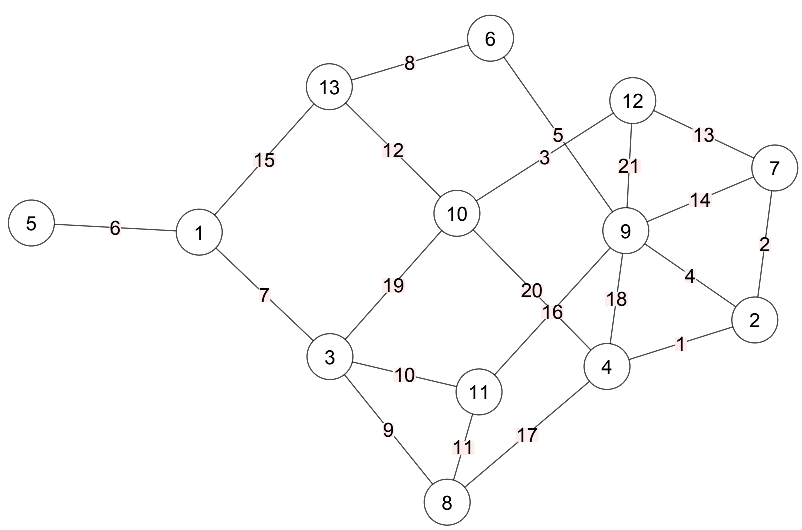
   Suppose you are given a large collection of n circular gears. The gears all operate in the same plane; each rotates about a fixed center position with its teeth pointing outward. For each pair of gears, you are told whether or not they mesh (so that they will turn together). One of the gears has a crank on it. Design an efficient algorithm that will determine (i) whether it is possible to turn the crank without breaking the gears, and (ii) whether *all* the gears move when the crank is turned. Prove that your algorithm is correct and determine its asymptotic running time.

2. *Route planning with variable travel times.* [20 points]

   Suppose you would like to navigate a system of roads, represented by a directed graph $G = (V, E)$. Because of variable traffic and weather conditions, the time required to travel along an edge varies with time. Fortunately, you are given access to a procedure that can reliably predict how long it will take to travel along an edge at any chosen starting time. Given an edge $e = (u, v) \in E$ and a starting time t, the procedure returns the amount of time $r_e(t) \geq 0$ that it will take to travel from vertex u to vertex v along edge e, assuming you leave u at time t. The travel times have the property that if $t' > t$, then $t' + r_e(t') > t + r_e(t)$ (i.e., if you leave later, you will arrive later), but are otherwise arbitrary. The running time of the procedure is $O(1)$. Design a polynomial-time algorithm to find the fastest way to get from a given starting vertex to a given destination vertex, starting at time 0. Prove that your algorithm is correct and analyze its running time. (Hint: consider adapting Dijkstra's algorithm to this setting.)

3. *Minimum spanning tree.* [15 points]

   Run Kruskal's algorithm to find a minimum spanning tree of the following weighted graph.



   In addition to showing the minimum spanning tree produced as output, indicate the order in which the edges are added to the tree. (*Note:* You should either run the algorithm by hand, or implement it in a programming language of your choice. You may *not* use someone else's implementation.)

4. *Scheduling fire drills.* [20 points]

   Suppose you are given a set of intervals $[s_i, f_i]$ specifying starting and finishing times for classes indexed by $i \in \{1, 2, \ldots, n\}$. You would like to schedule fire drills at times $t_1, t_2, \ldots, t_k$ so that a drill happens during every class (i.e., for all $i \in \{1, 2, \ldots, n\}$ there is a $j \in \{1, 2, \ldots, k\}$ such that $t_j \in [s_i, f_i]$). It is okay if more than one drill is scheduled during a given class, but you would like to minimize the total number of fire drills (i.e., the value k). Design an efficient algorithm for this problem, prove its correctness, and analyze its running time.

5. *Finding a majority.* [20 points]

   Given an n-element array A, a majority element is one that appears *strictly more than* $n/2$ times. Note that the majority element is unique if it exists. Your task is to design an efficient algorithm to tell whether a given array A has a majority element, and if so, to find that element. The elements of A are large and complicated, so you cannot easily make comparisons of the form "is $A[i] \geq A[j]$?". However, you can answer questions of the form "is $A[i] = A[j]$?" in constant time.

   Show how to solve this problem in $O(n \log n)$ time. Prove correctness and the bound on the running time.

6. *Collaboration.* [5 points]

   Write "I understand the course collaboration policy and have followed it when working on this assignment." List the other students with whom you discussed the problems, or else indicate that you did not discuss any problems with your classmates.