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

Homework 1

Due date: Thursday, February 6, 2025

Problem 1. In the (deg +1) vertex coloring problem, we are given an undirected graph $G = (V, E)$ and the goal is to find a coloring of vertices of G such that (i) no edge is monochromatic, and (ii) every vertex $v \in V$ receives a color from the set $\{1, 2, \ldots, \deg(v) + 1\}$ where $\deg(v)$ is the degree of v in G. The difference of this problem with the $(\Delta + 1)$ coloring problem we saw in Lecture 1 is that vertices that have a lower degree here can only receive a color from a smaller range of colors as well (as opposed to all vertices having access to the same $(\Delta + 1)$ colors).

Suppose we are given access to both adjacency list and adjacency matrix of G (and we can read degree of each vertex from its adjacency list in $O(1)$ time). Modify the $(\Delta + 1)$ coloring algorithm of Lecture 1 to solve the (deg +1) coloring problem in $O(n\sqrt{n \log n})$ expected time. **(20 points)**

Problem 2. Prove that in any undirected graph $G = (V, E)$ with minimum cut size $\lambda(G)$, the total number of cuts whose value is at most $\alpha \cdot \lambda(G)$ for a given parameter $\alpha \geqslant 1$ is at most $O(n^{2\alpha})$.

Hint: Recall Karger's algorithm for minimum cut from Lecture 2 and that it can additionally be used to prove the number of minimum cuts is $O(n^2)$ at most. **(20 points)**

Problem 3. In this question, we examine a way of speeding up Karger's minimum cut algorithm, due to Karger and Stein (JACM 1996). Basically, the issue with Karger's algorithm is that it "waits too long" before repeating the algorithm. In other words, we had the basic contraction algorithm that succeeds with probability roughly $1/n^2$, and then we repeat it $O(n^2)$ to boost the probability to a constant.

But, the probability that the basic contraction algorithm fails in its first step is very low—only $2/n$—, which is good for us, while the probability that it fails in its last step is very high—already 1/3—which is undesirable. Karger-Stein algorithm is exploiting this phenomenon cleverly by *interjecting* repetition steps in the middle of basic contraction algorithm. Basically, we are unlikely to destroy the min-cut in the early steps of the contraction algorithm, so why repeat all these steps altogether?

(a) Show that each contraction operation can be implemented in $O(n)$ time. Conclude that the original Karger's algorithm that succeeds with probability at least 2/3 can be implemented in $O(n^4)$ time. **(5 points)**

(b) Karger's basic contraction algorithm runs $n - 2$ contractions consecutively. Instead, consider running only $n - n/\sqrt{2}$ random contraction steps. Prove that the probability that a fixed minimum cut survives this contraction process is at least 1/2. **(2.5 points)**

(c) Consider the following Karger-Stein algorithm: starting from a multi-graph G on n vertices, randomly contract edges until $n/\sqrt{2}$ vertices remain; call the new graph G'. Recursively, run *two* copies of the algorithm *independently* on G' and return the smallest of the two cuts found as the answer.

Define the recurrence $T(n)$ as the worst-case runtime of the above algorithm on n-vertex graphs. Prove:

$$T(n) \leqslant O(n^2) + 2 \cdot T(n/\sqrt{2}).$$

Use this recurrence to bound the runtime of the algorithm with $O(n^2 \cdot \log n)$ time. **(7.5 points)** (*d*) Let $P(n)$ denote the worst-case probability that the above algorithm returns a minimum cut of a given n-vertex graph. Prove:

$$P(n) \geqslant \frac{1-(1-P(n/\sqrt{2}))^2}{2}.$$

Use this recurrence to bound the probability of success of the algorithm with $\Omega(1/\log n)$.

(7.5 points)

Hint: The easiest way (that I know of!) to solve such a recurrence is by induction. But as any other inductive prove, you should first figure out what you are proving.

(*e*) Combine all the above steps to obtain an algorithm that finds a minimum cut in $\tilde{O}(n^2)$ time with high probability. This is a pretty fast algorithm now! (on dense graphs, this is just tiny bit slower than reading the input graph itself). **(2.5 points)**

Problem 4. Recall the power of two choices in balls and bins experiment from Lecture 4. Suppose that each ball, instead of picking two bins, picks $d \geqslant 2$ bins chosen independently and uniformly at random and is placed in the bin with the lowest load (breaking the ties arbitrarily).

Prove that, for any $d \geqslant 2$, if we throw $n/100$ balls this way into n bins, then, the maximum load is

$$O(\frac{\log\log n}{\log d})$$

balls with high probability. **(20 points)**

Problem 5. Let $p \in (0,1)$ be a parameter and G be a random graph on n vertices by picking each possible edge independently and w.p. p. Prove that if

$$p = \frac{100 \ln n}{n},$$

then, with high probability, the minimum cut in G is of $\Theta(\log n)$ size. **(15 points)**

Problem 6 (Extra credit). Let us consider the balls and bins experiment of Problem 4 but with a simple twist. Suppose we partition the bins into d groups of consecutive bins, each of size n/d (assume d divides n for simplicity). Then, each ball picks one bin from each group chosen uniformly at random and joins the bin with the smallest load. In case of the ties, it picks the bin with the smallest ID.

Prove that, for any $d \geqslant 2$, if we throw $n/100$ balls into n bins, then the maximum load is

$$O(\frac{\log\log n}{d})$$

balls with high probability. Note that this bound is considerably stronger than that of Problem 4 for larger values of d. **(+10 points)**

Problem 7 (Extra credit). Let $G = (V, E)$ be an arbitrary undirected graph and $k \geqslant 1$ be an integer. Suppose for every vertex $v \in V$, we sample k edges incident on G independently and uniformly at random with repetition, to obtain a subgraph H of G.

Prove that with high probability, there are

$$O(\frac{n}{k} \cdot \text{polylog}(n)).$$

edges inside G that are between the connected components of H. **(+10 points)**