Weekplan: Lowest Common Ancestors and Range Minimum Queries

Inge Li Gørtz

References and Reading

[1] The LCA problem revisited, M. A. Bender, M. Farach-Colton, Latin American Symposium 2000.

[2] Scribe notes from MIT

[3] Fast Algorithms for Finding Nearest Common Ancestors, D. Harel and R. E. Tarjan, SIAM J. Comput., 13(2), 338–355.

We recommend reading [1] and [2] in detail before the lecture. [3] provides background on LCA. In the pretest (see DTU Learn) you can check if you understand the basics before the lecture. The pretest is an absolute minimum for what you should know before the lecture.

Exercises

1 [w] **RMQ** Consider the array $A = [3,4,5,4,5,4,5,4,3,2,1,0,1,0,1,2,3,4,3,4,3,2,1,2,3,2,3,4,5,6,7,6]$.

1.1 Give the arrays A' and B used for the sparse table in the two level ±1RMQ data structure.

1.2 Write down the normalized blocks.

1.3 Write down the tabulation solution (2-dimensional array) for first and second last of the normalized blocks.

1.4 Show how the query RMQ(3, 29), RMQ(2,5), and RMQ(7,8) is performed (you do not have to construct the sparse array data structure for A', but explain which intervals that are used in the queries).

2 Size of blocks In the RMQ data structure we divided the array into blocks of length $\frac{1}{2}\log n$. What happens if we instead use a block size of

- $\log n$
- $\frac{3}{4}\log n$

3 Range X Queries We saw how to support range minimum queries on an array A of n elements in linear space and constant time. Try to support the following similar queries on A:

- Range Maximum Queries
- Range Sum Queries
- Range Median Queries

Try to explain when we can use the techniques from the RMQ data structure on a problem.

4 Reduction between RMQ and LCA In the lecture we saw how to reduce RMQ to LCA via a Cartesian tree and from LCA to RMQ.

4.1 Build the Cartesian tree T for the array $A = [3,5,1,3,8,6,9,2,42,4,7,12]$.

4.2 Reduce LCA on T to ±1RMQ. That is, construct the array for the ±1RMQ instance.

5 Distance Queries in Trees Let T be a unrooted tree in which each edge has an integer weight. The distance between two nodes u and v is the sum of edge weights on the path between u and v. Give a linear-space data structure for T that can report the distance between any pair of nodes in constant time.

6 First Covering Ancestor Let T be a rooted tree with n nodes. We say that T is *labeled* if each leaf is associated with a *label* from a set of labels $L = \{0, \ldots, l-1\}$. Given a node $v \in T$, the subtree rooted at v, denoted $T(v)$, is the tree consisting of v and all descendants of v. A node $v \in T$ *covers a label* ℓ if $T(v)$ contains a leaf labeled ℓ. Given a leaf $v \in T$ and a label $\ell \in L$, a *first covering ancestor query* is defined as follows.

- FCA(v, ℓ): return the deepest ancestor a of v such that a covers ℓ.

Given a labeled tree T, the *first covering ancestor problem* is to preprocess T into a compact data structure that supports first covering ancestor queries.

Give a linear-space data structure for the first covering ancestor problem that supports fast FCA queries.

7 Cartesian Trees Give an efficient algorithm for constructing the Cartesian tree of an array with n elements.