Efficient Lowest Common Ancestor (LCA) Queries

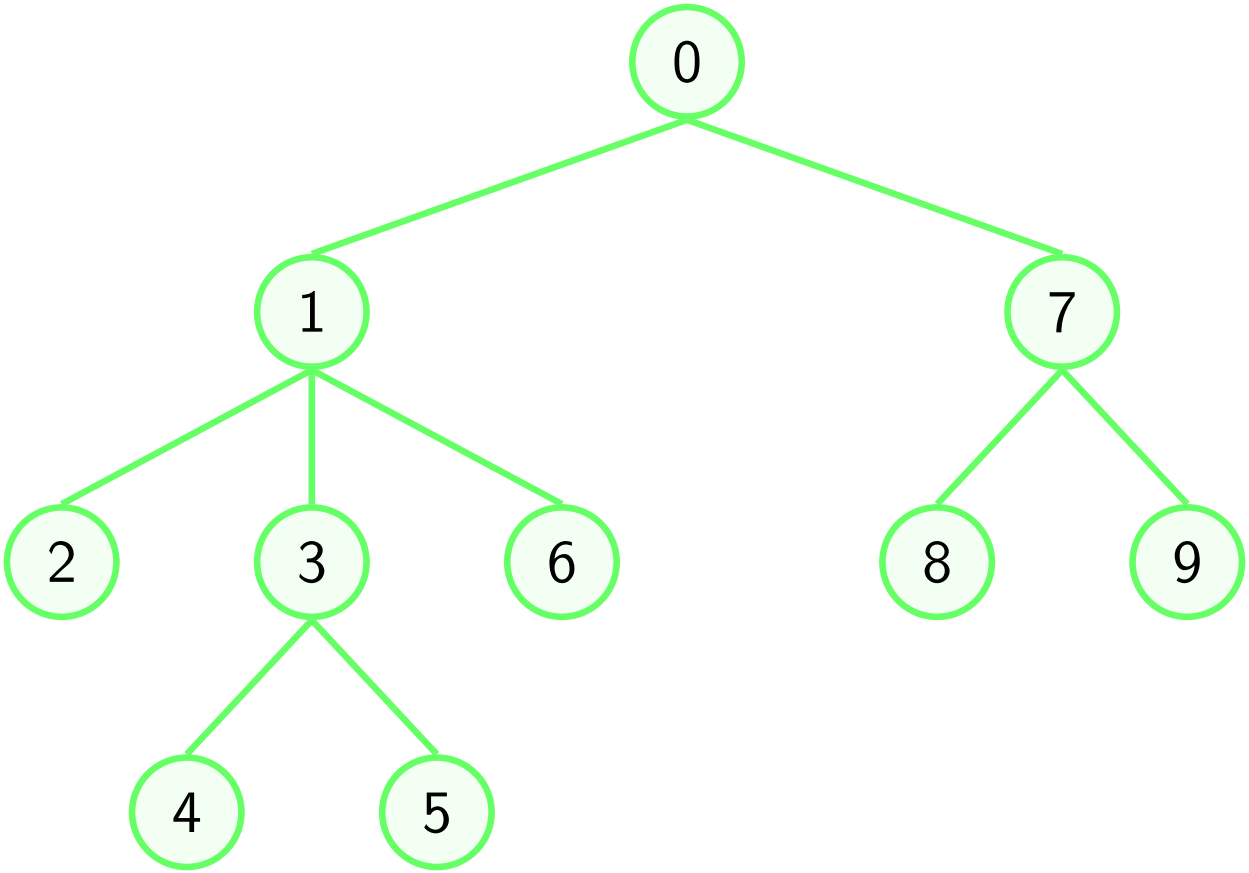
And Range Minimum Query (RMQ) Problem

Bo Gong

bogong.ustc@gmail.com

February 13, 2015

The Problem: Lowest Common Ancestor (LCA)



Credit: Steven Halim, Felix Halim. *Competitive Programming 3*. 2013.

