Lowest Common Ancestor Problem

Let T be a rooted tree on n nodes with root node r. A node w is said to be the lowest common ancestor of nodes u and v (denoted by $LCA(u, v)$) if it is ancestor of both u and v, and none of its children is an ancestor of both u, v in T. The aim is to maintain a data structure for the given tree T that can answer LCA queries efficiently.

Using the following two ideas, the LCA queries can be answered in $O(1)$ by using $O(n \log n)$ space.

1. Linearization of T using Euler Tour -
First the tree is transformed into an array as follows. Consider any Euler tour of T starting from node r. Let L be a list storing the pairs $(u, d(u))$ in the order in which the nodes get visited by the Euler tour, where $d(u)$ denotes the depth of node u in T. Then L is of size $2n - 1$. For each u in T the following information is stored.

1. Each node $u \in T$ stores an index i_u pointing to one of the occurrences of pair $(u, d(u))$ in L.
2. All the occurrences of pair $(u, d(u))$ in L stores a pointer to the corresponding node u in T.

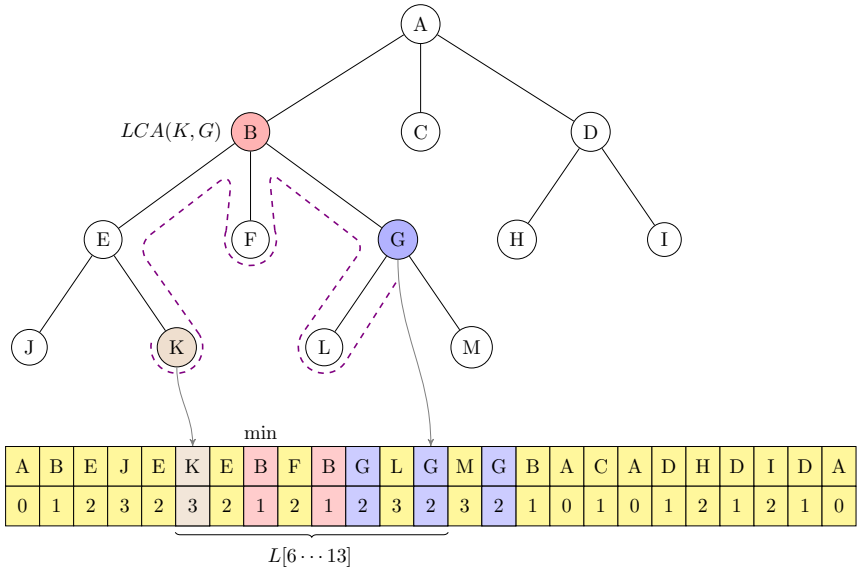

Figure 1: (i) Node K appears exactly once in L at index 6, so $i_K = 6$; (ii) Node G appears thrice at indices 11, 13 and 15, so i_G is an arbitrary value among these, which is 13; (iii) The LCA of K and G is the node in $L[6 \cdots 13]$ with the minimum depth value, which is B.

Observation 1. *For any two nodes u and v in T, let $(w, d(w))$ be a pair having minimum depth value in $L[i_u \cdots i_v]$, then $w = LCA(u, v)$.*

Thus to answer the LCA queries efficiently we need data structures that can solve the Range Minima problem described below.

Problem 1. *Let A be an array storing integers. We need to maintain a data structure for this array such that given any two indices i and j, we can report the minimum value in the subarray $A[i \cdots j]$ efficiently. Let the minimum value in subarray $A[i \cdots j]$ be denoted by $\min_A(i, j)$.*

2. Solution of Range Minima problem -
Let i, j be the indices of the range for which we want to compute the minimum value. Now let k be the largest integer such that $2^k \leq j - i$. Then note that interval $[i, .., j]$ is union of intervals $[i, .., i + 2^k]$ and $[j - 2^k, .., j]$. Therefore for each index i from 1 to n, we maintain two arrays P_i and Q_i each of logarithmic size. The array $P_i[k]$ stores the value $\min_A(i, i + 2^k)$ and the array $Q_i[k]$ stores the value $\min_A(i - 2^k, i)$. Thus $\min_A(i, j)$ is equal to $\min\{P_i[k], Q_j[k]\}$. The total space used will be $O(n \log n)$.

Observation 2. *The range minima queries for an array of size n can be answered in $O(1)$ time by using $O(n \log n)$ space data structure.*

The theorem 1 below follows directly from the fact that the solution for Range Minima problem can also be modified to compute a pair $(w, d(w))$ having the minimum depth value in the range $L[i_u \cdots i_v]$.

Theorem 1. *The LCA queries on a rooted tree with n nodes can be answered in $O(1)$ time by using $O(n \log n)$ space data structures.*

Extra Reading: Linear Space and constant query time Solution

Now to reduce the space to linear observe that the the list L is not any simple list storing integers, but it satisfies the property that the values of any two consecutive elements differ exactly by 1. So we divide the list into small blocks and construct a new array B having exactly one element corresponding to each of the small blocks. The Range Minima problem is solved separately for the blocks and array B.

More specifically L is divided into N blocks each of size $\frac{\log n}{2}$. Thus N is equal to $2n/\log n$. Let B be the new array of size N such that $B[i]$ stores a pair having the minimum depth value in block i. Thus the range minima queries on array B can be answered in $O(1)$ time by using $O(N \log N)$ space data structures, which is $O(n)$ space. Now corresponding to each block there is a base value, i.e. the value at its first index, and a logical array of size $\frac{\log n}{2}$. The entries in a logical array are $+1$ or -1 depending on whether the value in a cell increases by 1, or decreases by 1.

Lemma 1. *There are at most $\sqrt{n}$ logical arrays of size $\frac{\log n}{2}$.*

Proof. Each entry of a logical array is either $+1$ or -1, and the logical arrays are of size $\frac{\log n}{2}$, therefore, the total number of distinct logical arrays is $2^{\log n/2} = \sqrt{n}$. □

Lemma 2. *The Range Minima problem for all N blocks can be answered in $O(1)$ time by using $O(n + \sqrt{n}\log^2 n)$ space data structures.*

Proof. Using the simple approach of storing the solutions for all possible range combinations, the range minima queries for a single block can be answered in constant time by using $O(log^2 n)$ space. Now there are at most $\sqrt{n}$ distinct logical arrays, thus solutions for all logical arrays can be stored in $O(\sqrt{n}\log^2 n)$ space. As each of the blocks can store a pointer to the logical array corresponding to it, the range minima queries for all the blocks can be answered in constant time by using a total of $O(n + \sqrt{n}\log^2 n)$ space. □

Recall that to compute the LCA of nodes u and v we need to compute a pair $(w, d(w))$ having the minimum depth value in range $L[i_u \cdots i_v]$. Now let r and s be the indices of the blocks corresponding to i_u and i_v in array B. Note that if the pair having the minimum depth lies in the r^{th} block or s^{th} block, then it can be computed in constant time by using lemma 2. If it lies in the blocks between r and s, then the minimum depth value will be $\min_B(r + 1, s - 1)$. Thus by comparing the three values corresponding to r^{th} block, s^{th} block, and $\min_B(r + 1, s - 1)$, the pair $(w, d(w))$ having the minimum depth value in range $L[i_u \cdots i_v]$ can be computed in $O(1)$ time. Thus, the result follows.

Theorem 2. *The LCA queries on a rooted tree with n nodes can be answered in $O(1)$ time by using $O(n)$ space data structures.*