HW 9 (due Wednesday, at noon, November 7, 2018)

CS 473: Algorithms, Fall 2018 Version: **1.1**

Submission guidelines and policies as in homework 1.

1 (100 PTS.) Tikun Flow.

You are given an instance of network flow G, s, t with all capacities being integer numbers.

1.A. (30 PTS.) Given a flow f in G, and its residual network G_f, describe an algorithm for computing the augmenting path from s to t that has maximum residual capacity. Prove the correctness of your algorithm.

1.B. (20 PTS.) Prove, that if the maximum flow in G_f has value τ, then the augmenting path you found in (A) has capacity at least τ/m.

1.C. (10 PTS.) Consider the algorithm that starts with the empty flow f, and repeatedly use the above to find an augmenting path, until s and t are disconnected. Prove that this algorithm computes the maximum flow in G.

1.D. (20 PTS.) Consider the algorithm from the previous part, and the flow g it computes after m iterations. Prove that $|g| \geq C/\beta$, for some constant $\beta > 1$, where C is the value of the maximum flow. What is the value of β? (The smaller the better.)

1.E. (20 PTS.) Give a bound, as tight as possible, on the running time of the above algorithm for computing maximum flow, as a function of n, m, and the maximum flow value in G.

2 (100 PTS.) Matchless cut.

2.A. (10 PTS.) Let $\mathsf{G} = (V, E)$ be a directed graph, with source $s \in V$, sink $t \in V$, and nonnegative edge capacities $\{c_e\}$. Give a polynomial-time algorithm to decide whether G has a *unique* minimum s-t cut (i.e., an s-t of capacity strictly less than that of all other s-t cuts).

2.B. (30 PTS.)

Suppose you're looking at a flow network G with source s and sink t, and you want to be able to express something like the following intuitive notion: Some nodes are clearly on the "source side" of the main bottlenecks; some nodes are clearly on the "sink side" of the main bottlenecks; and some nodes are in the middle. However, G can have many minimum cuts, so we have to be careful in how we try making this idea precise.

Here's one way to divide the nodes of G into three categories of this sort.

- We say a node v is ***upstream*** if, for all minimum s-t cuts (A, B), we have $v \in A$ – that is, v lies on the source side of every minimum cut.
- We say a node v is ***downstream*** if, for all minimum s-t cuts (A, B), we have $v \in B$ – that is, v lies on the sink side of every minimum cut.
- We say a node v is ***central*** if it is neither upstream nor downstream; there is at least one minimum s-t cut (A, B) for which $v \in A$, and at least one minimum s-t cut (A', B') for which $v \in B'$.

Give an algorithm that takes a flow network G and classifies each of its nodes as being upstream, downstream, or central. The running time of your algorithm should be within a constant factor of the time required to compute a *single* maximum flow.

2.C. (30 PTS.)

You are given three disjoint sets X, Y, Z of n vertices in a weighted graph G (G has N vertices and M edges) – the distance between any two vertices is the shortest path metric of the graph. Our purposes is to output a set $\mathcal{T}$ of n disjoint triplets of points $(x_i, y_i, z_i) \in X \times Y \times Z$, such that all the vertices in $X \cup Y \cup Z$ are included in exactly one such triplet, and furthermore, the price function

$$f(\mathcal{T}) = \max_{i=1}^{n}\Bigl(d(x_i, y_i) + d(y_i, z_i) + d(z_i, x_i)\Bigr)$$

is minimized, where $d(p, q)$ denotes the shortest path distance between p and q in G. Provide a polynomial time constant (the smaller the better) approximation algorithm for this problem. What is the approximation constant? (Hint: Network flow.)

2.D. (30 PTS.) You are given a set P of n vertices in a weighted graph G (as above, with N vertices and M edges), and a set Q of m vertices (i.e., base stations). The ith base station b_i, can serve at most α_i clients, and each client has to be in distance at most r_i from it, for $i = 1, \ldots, m$ (as before, we use the shortest path distances in the graph). Describe a polynomial time algorithm that computes for each client which base station it should use, and no base station is assigned more clients that it can use. The target is of course to serve as many clients as possible. What is the running time of your algorithm?

3 (100 PTS.) Easy peasy.

3.A. (50 PTS.) Show that a maximum flow in a network $\mathsf{G} = (\mathsf{V}, \mathsf{E})$ can always be found by a sequence of at most $|\mathsf{E}|$ augmenting paths. [Hint: Determine the paths after finding the maximum flow.]

3.B. (50 PTS.) Suppose that a flow network $\mathsf{G} = (\mathsf{V}, \mathit{Edges})$ has symmetric edges, that is, $(u, v) \in \mathsf{E}$ if and only if $(v, u) \in \mathsf{E}$. Show that the Edmonds-Karp algorithm terminates after at most $|\mathsf{V}||\mathsf{E}|/4$ iterations. [Hint: For any edge (u, v), consider how both $\delta(s, u)$ and $\delta(v, t)$ change between times at which (u, v) is critical.]