1 List Ranking

Prove the correctness of Wyllie’s pointer hopping algorithm for list ranking.

2 Recurrences

Prove that the following recurrence solves to $O(\log \log N)$:

$$T(n) = \begin{cases} 
T\left(\frac{n}{2}\right) + O(1) & \text{if } n \geq \frac{N}{\log N} \\
O(1) & \text{otherwise}
\end{cases}$$

3 Deterministic list ranking

Consider the following algorithm for list ranking on a linked list $A$ of size $N$:

1: Run a single round of deterministic coin tossing to pick an independent set $I$.
2: for each node $v \in I$ in parallel do
3: Follow edges from $v$ until the first $v' \in I$ counting the number of visited edges $w_v$.
4: Set the weight of $v$ to $w_v$.
5: Create a shortcut link from $v$ to $v'$.
6: end for
7: Run Wyllie’s pointer hopping algorithm on the linked list consisting of $I$ (with new weights) and newly created shortcut links.
8: for each node $v \in I$ in parallel do
9: Run sequential list ranking algorithm backward on original edges starting from $v$ (and its newly computed rank as weight) until the first $v'' \in I$.
10: end for

(a) Prove that the algorithm correctly computes the ranks in the linked list.

(b) Analyze the work and runtime of the algorithm. Explain why you cannot achieve $O(\log N)$ time for $p = N/\log N$ processors.
4 In-order traversal of a tree

Given a tree $T = (V, E)$ rooted at node $r$, in-order traversal of the tree consists of in-order traversal of the left subtree of $r$, followed by $r$, and followed by in-order traversal of the right subtree of $r$. Design a parallel algorithm that computes the index of each node in the in-order traversal of $T$. 