Range Minimum Queries and Lowest Common Ancestor

Inge Li Gørtz

Range Minimum Queries and Lowest Common Ancestor

- Range Minimum Queries (RMQ) and Lowest Common Ancestor (LCA)
- RMQ
  - Simple solutions
  - Better solution
  - 2-level solution
- Reduction between RMQ and LCA

Range Minimum Queries

- Range minimum query problem. Preprocess array A[1…n] of integers to support
  - RMQ(i,j): return the (entry of) minimum element in A[i…j].

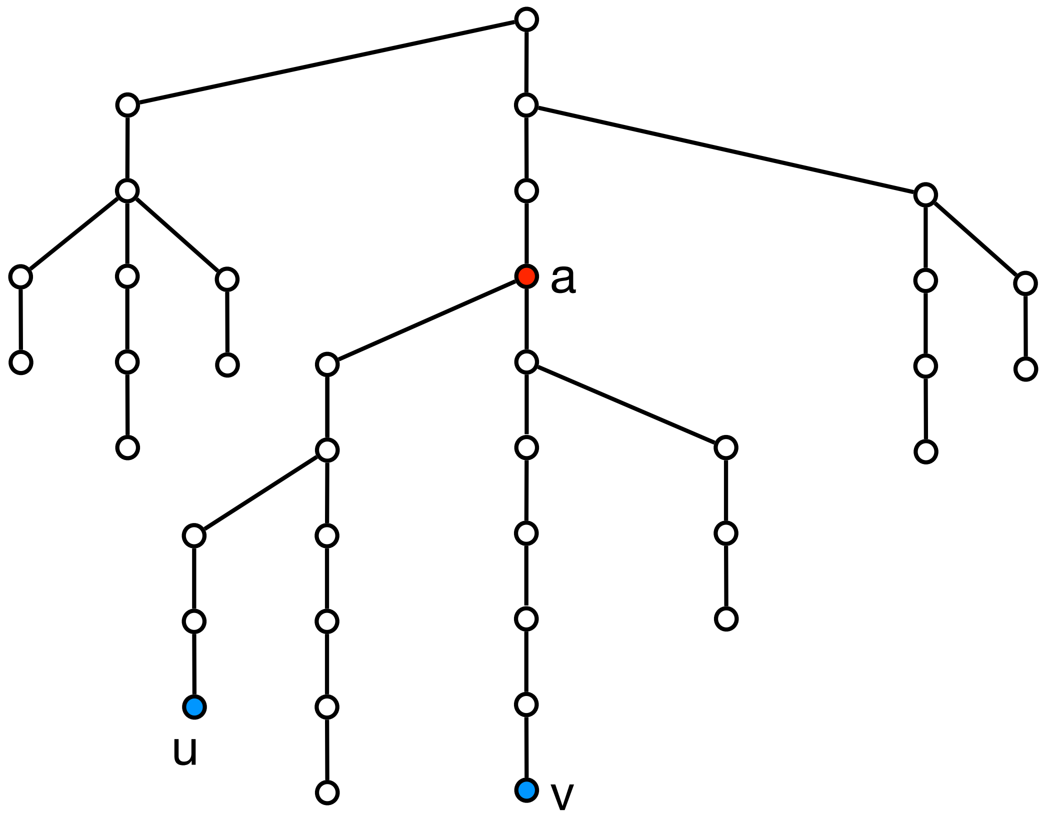
1	2	3	4	5	6	7	8	9	10
1	7	12	8	2	5	1	4	8	3

  - RMQ(3,6) = 2 (index 5)

- Basic (extreme) solutions
  - Linear search:
    - Space: O(n). Only keep array (no extra space)
    - Time: O(j-i) = O(n)
  - Save all possible answers: Precompute and save all answers in a table.
    - Space: $O(n^2)$ pairs => $O(n^2)$ space
    - Time: O(1)

Lowest Common Ancestor

- Lowest common ancestor problem. Preprocess rooted tree T with n nodes to support
  - LCA(u,v): return the lowest common ancestor of u and v.



