“CS 374” Fall 2015 — Homework 8

Due Tuesday, November 3, 2015 at 10am

• • • Some important course policies • • •

- **You may work in groups of up to three people.** However, each problem should be submitted by exactly one person, and the beginning of the homework should clearly state the names and NetIDs of each person contributing.
- **You may use any source at your disposal**—paper, electronic, or human—but you ***must*** cite ***every*** source that you use. See the academic integrity policies on the course web site for more details.
- **Submit your pdf solutions in Moodle.** See instructions on the course website and submit a separate pdf for each problem. Ideally, your solutions should be typeset in LaTex. If you hand write your homework make sure that the pdf scan is easy to read. Illegible scans will receive no points.
- **Avoid the Three Deadly Sins!** There are a few dangerous writing (and thinking) habits that will trigger an automatic zero on any homework or exam problem. Yes, we are completely serious.
  - Give complete solutions, not just examples.
  - Declare all your variables.
  - Never use weak induction.
- Unlike previous editions of this and other theory courses we are not using the “I don’t know” policy.

See the course web site for more information.

If you have any questions about these policies,
please don’t hesitate to ask in class, in office hours, or on Piazza.

---

1. Let $G = (V, E)$ be a directed graph with non-negative edge lengths; $\ell(e)$ denotes the length of edge e. Given two nodes $s, t \in V$ an s-t walk P is a sequence of nodes $s = v_0, v_1, \ldots, v_k = t$ such that for each $0 \le i < k$ $(v_i, v_{i+1}) \in E$. Note that a walk allows nodes and edges to be repeated. Given an s-t walk P we denote by $\#P$ the number of edges in the walk. We say that an s-t walk P is a Mod7 walk if $\#P$ is congruent to 0 (mod 7).

   - Give an example of a graph $G = (V, E)$ and nodes $s, t \in V$ such that there is no Mod7 s-t walk in G.
   - Give an example of a graph $G = (V, E)$ and nodes $s, t \in V$ such that there is a Mod7 s-t walk but any Mod7 s-t walk has more than $2n$ edges. In particular a node may appear more than twice on the walk.
   - Show that if there is a Mod7 s-t walk in G there is one in which there are at most $7n$ edges. *Hint:* Argue that if a node appears 7 or more times in a walk you can eliminate a portion of the walk without changing the Mod7 property.
   - Use the preceding property to design an efficient algorithm that correctly decides whether G has a Mod7 s-t walk and outputs the shortest one if one exists.

2. We saw in lecture that Borouvka's algorithm for MST can be implemented in $O(m \log n)$ time where m is the number of edges and n is the number of nodes. We also saw that Prim's algorithm can be implemented in $O(m + n \log n)$ time. Obtain an algorithm for MST with running time $O(m \log \log n)$ by running Borouvka's algorithm for some number of steps and then switching to Prim's algorithm. This algorithm is better than either of the algorithms when $m = \Theta(n)$. Formalize the algorithm, specify the parameters and argue carefully about the implementation and running time details. Briefly justify the correctness of the algorithm assuming that the edge weights are unique.

3. Let $G = (V, E)$ an undirected edge-weighted graph. The bottleneck weight of a spanning tree T is the weight of the maximum weight edge in T. The minimum spanning tree of a graph is also a spanning tree which minimizes the bottleneck weight; try to prove this for yourself. You can compute the MST and hence a minimum bottleneck weight spanning tree in $O(m + n \log n)$ time. Can we do better? It turns out to be yes via a nice clever algorithm that you will work out below.

   - Consider the following algorithm for finding a minimum bottleneck spanning tree. First, given a number α describe a linear-time algorithm that checks whether there is a spanning tree in G with bottleneck weight at most α; this is a decision procedure. Using the decision procedure as a black box, use binary search on the sorted edge weights to find a minimum bottleneck weight spanning tree in $O(m \log n)$ time. Your goal here is to understand the algorithm and write down a formal description and justify its running time.
   - Now improve the running time of the algorithm to be linear by using the following idea. Instead of sorting the edge weights find the median of the edge weights in linear time using the selection algorithm we discussed previously in the course. Let us call the median edge weight β. Now use the decision procedure to check whether there is a bottleneck spanning tree of weight at most β. In either case show how you can effectively recurse on a graph with at most half the edges of the original graph to obtain a linear time algorithm. Write down the algorithm carefully and justify its running time. No proof of correctness necessary but explain how the steps of your algorithm so that it can be properly understood.

Some other problems to think about. **Not part of the home work**.

1. - Describe and analyze a modification of Bellman-Ford shortest-path algorithm that actually returns a negative cycle if any such cycle is reachable from s, or a shortest-path tree if there is no such cycle. The modified algorithm should still run in $O(|V||E|)$ time.
   - Describe and analyze a shortest-path algorithm that computes the correct shortest path distances from s to every other vertex of the input graph, even if the graph contains negative cycles. Specifically, if any walk from s to v contains a negative cycle, your algorithm should end with $dist(v) = -\infty$; otherwise, $dist(v)$ should contain the length of the shortest path from s to v. The algorithm should still run in $O(|V||E|)$ time.

2. Suppose we are given a directed graph $G = (V, E)$ with non-negative edge lengths. Moreover, the edges are partitioned into red, blue and white. An s-t walk in G is a flag walk if the colors of the edges in the walk follow the pattern “red, blue, white” which is repeated; the walk may end in a red or blue or white edge. Describe an efficient algorithm to find the shortest s-t flag walk or determine that there is no such walk.