What I will present

1. The Problem: Lowest Common Ancestor (LCA)
2. An O(VlogV + Q) solution
   - Reduce LCA to Range Minimum Query (RMQ)
   - Range Minimum Query (RMQ): Sparse Table (ST) algorithm
3. An O(VlogV + QlogV) solution
   - Dynamic Programming

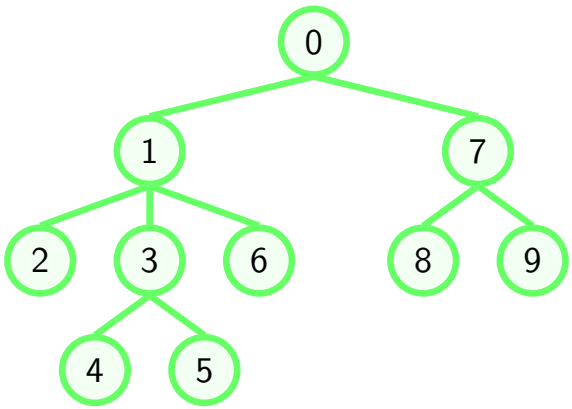
Next

1. The Problem: Lowest Common Ancestor (LCA)

2. An O(VlogV + Q) solution
   - Reduce LCA to Range Minimum Query (RMQ)
   - Range Minimum Query (RMQ): Sparse Table (ST) algorithm

3. An O(VlogV + QlogV) solution
   - Dynamic Programming

Reduce LCA to Range Minimum Query (RMQ)

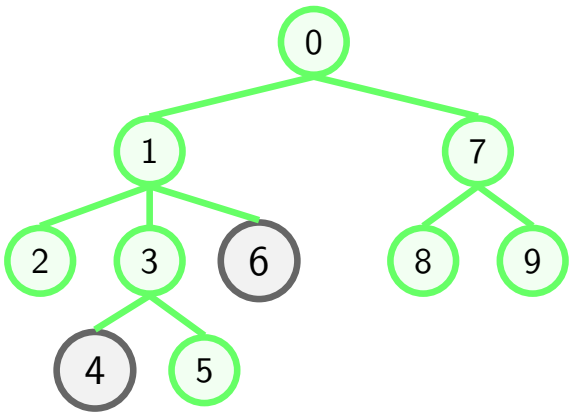



```
int L[2*MAX_N], T[2*MAX_N], F[MAX_N], idx;
void dfs(int cur, int depth) {
    F[cur] = idx;
    T[idx] = cur;
    L[idx++] = depth;
    for (int i=0;i<children[cur].size();i++){
        dfs(children[cur][i], depth+1);
        T[idx] = cur; // backtrack to cur node
        L[idx++] = depth;
    }
}
void buildRMQ() {
    idx = 0;
    memset(F, -1, sizeof F);
    dfs(0, 0); // assume root at index 0
}
```

Index	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
First Occurrence	0	1	2	4	5	7	10	13	14	16	-1	-1	-1	-1	-1	-1	-1	-1	-1
Traversal Order	0	1	2	1	3	4	3	5	3	1	6	1	0	7	8	7	9	7	0
Level	0	1	2	1	2	3	2	3	2	1	2	1	0	1	2	1	2	1	0

Credit: Steven Halim, Felix Halim. *Competitive Programming 3*. 2013.

Reduce LCA to Range Minimum Query (RMQ)

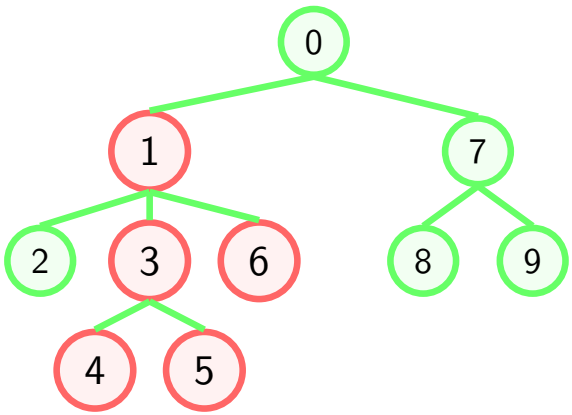


Take LCA(4,6) as an example:

Index	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
First Occurrence	0	1	2	4	5	7	10	13	14	16	-1	-1	-1	-1	-1	-1	-1	-1	-1
Traversal Order	0	1	2	1	3	4	3	5	3	1	6	1	0	7	8	7	9	7	0
Level	0	1	2	1	2	3	2	3	2	1	2	1	0	1	2	1	2	1	0

Credit: Steven Halim, Felix Halim. *Competitive Programming 3*. 2013.