LCA(u,v) = a

Lowest Common Ancestor

- Basic (extreme) solutions
  - Linear search: Follow paths to root and mark when you visit a node.
    - Space: O(n). Only keep tree (no extra space)
    - Time: O(depth of tree) = O(n)
  - Save all possible answers: Precompute and save all answers in a table.
    - Space: $O(n^2)$ pairs => $O(n^2)$ space
    - Time: O(1)

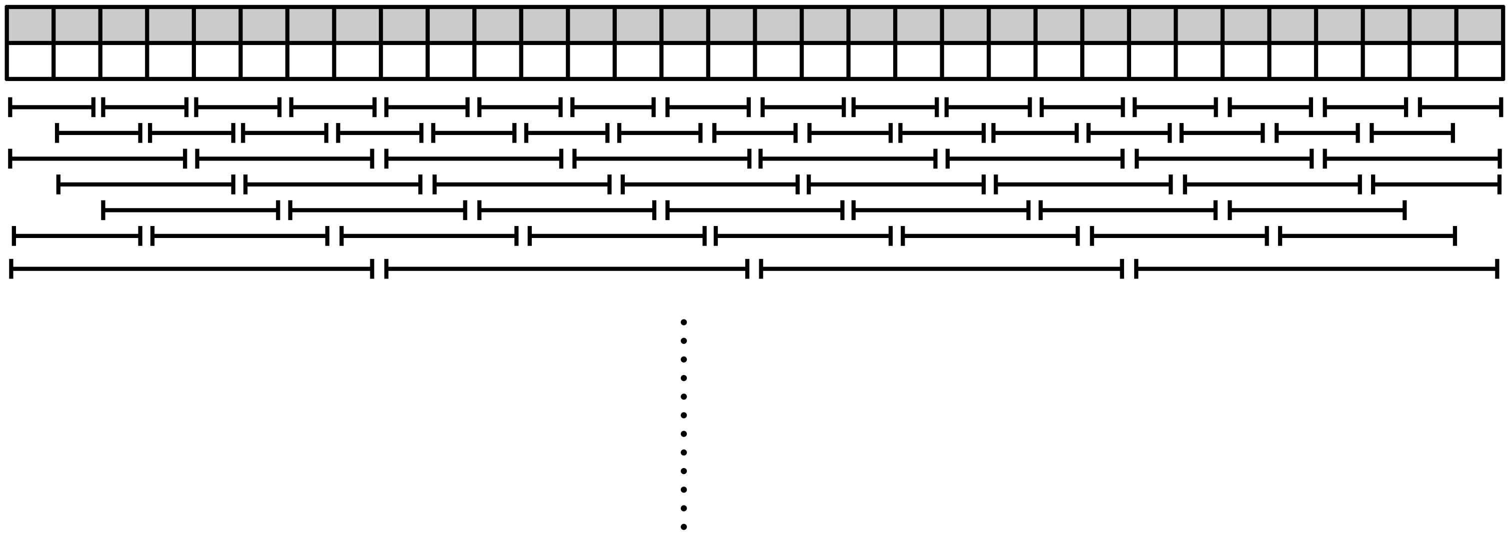
RMQ and LCA

- Outline.
  - Can solve both RMQ and LCA in linear space and constant time.
    - First solution to RMQ
    - Solution to a special case of RMQ.
    - See that RMQ and LCA are equivalent (can reduce one to the other both ways).

RMQ

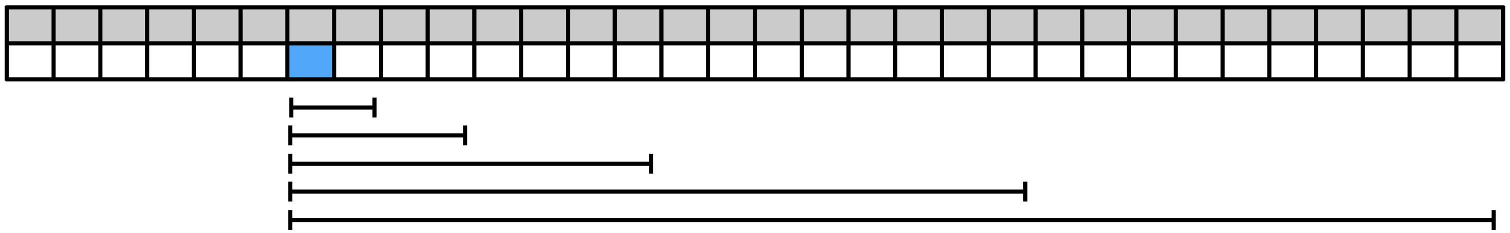
RMQ: Sparse table solution

- Save the result for all intervals of length a power of 2.



