Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Institute of Theoretical Computer Science
Mohsen Ghaffari, Angelika Steger, David Steurer, Emo Welzl, Peter Widmayer

Algorithms, Probability, and Computing Exercises KW50 HS18

General rules for solving exercises

- When handing in your solutions, please write your exercise group on the front sheet:

  Group A: Wed 13–15 CAB G 56

  Group C: Wed 16–18 CAB G 52

- This is a theory course, which means: if an exercise does not explicitly say "you do not need to prove your answer", then a formal proof is **always** required.

The following exercises will be discussed in the exercise class on December 12, 2018. Please hand in your solutions not later than December 11.

Exercise 1

In Section 6.3.1, as a part of the algorithm for list ranking, we call each element of the linked list head or tail, using a fair coin toss, and independently of each other. Then, an element is in set I if and only if it holds a head coin and its successor holds a tail coin. We have $\mathbb{E}[|I|] \geq n/8$, because if we break all elements into $n/2$ disjoint pairs of consecutive elements, each first element in a pair has at least $1/4$ probability to be in I. By applying the Chernoff bound, prove that with probability at least $1 - \mathcal{O}(1/n^{10})$, we have $|I| \geq n/16$.

Theorem 1. *(Chernoff bound) Suppose $X_1, \cdots, X_n$ are independent random variables taking values in $\{0, 1\}$ and let X denote their sum, then for any $0 \leq \delta \leq 1$*

$$\Pr[X \leq (1-\delta)\mathbb{E}[X]] \leq e^{-\frac{\delta^2 \mathbb{E}[X]}{2}}$$

$$\Pr[X \geq (1+\delta)\mathbb{E}[X]] \leq e^{-\frac{\delta^2 \mathbb{E}[X]}{3}}.$$

Exercise 2

(Exercise 6.3 from the lecture notes)

In Section 6.3.2., we assume that the tree input is provided as the adjacency lists of its nodes. Suppose that instead of adjacency lists, the graph is input as an $n \times n$ binary adjacency matrix where the entry at location (i, j) is 1 if the i^{th} and the j^{th} nodes are adjacent, and 0 otherwise. Devise an algorithm with $O(\log n)$ depth and $O(n^2)$ work that transforms this adjacency matrix to adjacency linked lists, one for each node.

Exercise 3

(Exercise 6.5 from the lecture notes)

Modify the approach for computing a pre-order of nodes of a given tree $T = (V, E)$ from the lecture notes so that it provides a post-order numbering $\text{post} : V \to \{0, \ldots, n-1\}$ of the nodes. That is, for each node v, we have a post-order numbering of the subtree rooted in the first child of v, then a post-order numbering of the subtree rooted in the second child of v, etc, followed by node v itself. In particular, you should have $\text{post}(r) = n - 1$. Argue that the algorithm provides the correct ordering, and explain why it has $O(\log n)$ depth and $O(n)$ computation.

Exercise 4

(Exercise 6.6 from the lecture notes)

Devise a parallel algorithm with $O(\log n)$ depth and $O(n)$ total computation that computes for each node v in a tree $T = (V, E)$ with root $r \in V$, the number of its descendants, i.e., the total number of nodes in the subtree rooted at node v.

Exercise 5

(Exercise 6.7. from the lecture notes)

Suppose that each leaf node u in a tree $T = (V, E)$ with root $r \in V$ is given a number $b(u)$. Devise a parallel algorithm with $O(\log n)$ depth and $O(n)$ total computation that computes for each node v the summations of the $b(u)$ values over all the leaves u that are descendants of v.