Homework 1

Topics in Graph Algorithms
CS598CCI, Spring 2020
Due: 2/10/2020 10am

Instructions and Policy: Each person should write up their own solutions independently. You need to indicate the names of the people you discussed a problem with. Solutions to most of these problems can be found from one source or the other. Try to solve on your own first, and cite your sources if you do use them.

Please write clearly and concisely. Refer to known facts. You should try to convince me that you know the solution, as quickly as possible.

Do as many problems as you can. I expect you do at least 3.

Optional Problem. We saw an algorithm in class to find the transitive closure of a directed graph $G = (V, E)$ with n nodes in $O(n^\omega \log n)$ time. Obtain an algorithm with $O(n^\omega)$ time. For this consider a reduction to a DAG first and then do a divide and conquer on a topological sort of the DAG.

Problem 1. Suppose $G = (V, E)$ is a directed graph with unit edge lengths. The diameter of the graph is defined as $\max_{u,v \in V} d_G(u, v)$.

- Describe an algorithm with running time $O(n^\omega \log n)$ to compute the diameter of G.
- For any fixed $\epsilon > 0$ describe an algorithm with running time $O(n^\omega \log n)$ time to compute a $(1 + \epsilon)$-approximation to all distances. That is, your algorithm should output a number $d(i, j)$ for each pair of vertices (i, j) such that $d_G(i, j) \leq d(i, j) \leq (1 + \epsilon) d_G(i, j)$. How does the running time of your algorithm depend on ϵ?

Problem 2. We have seen Seidel's approximation for APSP in undirected unweighted graphs via matrix multiplication. Can one obtain subcubic algorithms for APSP *without* using matrix multiplication? This is open, however, some relaxed versions of APSP have been studied. The goal in this problem is to (re)derive an algorithm that yields an additive $+2$ approximation for all distances using a relatively simple combinatorial algorithm. Follow the ideas below. Let $G = (V, E)$ be an undirected unweighted graph with m edges and n vertices.

- Partition the vertices into high degree vertices $H = \{v \mid \deg(v) \geq \sqrt{n}\}$ and the low degree vertices $L = V \setminus H$. Show that there is a vertex set $D \subseteq V$ of size $O(\sqrt{n} \log n)$ such that the following holds for each $u \in H$: $u \in D$ or some neighbor of u is in D. In other words D dominates H. Describe an algorithm to obtain such a D in $O(m \log n)$ (expected) time — you can use a randomized algorithm or a greedy algorithm.

- Use BFS to find shortest paths from all vertices in D.
- Find shortest paths for each pair (i, j) among paths that do not include any vertex from H.
- Put together the distances from the preceding two steps to obtain an estimate $d(i, j)$ for each vertex pair (i, j) such that $d_G(i, j) \leq d(i, j) \leq d_G(i, j) + 2$ such that the run time of the algorithm is $O(n^{2.5} \log n)$.

Problem 3. Given a directed graph $G = (V, E)$ finding the longest s-t path is NP-Hard via an easy reduction from Hamilton Path. In this problem we explore a simple yet general technique called color coding that plays a fundamental role in fixed parameter tractability. The goal is to derive an algorithm that checks whether there is an s-t path in G of length at least k in time $c^k\text{poly}(n)$ for some fixed c. When k is small this is useful.

- Suppose we independently color each vertex $v \in V$ with a color from $\{1, 2, \ldots, k\}$. What is the probability that a specific set $S \subset V$ with $|S| = k$ has all vertices with different colors?
- Suppose each $v \in V$ has a color from $\{1, 2, \ldots, k\}$. Describe an algorithm with running time $2^k\text{poly}(n)$ to check whether there is an s-t path P such that all vertices in P have distinct colors.

Problem 4. Let $G = (V, E)$ be an undirected graph with non-negative edge lengths $\ell : E \to R^+$. Given nodes s, t we wish to find a shortest s-t path with an odd number of edges. Show that this problem can be solved via matching techniques by following the hint below. Suppose s, t have degree 1. Consider the reduction of the maximum weight matching problem to the maximum weight perfect matching problem. Adapt this reduction and show how a minimum cost perfect matching can be used to obtain a shortest odd length path from s to t. How can you get rid of the assumption that s, t have degree 1? Extend the ideas for finding a shortest even length s-t path.