RMQ: Sparse table solution

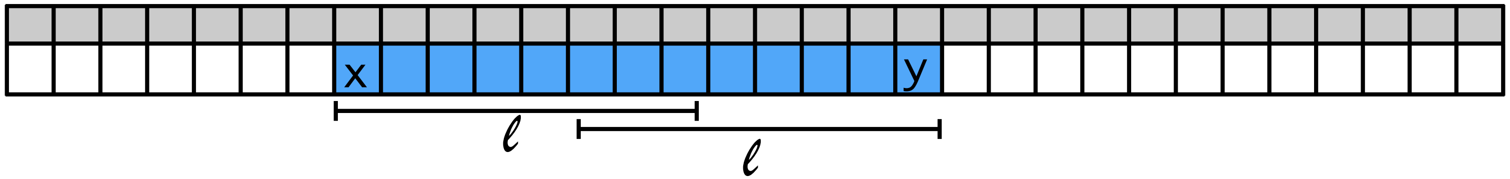
- For all positions we have all power of 2 length intervals starting at that position.



- Space: O(n log n)

RMQ: Sparse table solution

- Query:



  - Any interval the union of two power of 2 intervals.
  - Query the two intervals and take minimum
- Time: O(1)

RMQ: Linear space

- Consider ±1RMQ: consecutive entries differ by at most 1.

1	2	3	4	5	6	7	8	9	10	11	12	13
4	5	6	5	4	3	2	3	2	3	4	5	4

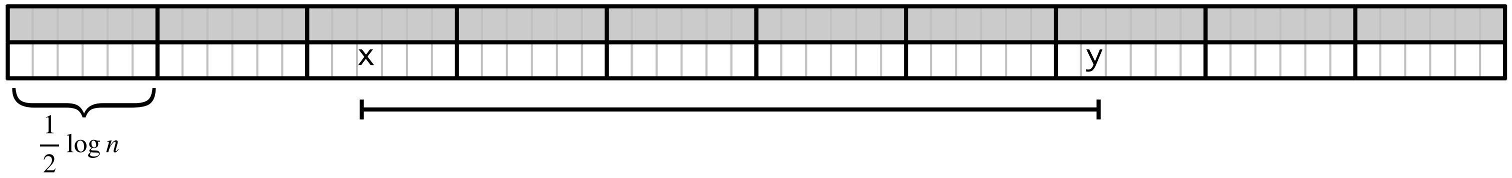
- 2-level solution: Combine
    - $O(n \log n)$ space, $O(1)$ time
    - $O(n^2)$ space, $O(1)$ time.

$$\Downarrow$$

- $O(n)$ space, $O(1)$ time.

