CS 761: Randomized Algorithms **University of Waterloo: Winter 2025**

Homework 3

Due date: Thursday, April 2, 2025

Problem 1. Recall the definition of a **Low Diameter Decomposition (LDD)** from Lecture 15. In this problem, we design another algorithm for constructing an LDD of a given graph $G = (V, E)$ with diameter-bound D:

1. Pick a number d from $\{1, \ldots, D/2\}$ uniformly at random.
2. Pick a permutation π of vertices uniformly at random.
3. Iterate over vertices in the order of π: for any vertex v, find all vertices reachable from v with distance at most d and cluster the *unclustered* ones together.[1]

Prove that the weak diameter of each cluster is at most D deterministically and that for any edge $(u, v) \in E$,

$$\Pr(u \text{ and } v \text{ clustered differently}) = \frac{O(\log n)}{D}.$$

(20 points)

Problem 2. Let $G = (V, E)$ be a *directed* graph. Sample each edge of G independently and with probability

$$p := \frac{100 \cdot n}{\varepsilon^2 \cdot m}.$$

Prove that if we return a **maximum directed** cut in the sampled graph, we obtain a $(1-\varepsilon)$-approximation to the maximum directed cut of the original graph. **(20 points)**

Problem 3. Two rooted trees T_1 and T_2 are considered **isomorphic** to each other iff there exists a bijection ϕ from vertices of T_1 to vertices of T_2 such that if u is a vertex in T_1 with child-vertices $v_1, \ldots, v_k$, then $\phi(u)$ is a vertex in T_2 with child-vertices $\phi(v_1), \ldots, \phi(v_k)$.[2] In less formal terms, two trees are isomorphic if their "structure" looks the same once we ignore labels of vertices.

Design a polynomial time randomized algorithm for determining if two trees are isomorphic or not.

(30 points)

Hint: Use the polynomial method we learned in Lectures 13 and 14 by associating a polynomial to each vertex of the tree which is defined recursively based on polynomials in its sub-tree.

Problem 4. Recall from Lecture 18 that any *bipartite* graph with maximum degree Δ admits a proper edge coloring with Δ colors, which can be found in $O(m \log \Delta)$ time deterministically. In this problem, we see a classical technique for using this algorithm to edge color general (non-bipartite) graphs with *roughly* $\Delta + \sqrt{\Delta}$ colors in near-linear time[3]. Throughout this question, we are going to assume that $\Delta = \omega(\log n)$.

[1] It is possible that v is already clustered before, and so in this step, v itself may not join the cluster centered at it.

[2] We do not assume any ordering among the child-nodes.

[3] Note that this is weaker then the algorithm we (partially) covered in the class, but until recently, this was the best known algorithm for the problem in certain parameter regimes.

1. Suppose we randomly partition the vertices of the graph into two parts by sending each vertex to each part uniformly and independently. Let H be the bipartite subgraph of G obtained by only considering the edges between the two parts. Prove that with high probability,

$$\text{maximum degree of } H \leqslant \frac{\Delta}{2} + O(\sqrt{\Delta \log n}) \quad \text{and} \quad \text{maximum degree of } G \setminus H \leqslant \frac{\Delta}{2} + O(\sqrt{\Delta \log n}).$$

(15 points)

2. Find a way to recursively apply the previous random partitioning technique to obtain a $\Delta + O(\sqrt{\Delta} \cdot \log n)$ coloring of any given general graph in near-linear time with high probability. **(15 points)**

   Hint: Once degrees of vertices become small enough in your recursion, you can run a greedy edge coloring algorithm that colors a graph of max degree Δ with $2\Delta - 1$ colors deterministically and in near-linear time.

Problem 5 (**Extra credit**)**.** We consider a "simple" twist to Problem 2. Suppose we have an *undirected* graph $G = (V, E)$ and we sample each *vertex* of G independently with probability

$$p := 100\varepsilon,$$

for some $\varepsilon > 0$ to obtain a graph H. Prove that with some large constant probability,

$$\text{maximum cut size in } H = (1 \pm \varepsilon) \cdot p^2 \cdot \text{maximum cut size in } G,$$

or provide a counter-example to this statement. **(+10 points)**