CS7800: Advanced Algorithms. Fall 2017
Homework 3

Instructor: Jonathan Ullman TA: Albert Cheu

Due Friday, September 29th at 11:59pm
(Email to neu.cs7800@gmail.com)

Homework Guidelines

Collaboration Policy. Collaboration on homework problems is permitted, however it will serve you well on the exams if you solve the problems by yourself. If you choose to collaborate, you may discuss the homework with at most 2 other students currently enrolled in the class. Finding answers to problems on the Web or from other outside sources (these include anyone not enrolled in the class) is strictly forbidden.

You must write up each problem solution by yourself without assistance, even if you collaborate with others to solve the problem. You must also identify your collaborators. If you did not work with anyone, indicate that on your submission. If asked, you must be able to explain your solution to the instructors.

Preparing and Submitting Solutions. You must type your solutions using LaTeX. Please use an 11-pt or larger font. Please submit both the source and PDF files using the naming conventions `lastname_hw3.tex` and `lastname_hw3.pdf`. Your name must be on the first page of the PDF.

Solution guidelines For problems that require you to provide an algorithm, you must give the following: (1) a precise description of the algorithm in English and, if helpful, pseudocode, (2) a proof of correctness, (3) an analysis of running time. You may use any facts from class in your analysis and you may use any algorithms from class as subroutines in your solution.

You should be as clear and concise as possible in your write-up of solutions. Communication of technical material is an important skill, so clarity is as important as correctness. A simple, direct analysis is worth more points than a convoluted one, both because it is simpler and less prone to error and because it is easier to read and understand. Points might be subtracted for solutions that are too long.

Problem 1 (20 pts)**.**

A *linear k-spanner* is a directed graph with vertex set $V = \{1, \ldots, n\}$ that contains a directed path of length at most k from vertex i to vertex j for all $1 \leq i < j \leq n$. Moreover, a linear k-spanner contains no edges that go "backwards", that is, all edges $(i, j) \in E$ satisfy $i < j$.

Intuitively, this graph is a directed path $(1, 2), (2, 3), \ldots, (n-1, n)$ augmented with additional edges to ensure that each vertex j is reachable from each vertex i by traversing at most k edges in the graph. For example, this is a linear 2-spanner on $V = \{1, \ldots, 5\}$ that uses just 6 out of the $\binom{5}{2} = 10$ possible edges.

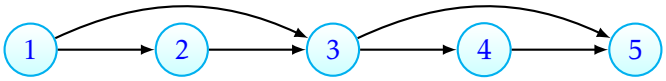

Figure 1: A linear 2-spanner on vertex set $V = \{1, 2, 3, 4, 5\}$ with 6 edges

The only way to construct a linear 1-spanner is to include all edges (i, j) with $i < j$, which includes $\binom{n}{2}$ edges. In this problem we will see that allowing paths of length $k \geq 2$ can substantially reduce the number of edges.

(a) Design an efficient algorithm that, given n, constructs a linear 2-spanner on n vertices with $O(n \log n)$ edges. Prove correctness (i.e., that you indeed output a linear 2-spanner and that it has sufficiently few edges).

(b) Consider the following algorithm for computing the edge set E of a linear 3-spanner.

> **Input:** A number of nodes n
> **Output:** (Allegedly) the edges of a 3-spanner on $V = \{1, \ldots, n\}$
>
> If $n \leq 3$, output $E = \{(i, j) | \ 1 \leq i < j \leq n\}$.
> Else:
>
> - Designate vertices $\lfloor \sqrt{n} \rfloor, \lfloor 2\sqrt{n} \rfloor, \ldots, \lfloor (\sqrt{n} - 1)\sqrt{n} \rfloor$ as **hubs**.
> - For each pair of hubs $h_1 < h_2$, add an edge (h_1, h_2) to E.
> - for each vertex $i \in \{1, ..., n\}$, let $\ell(i)$ be the hub immediately before i and $r(i)$ be the immediately after i. Add $(\ell(i), i)$ and $(i, r(i))$ to E.
> - For each k from 1 to $\sqrt{n}$, recursively construct a linear 3-spanner for segment of the graph between hubs $k-1$ and k (that is, between nodes $(k-1)\sqrt{n}$ and $k\sqrt{n}$) and add the resulting edges to E.
>
> Return E.

(i) Prove by induction that the above algorithm correctly constructs a linear 3-spanner.

(ii) Let $S(n)$ be the number of edges in E output by the algorithm. Give a recurrence for $S(n)$. You may assume, for simplicity, that $\sqrt{n}$ is always an integer.

(iii) Find the size of the linear 3-spanner returned by the algorithm by solving your recurrence.

(c) (Bonus. You are not required to turn in a solution.) Give an algorithm to construct a linear 4-spanner with asymptotically fewer edges than the linear 3-spanner we constructed in the previous problem. That is, if $T(n)$ is the number of edges in your linear 4-spanner on n vertices, then it should be the case that $T(n) = o(S(n))$ (i.e. $\lim_{n\to\infty} T(n)/S(n) = 0$)