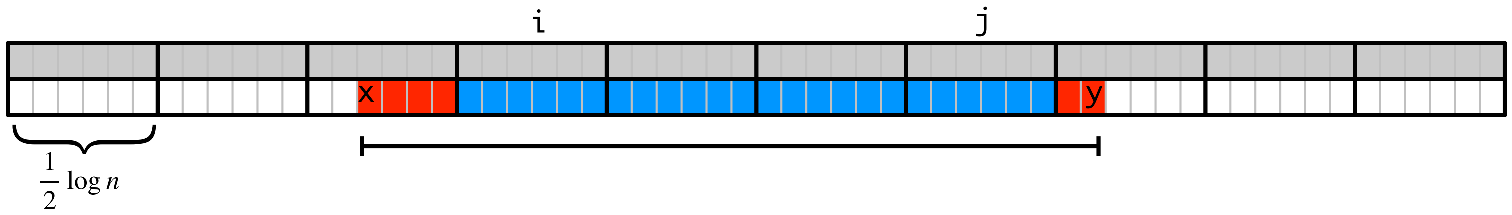
±1RMQ

- Divide A into blocks of size $\frac{1}{2}\log n$



±1RMQ

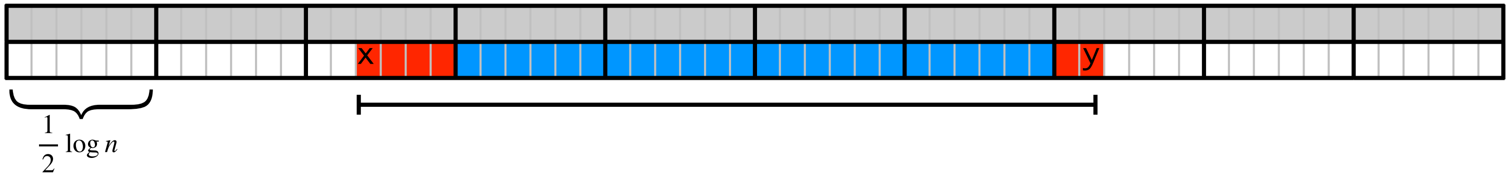
- Divide A into blocks of size $\frac{1}{2}\log n$



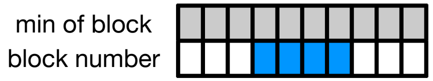
- 2-level data structure:
  - Sparse table on blocks
  - Tabulation inside blocks.

- RMQ(x,y) = min{ RMQ on blocks i to j,
  RMQ inside block i-1,
  RMQ inside block j+1 }.

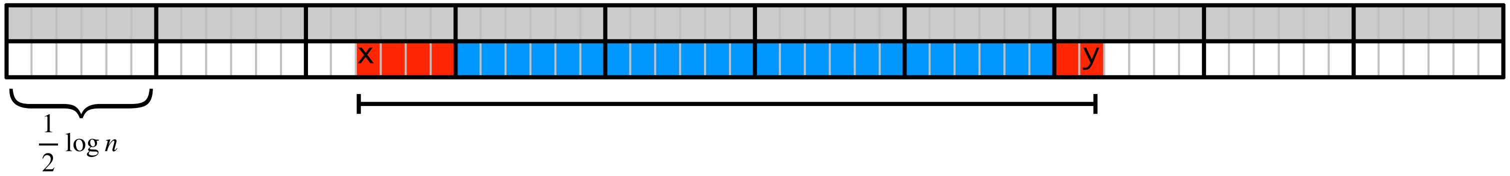
±1RMQ: Data structure on blocks

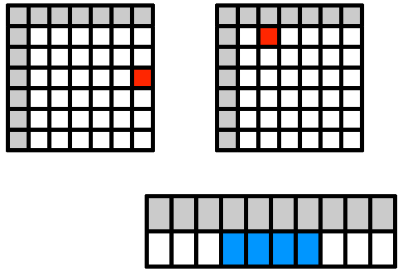


- Two new arrays.
    - Array A’: minimum from each block
    - B: position in A where A’[i] occurs.
- Sparse table data structure on A’.
- Space: O(|A’| log |A’|) = O(n).
- Time: O(1)



±1RMQ: Data structure inside blocks





- Precompute and save all answers for each block.
- Gives solution using
  - Space: O(n) + space for precomputed tables.
  - Time: O(1) + O(1) + O(1) = O(1).

2 table lookups

sparse table

min{•,•,•}

±1RMQ: Storing the tables

- Naively: $\log^2 n$ for each table => n log n space. 😕

- Observation: If X[i] = Y[i] + c then all RMQ answers are the same for X and Y.
    - X = [7, 6, 5, 6, 5, 4]
    - Y = [3, 2, 1, 2, 1, 0]