Reduce LCA to Range Minimum Query (RMQ)



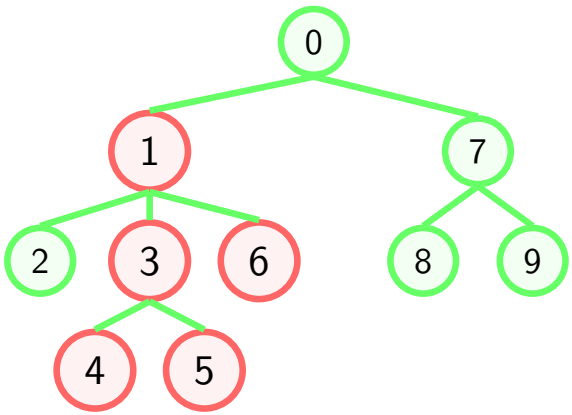
Take LCA(4,6) as an example:

- F[4] = 5, F[6] = 10
- Need to find smallest depth in T[5..10]

Index	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
First Occurrence	0	1	2	4	5	7	10	13	14	16	-1	-1	-1	-1	-1	-1	-1	-1	-1
Traversal Order	0	1	2	1	3	4	3	5	3	1	6	1	0	7	8	7	9	7	0
Level	0	1	2	1	2	3	2	3	2	1	2	1	0	1	2	1	2	1	0

Credit: Steven Halim, Felix Halim. *Competitive Programming 3*. 2013.

Reduce LCA to Range Minimum Query (RMQ)



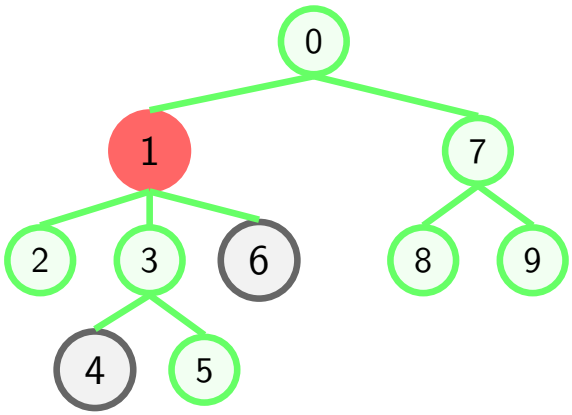
Take LCA(4,6) as an example:

- F[4] = 5, F[6] = 10
- Need to find smallest depth in T[5..10]
- Calling RMQ(5,10) on array L returns index 9

Index	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
First Occurrence	0	1	2	4	5	7	10	13	14	16	-1	-1	-1	-1	-1	-1	-1	-1	-1
Traversal Order	0	1	2	1	3	4	3	5	3	1	6	1	0	7	8	7	9	7	0
Level	0	1	2	1	2	3	2	3	2	1	2	1	0	1	2	1	2	1	0

Credit: Steven Halim, Felix Halim. *Competitive Programming 3*. 2013.

Reduce LCA to Range Minimum Query (RMQ)



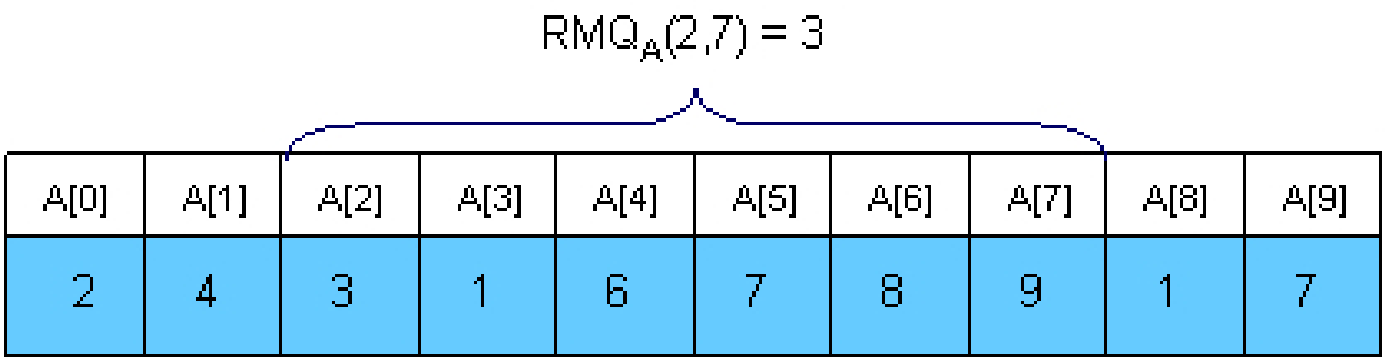
Take LCA(4,6) as an example:

