# 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.

[4] Competitive Programmer's Handbook, section 9.3, Antti Laaksonen.


We recommend reading [1], [2] and [4] in detail before the lecture. [3] provides background on LCA.

## Exercises

**1 Sparse table** Show that we can find the results for all power-of-two intervals in $O(n \log n)$ time.

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

**2.1** Give the arrays $A'$ and $B$ used for the sparse table in the two level ±1RMQ data structure. Use block size 3.

**2.2** Construct the sparse table solution for $A'$.

**2.3** How many different tabulation tables do we need to store (how many different describing sequences/normalized blocks are there)?

**3 Size of blocks** In the ±1RMQ 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

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

**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** [$w$] **Segment tree** Construct the RMQ segment tree for the array $A = [4,2,7,3,5,1,2,8,9,8,4,5,3,6,9,3]$.

**7 Range Updates** In the range update problem we want to preprocess an array $A$ to efficiently support the following operations:

- ADD($i$,$j$, $k$): Add $k$ to each of the entries $A[i]\dots A[j]$.
- Lookup($i$): Return the value $A[i]$.

Give an efficient solution to solve the range update problem. *Hint:* Consider the difference array containing the differences between adjacent positions in $A$.

**8 Range Smallest and Range Uniqueness** Let $A$ be an array of length $n$. Consider the following queries:

- RS($i, j, t$): return all integers $\leq t$ in $A[i, j]$.
- RU($i, j$): return the unique integers in $A[i, j]$.

Solve the following exercises.

**8.1** [$w$] Draw the array $A[1, 12] = [4, 1, 3, 2, 1, 4, 4, 3, 3, 1, 2, 5]$. Show the result of RS(5, 11, 3), and RU(5, 11).

**8.2** Give a compact data structure that supports *RS* queries. Your query time should be output-sensitive.

**8.3** Define the *predecessor array* $P$ of $A$ as the array $P$ such that $P[i] = \max\{1 \leq j < i, A[j] = A[i]\} \cup \{0\}$. Draw the predecessor array $P$ of example array from exercise **8.1**.

**8.4** [$*$] Give a compact data structure that supports RU queries $A$. Your query time should be output-sensitive. *Hint:* find a way to use the predecessor array.