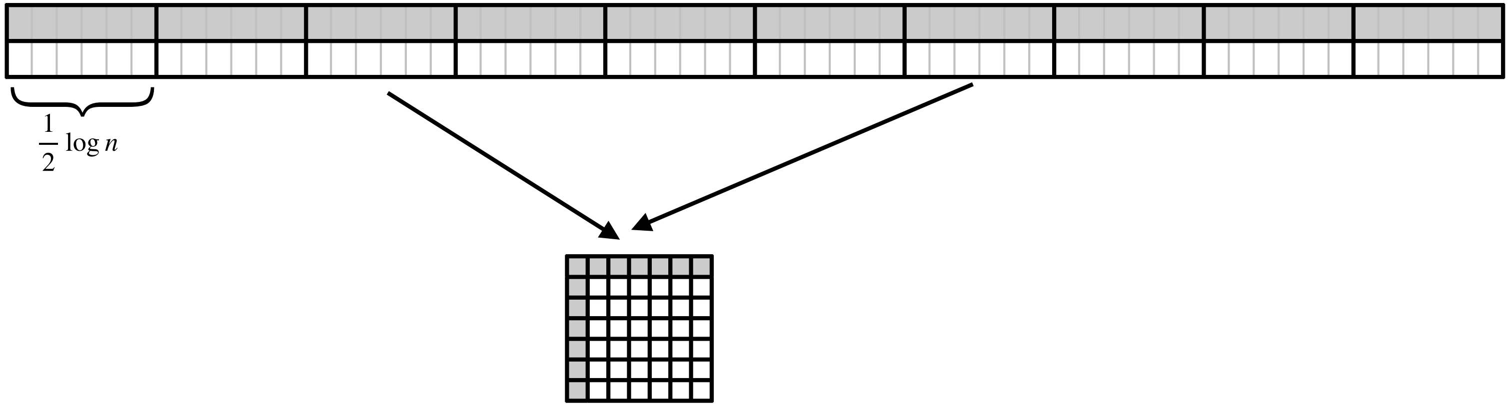
- Normalize blocks:
    - X = [0, -1, -2, -1, -2, -3] = Y

- Normalized block described by sequence of +1s and -1s:
    - X = Y = -1, -1, +1, -1, -1.

- How many different normalized blocks are there?
    - length of sequence = $\frac{1}{2}\log n - 1$
    - #sequences = $2^{\frac{1}{2}\log n - 1} \leq \sqrt{n}$.

±1RMQ: Data structure inside blocks

- Precompute and save all answers for each normalized block.
- Size of a table: O($\log^2$ n)
- For each block save which precomputed table it uses.



- Space: $O(\sqrt{n} \cdot \log^2 n) + O(n/\log n) = O(n)$
- Plugging into 2-level solution:
  - Space: O(n) + space for precomputed tables = O(n).

LCA and RMQ

RMQ and LCA

- We will show
  - RMQ $\xrightarrow[\text{reduces to}]{}$ LCA $\xrightarrow[\text{reduces to}]{}$ ±1RMQ

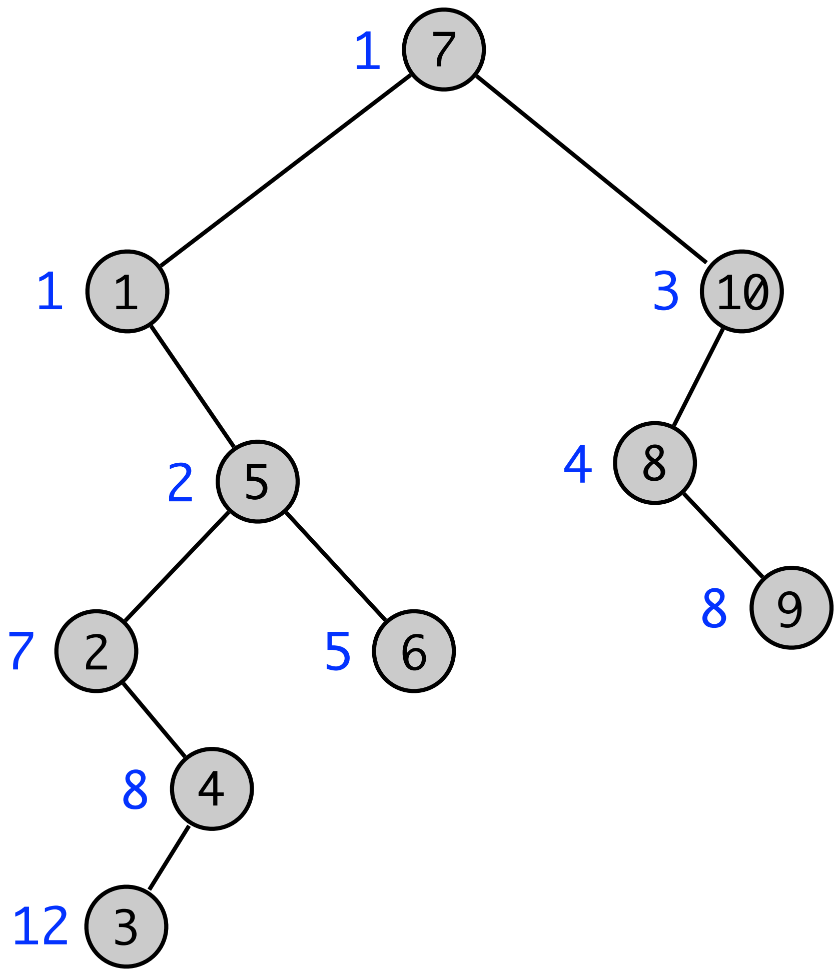
If there is a solution to LCA using s(n) space and t(n) time, then there is a solution to RMQ using O(s(n)) space and O(t(n)) time.