- F[4] = 5, F[6] = 10
- Need to find smallest depth in T[5..10]
- Calling RMQ(5,10) on array L returns index 9
- The value of T[9] = 1
- Report LCA(4,6) = 1

Index	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
First Occurrence	0	1	2	4	5	7	10	13	14	16	-1	-1	-1	-1	-1	-1	-1	-1	-1
Traversal Order	0	1	2	1	3	4	3	5	3	1	6	1	0	7	8	7	9	7	0
Level	0	1	2	1	2	3	2	3	2	1	2	1	0	1	2	1	2	1	0

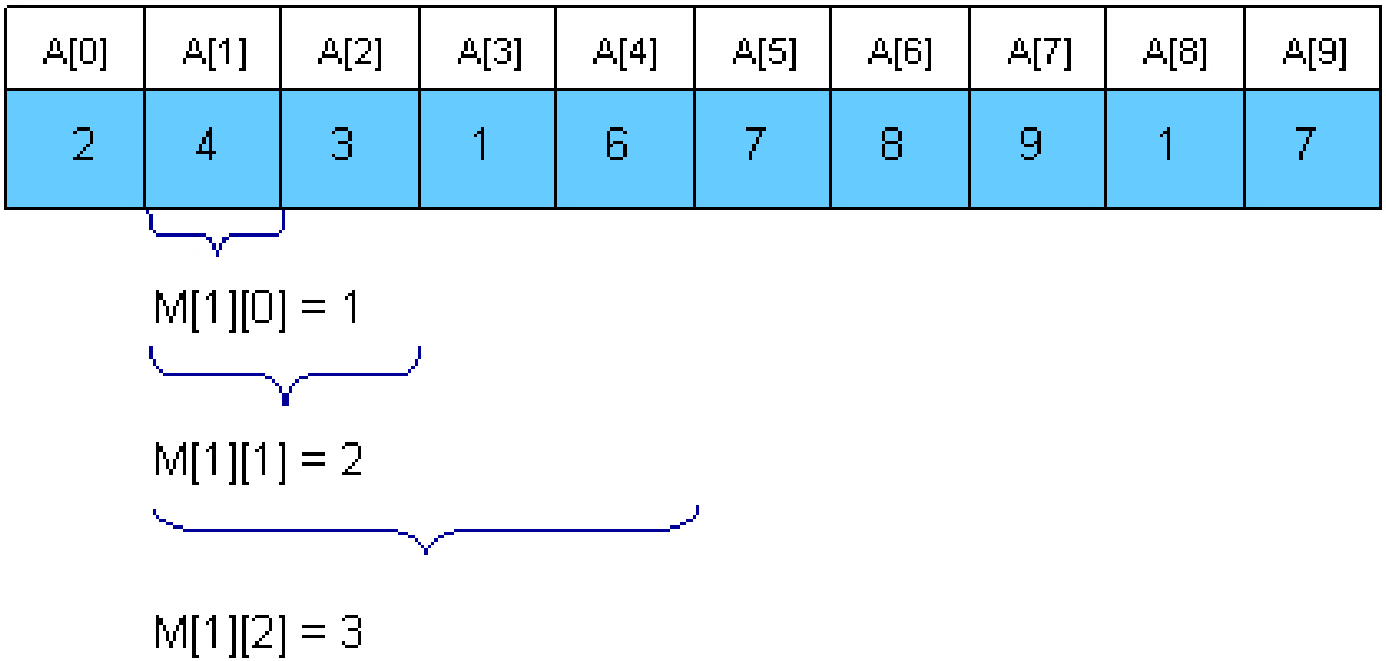
Credit: Steven Halim, Felix Halim. *Competitive Programming 3*. 2013.

