Algorithms and Data Structures: Assignment 2, due Sept. 27

- Please read the assignment policies on the course homepage before starting the assignment.
- Any algorithm must be accompanied by a proof of correctness and a runtime analysis.

---

1. Reading: (i) Chapter 10 of CLRS, particularly 10.4. (ii) Sections 17.1, 17.2, 17.3 from CLRS

2. (10 marks) Show that $n-1$ comparisons are necessary and sufficient to find the minimum element in an unsorted array of n elements.

3. (15 marks) Show a comparison based algorithm for finding the minimum *and* maximum in an unsorted array of n elements using $\lceil 3n/2 \rceil - 2$ comparisons. Also show that $\lceil 3n/2 \rceil - 2$ comparisons are necessary to find the minimum and maximum.

4. (10 marks) Let $G = (V, E)$ be a directed acyclic graph $G = (V, E)$. Additionally, you are given a nonnegative, integral weight w_e on each edge $e \in E$, and two special vertices s, $t \in V$. Give an algorithm to find a max-weight path from s to t.

5. (15 marks) Given a matroid $(S, \mathcal{I})$, show that $(S, \mathcal{I}')$ is also a matroid, where $A \in \mathcal{I}'$ if $S \setminus A$ contains a maximal independent set in $\mathcal{I}$.

6. (15 marks) In class, we showed that if $(S, \mathcal{I})$ is a matroid, then for any nonnegative weights w on the elements of S, the greedy algorithm obtains a maximum weight independent set. Show that this is only true if $(S, \mathcal{I})$ is a matroid. That is, for a fixed downward-closed set system $(S, \mathcal{I})$, if the greedy algorithm obtains a maximum weight element of $\mathcal{I}$ for every assignment of nonnegative weights to elements of S, then $(S, \mathcal{I})$ is a matroid.

7. (10 marks) Exercise 10.4-6 (on tree representations with pointers) from CLRS.

8. (10 marks) Given a directed graph $G = (V, E)$ with weights on the edges, and which has a negative-weight directed cycle that is reachable from the source s. Give an efficient algorithm to list the vertices of such a cycle.

9. (15 marks) Let us modify the "cut rule" (in the implementation of decrease-key operation for a Fibonacci heap) to cut a node x from its parent as soon as it loses its 3rd child. Recall that the rule that we studied in class was when a node loses its 2nd child. Can we still upper bound the maximum degree of a node of an n-node Fibonacci heap with $O(\log n)$?

10. (15 marks) The following are Fibonacci-heap operations: *extract-min*$(\cdot)$, *decrease-key*$(\cdot, \cdot)$, and also *create-node*(x, k) which creates a node x in the root list with key value k. Show a sequence of these operations that results in a Fibonacci heap consisting of just one tree that is a linear chain of n nodes.