If there is a solution to ±1RMQ using s(n) space and t(n) time, then there is a solution to LCA using O(s(n)) space and O(t(n)) time.

RMQ to LCA

1	2	3	4	5	6	7	8	9	10
1	7	12	8	2	5	1	4	8	3

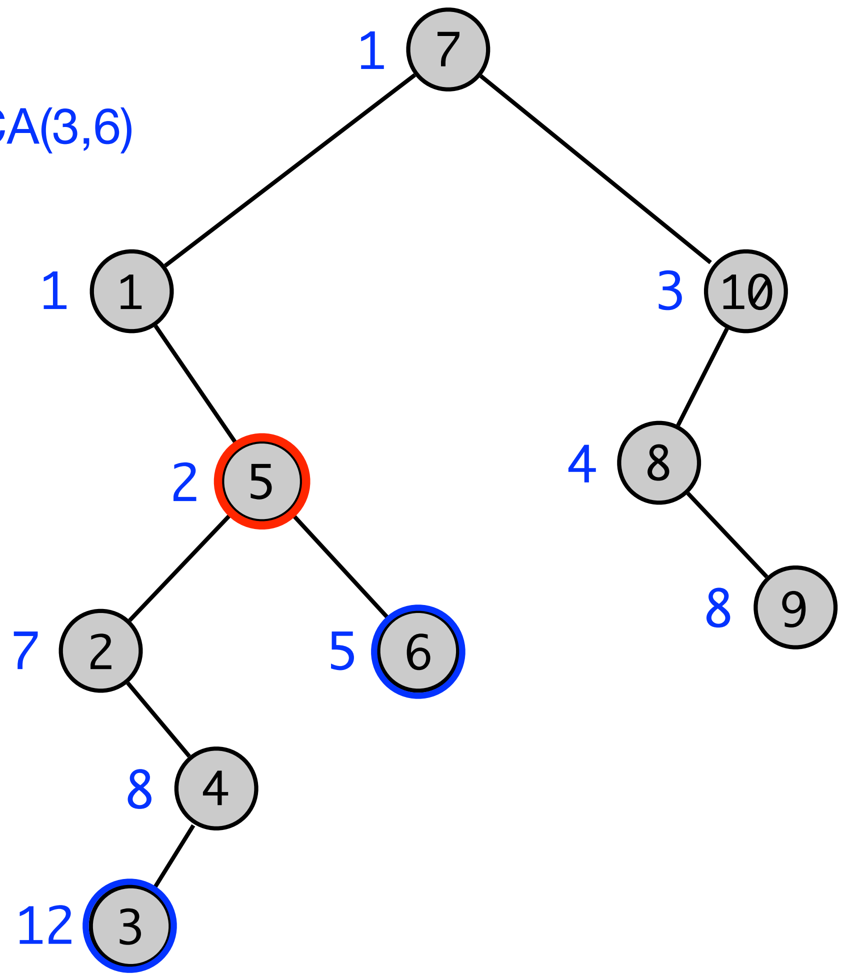
- Cartesian tree.