Range Minimum Query (RMQ)



Credit: danielp @ http://community.topcoder.com/tc?module=Static&d1=tutorials&d2=lowestCommonAncestor

Sparse Table (ST) algorithm



Credit: danielp @ http://community.topcoder.com/tc?module=Static&d1=tutorials&d2=lowestCommonAncestor

Sparse Table (ST) algorithm

$$k = \lfloor log_2(j - i + 1) \rfloor$$

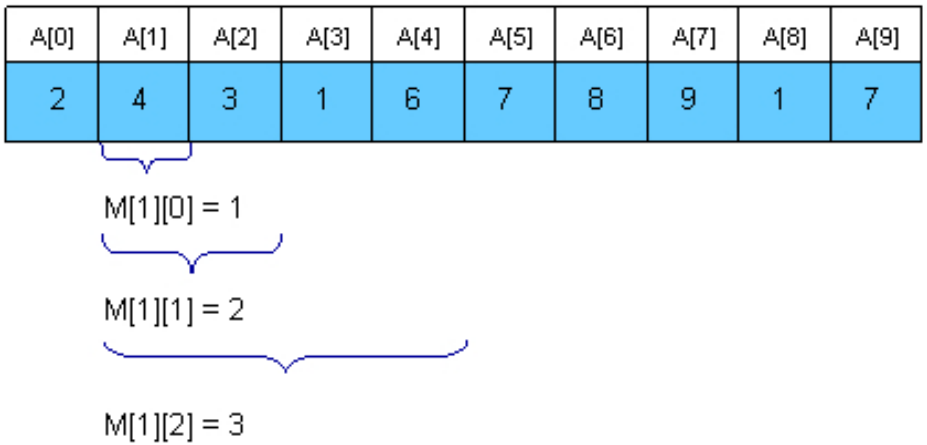
$$2^k \leq (j - i + 1)$$

Credit: Steven Halim, Felix Halim. *Competitive Programming 3*. 2013.

Sparse Table (ST) algorithm

To preprocess RMQ using dynamic programming:

- Keep an array M[0, N-1][0, logN].
- Consider a sub-array starting at i with length 2^j:
  - M[i][j] is the index of the minimum value.



Credit: danielp @ http://community.topcoder.com/tc?module=Static&d1=tutorials&d2=lowestCommonAncestor

Sparse Table (ST) algorithm

To preprocess RMQ using dynamic programming:

- Keep an array M[0, N-1][0, logN].
- Consider a sub-array starting at i with length 2^j:
  - M[i][j] is the index of the minimum value.

$$M[i][j] = \begin{cases} M[i][j-1], & A[M[i][j-1]] <= A[M[i+2^{j-1}-1][j-1]] \\ M[i+2^{j-1}-1][j-1], & otherwise \end{cases}$$

Credit: danielp @ http://community.topcoder.com/tc?module=Static&d1=tutorials&d2=lowestCommonAncestor

Sparse Table (ST) algorithm: Example

$$M[i][j] = \begin{cases} M[i][j-1], & A[M[i][j-1]] <= A[M[i+2^{j-1}-1][j-1]] \\ M[i+2^{j-1}-1][j-1], & otherwise \end{cases}$$

Example: Array A: 18, 17, 13, 19, 15, 11, 20

index	0	1	2
0	0	1	2
1	1	2	2
2	2	2	5
3	3	4	5
4	4	5	empty
5	5	5	empty
6	6	empty	empty

Credit: Steven Halim, Felix Halim. *Competitive Programming 3*. 2013.

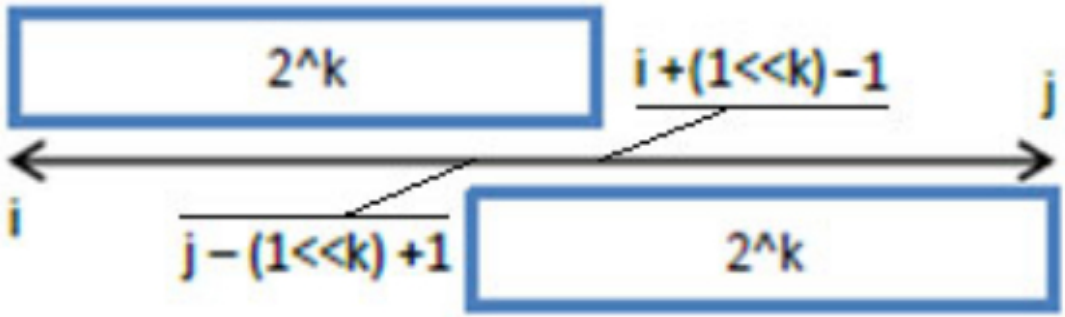
Preprocessing: O(VlogV)

```
void process2(int M[MAXN][LOGMAXN], int A[MAXN], int N) {
    int i, j;

    //initialize M for the intervals with length 1
    for (i = 0; i < N; i++)
        M[i][0] = i;
    //compute values from smaller to bigger intervals
    for (j = 1; 1 << j <= N; j++)
        for (i = 0; i + (1 << j) - 1 < N; i++)
            if (A[M[i][j - 1]] < A[M[i + (1 << (j - 1))][j - 1]])
                M[i][j] = M[i][j - 1];
            else
                M[i][j] = M[i + (1 << (j - 1))][j - 1];
}
```

