Massively Parallel Algorithms 02/22, 2019

Problem Set 00

Lecturer: Mohsen Ghaffari

NOTE: This problem set is about the basics that you should know from undergraduate courses on algorithm design and analysis (and particularly randomized algorithms). If you have trouble in solving any of the following problems, it might be the case that you are missing some of the required background material. Please consult with the instructor, in that case.

1 Basic Recursions

Determine the asymptotic answers of the following recursions:

1. $T(n) = 2T(n/3) + O(n)$.
2. $T(n) = 3T(n/2) + O(n)$.
3. $T(n) = 2T(n/2) + O(n)$.
4. $T(n) = 2T(n/2) + O(\sqrt{n})$.
5. $T(n) = 2T(n/2) + O(n/\log n)$.
6. $T(n) = T(\sqrt{n}) + 1$.

2 Estimation

You are holding a bag of balls, where each ball is either red or blue, though you cannot see the content of the bag. Each time, you can take take out a random ball and see its color (and potentially put it back). You want to devise a randomized test which answers whether at least a $1/10$ fraction of the balls is red or not. How many balls should you examine, to have a tester with precision $1 \pm \epsilon$ and certainty at least $1 - \delta$, for given values $\epsilon, \delta > 0$? In particular, if the fraction of red balls is above $(1 + \epsilon)/10$, your answer should be 'yes', with probability at least $1-\delta$. If the fraction of red balls is below $(1-\epsilon)/10$, your answer should be 'no', with probability at least $1 - \delta$. In other cases where the fraction is between these two thresholds, you do not need to provide a guarantee.

3 Selection

Given an array of n items, devise an algorithm that using expected $O(n)$ pairwise comparisons, finds the k^{th} largest element for a given $k \in \{1, \ldots, n\}$.

4 Quicksort

Recall the Quicksort algorithm where each time you take a random element from the list (we call this the pivot), and split the list into two parts of *larger* and *smaller*, by comparing all the other elements with the pivot. What is the probability that the i^{th}-largest element and the j^{th} largest element get compared with each other (ever, throughout the entire run of the algorithm)? Use this probability to prove that the expected number of comparisons in quicksort is at most $2n \ln n$.

5 Finding a Common Friend

Alice and Bob are new to a community of n people. Every day, each of Alice and Bob befriends another random person from the population (uniformly at random, without replacement, and independent of each other). How long does it take, in expectation, until they have a mutual friend? An upper bound that is tight up to a constant factor would be sufficient.

6 Rumor Spreading

Consider a population of n people, where initially one of them knows a rumor. Every day, each person who knows the rumor contacts another random person from the whole population and shares the rumor with him/her. What is the expected time until everyone knows the rumor? An upper bound that is tight up to a constant factor would be sufficient.

7 Independent Set

Consider an n-node graph with average degree $\bar{d}$, i.e., where the graph has $n\bar{d}/2$ edges. An independent set in the graph is a set of vertices no two of which are adjacent.

1. **(Motwani-Raghavan 5.3)** Prove that the graph has an independent set with at least $n/(2\bar{d})$ vertices.

   `Hint: Consider the random process of discarding each vertex (with all of its edges) with probability` $1 - 1/\bar{d}$`. How many vertices remain, and how many edges?`

2. **(Turan's Theorem)** Prove that the graph has an independent set with at least $n/(\bar{d}+1)$ vertices.

   `Hint: Order the vertices randomly and greedily build an independent set. What is the probability for a node` v `of degree` d_v `to be in the independent set?`

8 Balanced Coloring

Consider a ground set B of n elements and m susbets $S_1, \dots, S_m \subseteq B$ of this ground set. Prove that there exists a way to color the elements red or blue such that for each of the given m sets, the number of red and blue elements in this set differ by at most $O(\sqrt{n \log m})$.