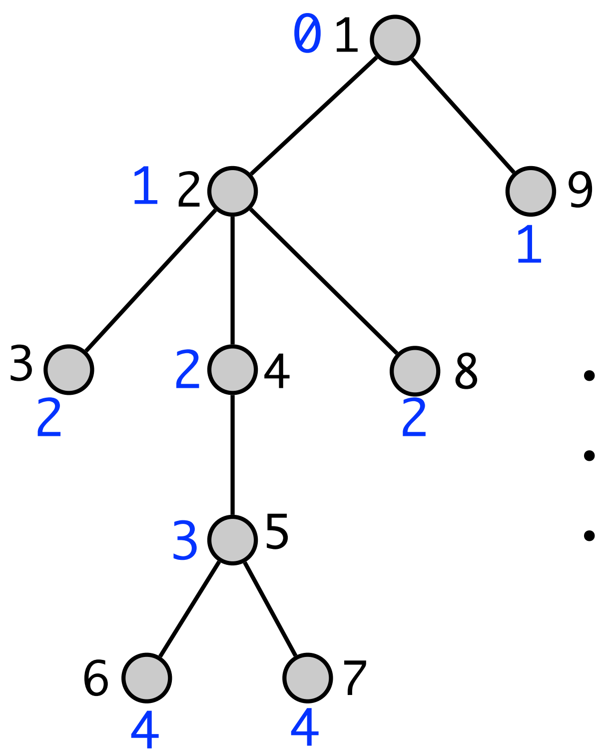
RMQ to LCA

1	2	3	4	5	6	7	8	9	10
1	7	12	8	2	5	1	4	8	3

- Cartesian tree.
- RMQ(3,6) = LCA(3,6)



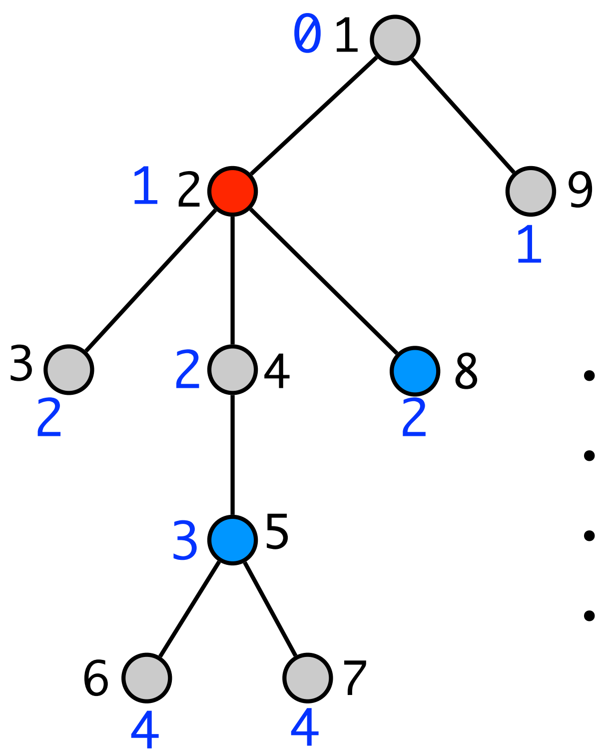
LCA to ±1RMQ



- E = [1, 2, 3, 2, 4, 5, 6, 5, 7, 5, 4, 2, 8, 2, 1, 9, 1]
- A = [0, 1, 2, 1, 2, 3, 4, 3, 4, 3, 2, 1, 2, 1, 0, 1, 0]
- R = [1, 2, 3, 5, 6, 7, 9, 13, 16]

- E: Euler tour representation: preorder walk, write node preorder number of node when met.
- A: depth of node node in E[i].
- R: first occurrence in E of node with preorder number i
- LCA(i, j) = E[RMQ(R[i], R[j])].

LCA to ±1RMQ



- E = [1, 2, 3, 2, 4, 5, 6, 5, 7, 5, 4, 2, 8, 2, 1, 9, 1]
- A = [0, 1, 2, 1, 2, 3, 4, 3, 4, 3, 2, 1, 2, 1, 0, 1, 0]
- R = [1, 2, 3, 5, 6, 7, 9, 13, 16]
- LCA(5,8) = RMQ(6, 13).

- E: Euler tour representation: preorder walk, write node preorder number of node when met.
- A: depth of node node in E[i].
- R: first occurrence in E of node with preorder number i
- LCA(i, j) = E[RMQ(R[i], R[j])].

RMQ and LCA

- Theorem. RMQ and LCA can be solved in $O(n)$ space and $O(1)$ query time.