Reminder: collaboration is not allowed. Many of these problems have solutions which can be found on the internet – please don’t look. You can of course use the internet (including the links provided on the course webpage) as a learning tool, but don’t go looking for solutions.

Please include proofs with all of your answers, unless stated otherwise.

1 Min-cost bounded path (34 points)

Suppose are are given a directed acyclic graph $G = (V, E)$, a cost function $c : E \to \mathbb{N}$, a length function $\ell : E \to \mathbb{N}$, two nodes $s, t \in V$, and a length bound $L$. Our goal is to compute the minimum cost $s \to t$ path whose length is at most $L$. In other words, find the path $P$ from $s$ to $t$ with $\sum_{e \in P} \ell(e) \leq L$ which minimizes $\sum_{e \in P} c(e)$.

(a) Give a quasipolynomial-time exact algorithm for this problem (i.e., an algorithm which solves the problem but runs in time polynomial in $n, m$ and $\max_{e \in E} c(e), \max_{e \in E} \ell(e)$).

(b) Using your algorithm from the previous part, give an FPTAS.

2 Min-degree Steiner Tree (33 points)

The Min-degree Steiner tree problem is the same as the Steiner tree problem, except instead of minimizing total cost our goal is to minimize the maximum degree. More formally, we are given an undirected graph $G = (V, E)$ and a subset $D \subseteq V$ of terminals. The goal is to find a tree $T$ in $G$ which spans all of $D$ (but not necessarily all of $V$) and which minimizes the maximum degree.

Show how to modify the local search algorithm for min-degree spanning tree to get a local search algorithm for min-degree Steiner tree that runs in polynomial time and returns a tree with maximum degree at most $2\Delta^* + \log n$ (where $\Delta^*$ is the maximum degree of the optimal tree).

Hint: you might need to slightly redefine a $u$-improvement.

3 Scheduling (33 points)

Consider the following scheduling problem on a single machine. We are given $n$ jobs, where each job $j \in [n]$ has a processing time $p_j$, a weight $w_j$, and a deadline $d_j$ (you can assume all of these values are integers). The objective is to schedule the jobs so as to maximize the total weight of the jobs that finish by their deadline.

(a) Prove that there always exists an optimal schedule in which all on-time jobs complete before all late jobs, and the on-time jobs complete in earliest deadline order.

(b) Use part (a) to design an $O(nW)$-time algorithm, where $W = \sum_{j=1}^{n} w_j$.

(c) Use part (b) to design an FPTAS.