CS 860: Modern Topics in Graph Algorithms **University of Waterloo: Winter 2024**

Homework 1

Due: Monday, March 4, 2024

Problem 1. Design an $O(m \log\log n)$ time **deterministic** algorithm for computing a minimum spanning tree of a given weighted undirected graph $G = (V, E)$. **(25 points)**

Hint: Recall that Prim's algorithm can be implemented in $O(m + n \log n)$ time using Fibonacci heaps.

Problem 2. Consider the following modification to the primal-dual algorithm for $(1 - \varepsilon)$-approximation of the bipartite matching problem in Lecture 2:

Algorithm 1. *An algorithm for matching on a bipartite graph $G = (L, R, E)$.*

1. *Let $y_u = 0$ for all $u \in L$, $z_v = 0$ for all $v \in R$, and $M = \emptyset$ initially.*
2. *Let $U = L$ be the set containing all unmatched vertices in L initially*[a].
3. *For $t = 1$ to $(100/\varepsilon^2)$ iterations:*
   - (a) *Create a graph H between U and R by connecting $u \in U$ to any vertex $v \in R$ that belongs to $\arg\min_{w \in N(u)} z_w$ (i.e., the minimum z-value neighbors of u).*
   - (b) *Compute a maximal matching M_H in H greedily (i.e., iterate over edges of H and if both endpoints are unmatched, add it to M_H).*
   - (c) *For any edge $(u, v) \in M_H$:*
     - i. *Remove u from U; if $z_v = 1$ skip this edge, otherwise, let w be the matched neighbor of v in M (which potentially may not even exist).*
     - ii. *Change the matching M such that u is matched to v instead and w is now unmatched; insert w to the set U,*
     - iii. *Update $z_v \leftarrow z_v + \varepsilon$ and $y_u = 1 - z_v$ and $y_w = 0$.*
4. *Output M as the final matching.*

[a] Although some unmatched vertices later will be removed from U, so U may not contain *all* unmatched vertices.

Prove that this algorithm outputs a $(1-\varepsilon)$-approximate maximum matching of any input bipartite graph and analyze its runtime. **(25 points)**

Hint: Notice that this algorithm makes substantial "progress" as long as the number of unmatched vertices is $> \varepsilon$ fraction of maximum matching size – you just need to find the right measure of "progress".

Problem 3. Consider the Palette Sparsification Theorem from Lecture 3. In this problem, we prove a different variant of this theorem.

An undirected graph $G = (V, E)$ is called ***k*-degenerate** if k is the smallest integer such that we can order vertices of V in a way that each vertex has at most k edges to vertices *before* it in this ordering. Prove the following statement for every $k \geqslant 1$, k-degenerate graphs $G = (V, E)$, and a given parameter $\varepsilon > 0$:

Suppose for every vertex $v \in V$, we sample $O(\log(n)/\varepsilon)$ colors $L(v)$ uniformly at random from $[(1+\varepsilon) \cdot k]$; then, with high probability, G is list-colorable from the sampled lists $\{L(v) \mid v \in V\}$.

(25 points)

Problem 4. Let P be a directed **path** of n vertices. Design an algorithm that given any integer $d \geqslant 2$, uses $\widetilde{O}(n/d)$ shortcutting edges H and reduce the diameter of $P \cup H$ to d. Prove that this is nearly optimal, i.e., any shortcutting set of edges that reduces diameter of a path to at most d requires $\Omega(n/d)$ shortcuts.

(25 points)

Problem 5 (**Extra Credit**)**.** In Lecture 2, we saw an algorithm that given a general (not necessarily bipartite) graph $G = (V, E)$ and a matching $M \subseteq E$ in G, computes an **augmenting path** for M in G (if it exists) in $O(mn)$ time.

Design an algorithm for the same problem that runs in $O(m \log n)$ time instead. **(+20 points)**

You receive half the credit even for an algorithm with $O(m + n^2)$ runtime.

Problem 6 (**Extra Credit**)**.** Consider Problem 4 again. What is the minimum diameter you can achieve if you are using only $O(n)$ shortcuts? You do not need to prove your answer is asymptotically minimum as long as it is the right answer (or at least close enough to it). **(+20 points)**

Hint: You can do better, much better, than a direct extension of your solution in Problem 4.