ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Institute of Theoretical Computer Science
Bernd Gärtner, Mohsen Ghaffari, Rasmus Kyng, Angelika Steger, David Steurer

Algorithms, Probability, and Computing **Exercises KW50** **HS21**

General rules for solving exercises

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

  Group A: Wed 14–16

  Group B: Wed 14–16

  Group C: Wed 16–18

  Group D: Wed 16–18

- 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 classes on December 15, 2021. You can hand in your solutions during the lecture break on December 13 or December 14 or via Moodle, no later than 4 pm at December 14.

Exercise 1

You throw a fair coin n times in a row, all throws being independent. Let's say you win if the coin comes up heads and you lose if the coin comes up tails. Show that the expected length of the longest winning streak is $\Theta(\log n)$.

Exercise 2

(Exercise 6.1 from the lecture notes)

Show that the maximum of n entries can be computed in $O(\log \log n)$ time-steps, using the CRCW version of PRAM with n processors.

Exercise 3

(Exercise 6.2 from the lecture notes)

Use Brent's principle to determine the smallest number of processors that would allow us to run the Parallel Prefix algorithm which we saw above in $O(\log n)$ time. Recall that algorithm had $O(\log n)$ depth and $O(n)$ total computation. Explain how the algorithm with this small number of processors works, that is, what each processor needs to do in each time step.

Exercise 4

(Exercise 6.3 from the lecture notes)

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 vertex.

Exercise 5

(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 $post : V \to \{0, \ldots, n-1\}$ 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 $post(r) = n - 1$. Argue that the algorithm provides the correct ordering, and explain why it has $O(\log n)$ depth and $O(n)$ computation.

Exercise 6

(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.