Weekplan: Level Ancestor

Philip Bille

References and Reading

[1] The Level Ancestor Problem Simplified, M. A. Bender, M. Farach-Colton, Theoret. Comp. Sci., 2003.

[2] Scribe notes from MIT.

[3] Finding level-ancestors in dynamic trees, P. F. Dietz, WADS 1991.

[4] Finding level-ancestors in trees, O. Berkman, U. Vishkin, J. Comput. System Sci., 1994

We recommend reading [1] and [2] in detail.

Exercises

1 Ancestor Data Structures Let T be a rooted tree with n nodes. We are interested in a data structure supporting the following operation on T.

- ancestor(v, w): return yes if v is an ancestor of w and no otherwise.

Give a simple and compact data structure that supports fast ancestor queries (without using a level ancestor data structure).

2 Long Path Decomposition Bounds Prove tight bounds for the number of long paths in a root-to-leaf path.

2.1 Find a tree with n nodes such that the maximum number of long paths on a root-to-leaf path is $\Omega(\sqrt{n})$.

2.2 [$*$] Show that any tree with n nodes has $O(\sqrt{n})$ long paths on a root-to-leaf path.

3 Level Ancestor on Shallow Binary Trees Let T be a rooted, binary tree with n nodes of height $O(\log n)$. Give a simple and compact data structure that supports fast level ancestor queries (without using a level ancestor data structure). *Hint:* A path in T can be encoded in a single word of memory.

4 Ladders Let T be a tree of height h with n nodes. Solve the following exercises.

4.1 Show that any root-to-leaf path can be covered by at most $O(\log h) = O(\log n)$ ladders.

4.2 Ladders are obtained by *doubling* the long paths. Suppose we instead extend long paths by a factor $k > 2$. What is the effect?

5 Few Leafs Suppose that your input tree has no more than $n/\log n$ leaves. Suggest a (slightly) simplified solution to the level ancestor problem with linear space and constant query time.

6 Heavy Paths Let T be a tree with n nodes. Define size(v) to be the number of descendant of v. Consider the following decomposition rule.

- First find a root-to-leaf path as follows. Start at the root. At each node continue to a child of maximum size, until we reach a leaf. Remove the resulting path and recursively apply the rule to the remaining subtrees.

The resulting paths are called the *heavy paths* and the edges not on a heavy path are *light* edges. Solve the following exercises.

6.1 [w] Draw a not to small example of the heavy path in a tree.

6.2 Give an upper bound on the number of heavy paths on any root-to-leaf path in T.

7 Weighted Level Ancestor Let T be tree with n nodes. Each edge is assigned a weight from $\{0, \ldots, u-1\}$, and the weight of a node v is the sum of the weight of the edges on the path from the root to v. We want a data structure that supports the following operation on T. Given a leaf ℓ and an integer x define

- WLA(ℓ, x): return the deepest ancestor of ℓ of weight $\leq x$.

7.1 [w] Give a simple data structure that supports WLA queries in $O(n^2)$ space and $O(\log \log u)$ time.

7.2 Give a data structure that supports WLA queries in $O(n)$ space and $O(\log n)$ time.

7.3 Consider the predecessor problem on n elements from a universe of size u. Any solution that uses $O(n)$ space requires at least $\Omega(\log \log u)$ query time. Can we hope to solve the weighted level ancestor problem in $O(n)$ space and $O(1)$ time?

7.4 [$*$] Give a data structure that supports WLA queries $O(n)$ space and $O(\log \log u)$ time. *Hint:* Use heavy path decomposition.