CSOR W4231.002 – Spring, 2018

Homework 5

Out: Monday, Apr 2, 2018
Due: 8pm, Monday, April 16, 2018

For all algorithms you design, you should always describe them clearly in English, give pseudocode, prove correctness and give the best upper bound that you can for the running time.

Collaboration is limited to discussion of ideas only. You should write up the solutions entirely on your own. You should list your collaborators on your write-up. If you do not type your solutions, be sure that your hand-writing is legible, your scan is high-quality and your name is clearly written on your homework.

I encourage you to work on all the recommended exercises before you solve problems 4 and 5.
You may not use any external resources such as the internet or other textbooks to solve this homework. You should adhere to the department's academic honesty policy (see the course website, too).

1. (a) (10 points) Let $G = (V, E, w)$ be a weighted undirected graph. Prove that if all edge weights are distinct, then the minimum spanning tree is unique.

   (b) (10 points) Show how to find the *maximum* spanning tree of a graph that is, the spanning tree of largest total weight.

2. (20 points) Let G be a connected graph, and T and T' two different spanning trees of G. We say that T and T' are *neighbors* if T contains exactly one edge that is not in T', and T' contains exactly one edge that is not in T.

   Now build the hypergraph H as follows: the nodes of H are the spanning trees of G, and there is an edge between two nodes of H if the corresponding spanning trees are neighbors.

   Prove that H is always connected, or provide an example of a connected graph G for which H is not connected.

3. (a) (10 points) A binary counter of unspecified length supports two operations: `increment` (which increases its value by 1) and `reset` (which sets its value back to zero). Show that, starting from an initially zero counter, any sequence of n `increment` and `reset` operations takes time $O(n)$; that is, the amortized time per operation in $O(1)$.

   (b) (10 points) Suppose you implement the disjoint-sets data structure on n elements **without** path compression. Give a sequence of m `Union` and `Find` operations that take $\Omega(m \log n)$ time.

4. *Before you solve this problem, you should work on recommended exercise 2.*

   (25 points) In many applications, on top of the node demands introduced in the previous problem, the flow must also make use of certain edges. To capture such constraints, consider the following variant of the previous problem.

   You are given a *flow network* $G = (V, E)$ *with demands* where every edge e has an integer capacity c_e, *and* an integer *lower bound* $\ell_e \geq 0$. A circulation f must now satisfy $\ell_e \leq f(e) \leq c_e$ for every $e \in E$, as well as the demand constraints. Determine whether a feasible circulation exists.

5. (25 points) Similarly to a flow network with demands, we can define a *flow network with supplies* where each node $v \in V$ now has an integer *supply* s_v so that if $s_v > 0$, v is a *source* and if $s_v < 0$, it is a *sink*, and the supply constraint for every $v \in V$ is $f^{\text{out}}(v) - f^{\text{in}}(v) = s_v$.

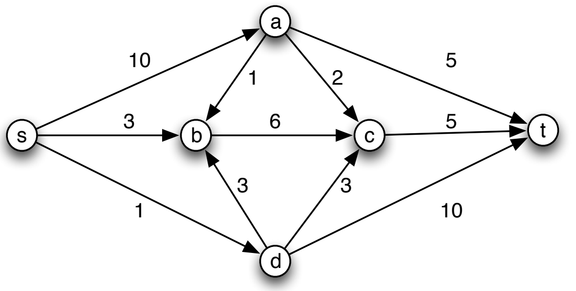
   In a *min-cost flow* problem, the input is a flow network with supplies where each edge $(i, j) \in E$ also has a cost a_{ij}. Given a flow network with supplies and costs, the goal is to find a feasible flow $f : E \rightarrow R^+$ —that is, a flow satisfying edge capacity constraints and node supplies— that minimizes the total cost of the flow.

   (a) Show that max flow can be formulated as a min-cost flow problem.

   (b) Formulate a linear program for the min-cost flow problem.

RECOMMENDED exercises: do NOT return, they will not be graded!

1. Run the Ford-Fulkerson algorithm on the following network, with edge capacities as shown, to compute the max *s*-*t* flow. At every step, draw the residual graph and the augmenting paths. Report the maximum flow along with a minimum cut.



2. A *flow network with demands* is a directed capacitated graph with potentially multiple sources and sinks, which may have incoming and outgoing edges respectively. In particular, each node $v \in V$ has an integer *demand* d_v; if $d_v < 0$, v is a *source*, while if $d_v > 0$, it is a *sink*. Let S be the set of source nodes and T the set of sink nodes.

   A *circulation with demands* is a function $f : E \to R^+$ that satisfies

   (a) *capacity constraints:* for each $e \in E$, $0 \leq f(e) \leq c_e$.

   (b) *demand constraints:* For each $v \in V$, $f^{\text{in}}(v) - f^{\text{out}}(v) = d_v$.

   We are now concerned with a decision problem rather than a maximization one: *is there* a circulation f with demands that meets both capacity and demand conditions?

   i. Derive a necessary condition for a feasible circulation with demands to exist.

   ii. Reduce the problem of finding a feasible circulation with demands to Max Flow.

3. There are many variations on the maximum flow problem. For the following two natural generalizations, show how to solve the more general problem by **reducing** it to the original max-flow problem (thereby showing that these problems also admit efficient solutions).

   - There are multiple sources and multiple sinks, and we wish to maximize the flow between all sources and sinks.
   - Both the edges *and the vertices* (except for s and t) have capacities. The flow into and out of a vertex cannot exceed the capacity of the vertex.