- Given an array A comprising n pairwise distinct integers, preprocess A and build data structures so that to efficiently answer queries of the following form: given two indices i, j of A with $i \leq j$, find an integer k with $i \leq k \leq j$, for which $A[k]$ is minimum among $A[i], A[i+1], \ldots, A[j]$.

- A naive algorithm is to do no preprocessing but to answer any query with i, j entries by traversing $A[i], A[i+1], \ldots, A[j]$. But this leads to taking a linear time to answer any query. This is not an efficient solution since the number of queries could be quite large.

  Here is another obvious algorithm: During preprocessing, for every i, j, find the minimum among $A[i], A[i+1], \ldots, A[j]$ by traversing that portion of A and store the index at which this minimum occurs in the ij^{th}-entry of an array $MinIdx$. For this algorithm, the preprocessing time is $O(n^3)$ and the space of preprocessed data structures is $O(n^2)$. The query algorithm needs to simply do the lookup in $MinIdx$, and hence the query time is $O(1)$. The time-space tradeoffs between these two algorithms is immediate.

  With the following code, the preprocessing time of the above algorithm can be reduced to $O(n^2)$:

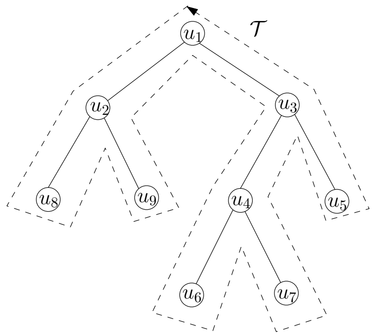
  for $i := 1$ to $n-1$
      if $A[i] < A[i+1]$ then $MinIdx(i, i+1) = i$
      else $MinIdx(i, i+1) = i+1$

  for $i := 1$ to $n-2$
      for $j := i+2$ to n
          if $A[MinIdx[i, j-1]] < A[j]$ then $MinIdx[i, j] = MinIdx[i, j-1]$
          else $MinIdx[i, j] = j$

- The lowest common ancestor (LCA) of two nodes u_i, u_j of a tree T is the node with minimum level id that occurs on the simple path between u_i and u_j in T. Below, we show by preprocessing a given rooted tree T, using range minimum queries, the LCA queries can be answered in $O(1)$ time. Essentially, we reduce the problem of answering LCA queries to range minimum queries. For convenience, we assume each internal node of input tree T has exactly two children.

  Consider an Eulerian tour $\mathcal{T}$ of T that starts at the root u_1 of T, visits every edge of T exactly twice, and returns to u_1. Refer to the figure below. During preprocessing, with a DFT of T, an Eulerian tour $\mathcal{T}$ of T can be computed. Further, the following arrays are computed during preprocessing:
  Let E be the array that stores the node ids in the order in which they occur along $\mathcal{T}$.
  Let L be the array that stores the level ids of nodes in the order in which they occur along $\mathcal{T}$.
  A variable $time$ is initialized to 0. While walking along $\mathcal{T}$, $time$ is incremented whenever any node of T is visited. Let F be the array in which $F[i]$ stores the $time$ at which u_i is visited for the first time along $\mathcal{T}$. That is, each node u_i on $\mathcal{T}$ is associated with a unique time $F[i]$.

  For any two nodes u_i and u_j with $F[i] < F[j]$, the LCA of u_i and u_j belongs to the sequence $E[F[i]]$, $E[F[i]+1], \ldots, E[F[j]]$. And, the levels of the nodes of this portion of $\mathcal{T}$ are stored in $L[F[i]]$, $L[F[i]+1]$, $\ldots$, $L[F[j]]$. Then, the LCA of any two nodes u_i and u_j of T is $E[k']$, where k' is the result of range minimum query in L with indices $F[i]$ and $F[j]$.



1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
u_1	u_2	u_8	u_2	u_9	u_2	u_1	u_3	u_4	u_6	u_4	u_7	u_4	u_3	u_5	u_3	u_1

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
0	1	2	1	2	1	0	1	2	3	2	3	2	1	2	1	0

1	2	3	4	5	6	7	8	9
1	2	8	9	15	10	12	3	5

Arrays E, L, and F, are shown in that order from top to bottom.

In the above example, the LCA of u_2 and u_5 is $E[7]$, which is u_1. Here, 7 is obtained via a range minimum query in L with indices $F[2] = 2$ and $F[5] = 15$.

Considering $L[i+1]$ is equal to either $L[i]+1$ or $L[i]-1$ for each i with $1 \leq i \leq 2n-2$, Bender et al.,[1] devised an algorithm that preprocesses L in $O(n)$ time to answer any range minimum query in L in $O(1)$ time.

References:

Geometric Spanner Networks by M. Smid and G. Narasimhan, Cambridge University Press, 2007.

[1] M. A. Bender, M. F.-Colton, G. Pemmasani, S. Skiena, and P. Sumazin. Lowest common ancestors in trees and directed acyclic graphs. Journal of Algorithms, 57 (2): 75–94, 2005.