Credit: danielp @ http://community.topcoder.com/tc?module=Static&d1=tutorials&d2=lowestCommonAncestor

Query: O(1)



$$RMQ_A(i,j) = \begin{cases} M[i][k], & A[M[i][k]] <= A[M[j-2^k+1][k]] \\ M[j-2^k+1][k], & otherwise \end{cases}$$

Credit: danielp @ http://community.topcoder.com/tc?module=Static&d1=tutorials&d2=lowestCommonAncestor

Time Complexity: O(VlogV + Q)

Total time complexity: O(VlogV + Q)

- Reduction of LCA to RMQ: O(V)
- RMQ - Preprocessing: O(VlogV)
- RMQ - Query: O(1)

Credit: danielp @ http://community.topcoder.com/tc?module=Static&d1=tutorials&d2=lowestCommonAncestor

Next

1. The Problem: Lowest Common Ancestor (LCA)

2. An O(VlogV + Q) solution
   - Reduce LCA to Range Minimum Query (RMQ)
   - Range Minimum Query (RMQ): Sparse Table (ST) algorithm

3. An O(VlogV + QlogV) solution
   - Dynamic Programming

Dynamic Programming

Dynamic Programming

- Compute a table P[1,N][1,logN]
- P[i][j] is the 2^j-th ancestor of i

$$P[i][j] = \begin{cases} T[i], & j = 0 \\ P[P[i][j-1]][j-1], & j > 0 \end{cases}$$

Credit: danielp @ http://community.topcoder.com/tc?module=Static&d1=tutorials&d2=lowestCommonAncestor

DP - Preprocessing: O(VlogV)

$$P[i][j] = \begin{cases} T[i], & j = 0 \\ P[P[i][j-1]][j-1], & j > 0 \end{cases}$$

```
void process3(int N, int T[MAXN], int P[MAXN][LOGMAXN]) {
    int i, j;

    //we initialize every element in P with -1
    for (i = 0; i < N; i++)
        for (j = 0; 1 << j < N; j++)
            P[i][j] = -1;

    //the first ancestor of every node i is T[i]
    for (i = 0; i < N; i++)
        P[i][0] = T[i];

    //bottom up dynamic programing
    for (j = 1; 1 << j < N; j++)
        for (i = 0; i < N; i++)
            if (P[i][j - 1] != -1)
                P[i][j] = P[P[i][j - 1]][j - 1];
}
```

Credit: danielp @ http://community.topcoder.com/tc?module=Static&d1=tutorials&d2=lowestCommonAncestor

DP - Query: O(logV)

```
int query(int N, int P[MAXN][LOGMAXN], int T[MAXN],
int L[MAXN], int p, int q) {
    int tmp, log, i;
    //if p is situated on a higher level than q then swap them
    if (L[p] < L[q])
        tmp = p, p = q, q = tmp;
    //we compute the value of [log(L[p)]
    for (log = 1; 1 << log <= L[p]; log++);
    log--;
    //we find the ancestor of node p situated on the same level
    //with q using the values in P
    for (i = log; i >= 0; i--)
        if (L[p] - (1 << i) >= L[q])
            p = P[p][i];
    if (p == q)
        return p;
    //we compute LCA(p, q) using the values in P
    for (i = log; i >= 0; i--)
        if (P[p][i] != -1 && P[p][i] != P[q][i])
            p = P[p][i], q = P[q][i];
    return T[p];
}
```

Credit: danielp @ http://community.topcoder.com/tc?module=Static&d1=tutorials&d2=lowestCommonAncestor

Time Complexity: O(VlogV + QlogV)

Total time complexity: O(VlogV + QlogV)

- DP - Preprocessing: O(VlogV)
- DP - Query: O(logV)

Credit: danielp @ http://community.topcoder.com/tc?module=Static&d1=tutorials&d2=lowestCommonAncestor

Questions?