Weekplan: Distributed Data Structures

Philip Bille Inge Li Gørtz Eva Rotenberg

References and Reading

[1] Nearest Common Ancestors: A Survey and a New Algorithm for a Distributed Environment, S. Alstrup, C. Gavoille, H. Kaplan, T. Rauhe, TOCS 2004.

We recommend reading [1] in detail.

1 Labeling Schemes for Trees Let T be a rooted tree with n nodes. Consider the following queries:

- sibling(v,w): determine if v is a sibling of w.
- adjacency(v,w): determine if there is an edge (v, w).
- ancestor(v,w): determine if v is an ancestor of w.

Solve the following exercises.

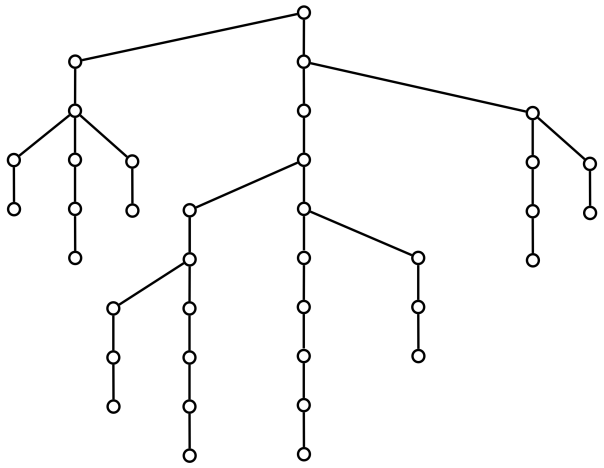
1.1 [w] Give an efficient labeling scheme for sibling queries.

1.2 [w] Give an efficient labeling scheme for adjacency queries.

1.3 Give an efficient labeling scheme for ancestor queries using labels of length $o(n)$ bits.

2 Label Length Encoding Consider the parent labeling scheme using $2\lceil \log n \rceil$ bit labels. The query algorithm assumes that we know the value $\lceil \log n \rceil$ to correctly extract the IDs. What if we do not know it? Show how to add $o(\log n)$ extra bits to the labels that will allow us to decode the label without knowning $\lceil \log n \rceil$.

3 Nearest Common Ancestor Labeling Schemes Consider the following tree T.



Solve the following exercises.

3.1 Show a heavy path decomposition of T.

3.2 Construct the labels of at least 4 non-trivial nodes in T using the $O(\log^2 n)$ bit labeling scheme.

3.3 Confirm that the scheme correctly computes nearest common ancestors for pairs of the constructed labels.

4 Query Algorithm Performance Consider the query algorithms for the various labeling schemes.

4.1 [w] Give a simple query algorithm for the parent labeling scheme.

4.2 Give a simple query algorithm for the ID encoding nca labeling scheme.

4.3 Give a simple query algorithm for the heavy path decomposition nca labeling scheme.

5 Path Decompositions Let T be a tree with n nodes. Consider the following path decompositions.

- The *leaf-heavy decomposition* picks a *leaf heavy child* with maximum number of descendant leaves at each node and classify that as a *leaf heavy node*. The remaining nodes are *leaf light nodes*. The *leaf-lightdepth* of T is the maximum number of edges to leaf light nodes on a root-to-leaf path in T.
- The *long-path decomposition* picks a *long child* of maximum height at each node and classify that as a *long node*. The remaining nodes are *short nodes*. The *shortdepth* of T is the maximum number of edges to short nodes on a root-to-leaf path in T.

Solve the following exercises.

5.1 Show that the leaf-lightdepth is at most $O(\log \ell)$, where ℓ is the total number of leaves in the tree.

5.2 [$*$] What bounds can you give on the shortdepth of a tree?

6 Variable-Length Encodings Suppose a label stores the concatenation of a sequence $v_1, \ldots, v_k$ of variable length codes of total length ℓ. Show how to add $O(\ell)$ information to the label that will allow us decode the sequence.

7 Alphabetic Codes Let T be a tree with n nodes and let $h_1, \ldots, h_k$ be the heavy paths from the root of T to a node v. Consider the topmost nodes $v_1, \ldots, v_k$ on the heavy paths. Solve the following exercises.

7.1 [w] Argue that $k = O(\log n)$ and $n = \text{size}(v_1) > \text{size}(v_2) > \cdots > \text{size}(v_k) > 0$.

7.2 Suppose we that for each node v_i, $1 < i \leq k$ store a code b_i of length at most

$$|b_i| \leq \log(\text{size}(v_{i-1}) - \log(\text{size}(v_i)) + O(1)$$

Show that $\sum_{i=2}^{k} |b_i| = O(\log n)$.

8 Lexicographic Comparison Let x and y be bitstrings stored in the rightmost (least significant) bits of two memory words. Given their lengths $|x|$ and $|y|$ show how to compare x and y lexicographically in constant time.

9 Range Minimum Queries Let A be an array A of n integers. Show how to preprocess A in $O(n)$ space to support the following *range minimum query* in constant time:

- RMQ(i, j): Return the minimum element among $A[i], A[i+1], \ldots, A[j]$.

Hint: Find a connection to nearest common ancestors.