Competitive Programming

From Problem 2 Solution in O(1)

Graph Theory

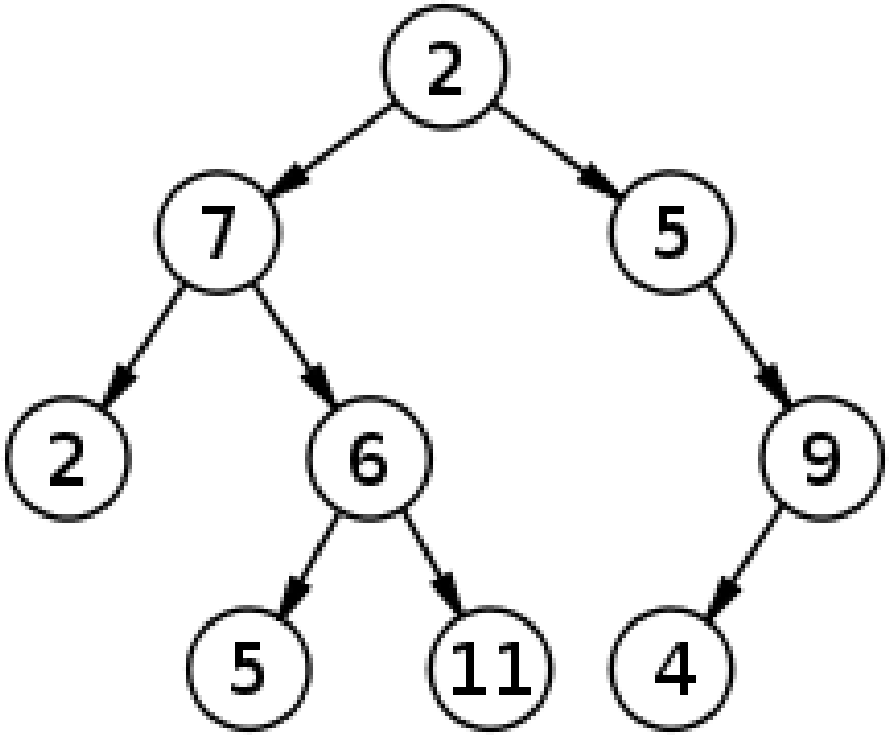
Lowest Common Ancestor (LCA)

Mostafa Saad Ibrahim
PhD Student @ Simon Fraser University

Acknowledgement

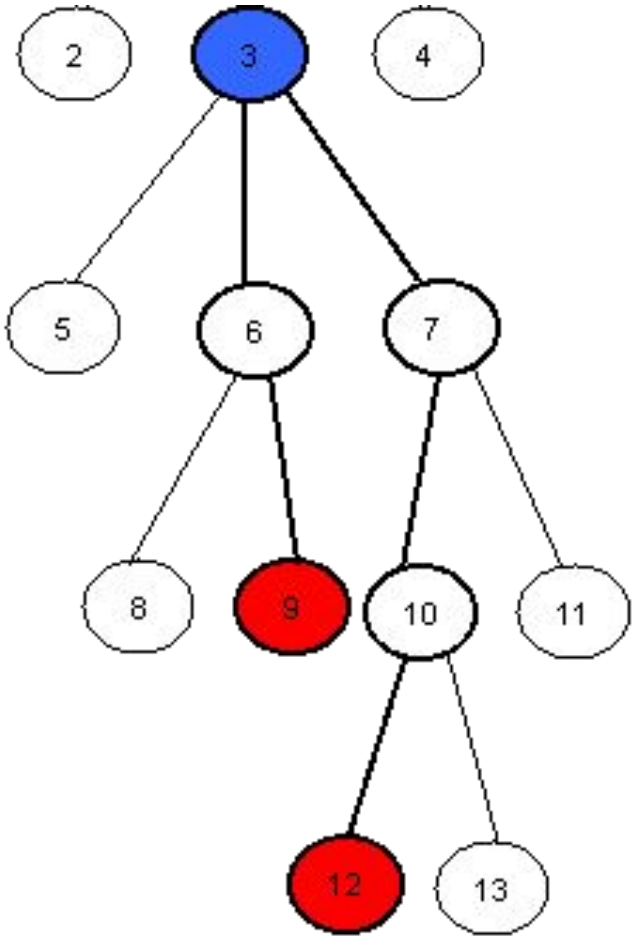
- Much of content based on topcoder article.
- See

Node Ancestors



Ancestors(11) = 6, 7, 2
Ancestors(4) = 9, 5, 2

Query LCA(9, 12)



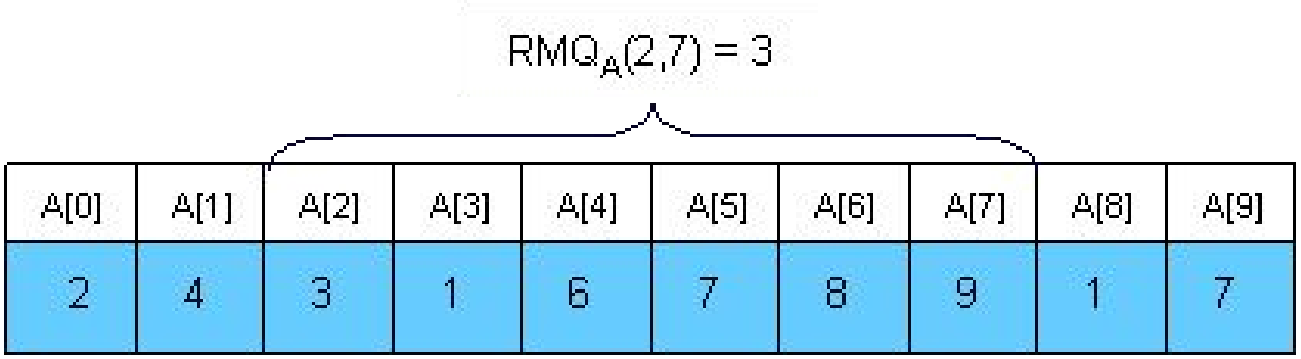
$LCA_T(9, 12) = 3$

Background

- Let’s revise some background
- Range Minimum Query Problem
- Euler Tour on Tree

Range Minimum Query (RMQ)

- Given array of N values
- Q queries (start, end): find min value in array
- BF: Direct loop per query O(N)
- For Q queries: O(NQ)



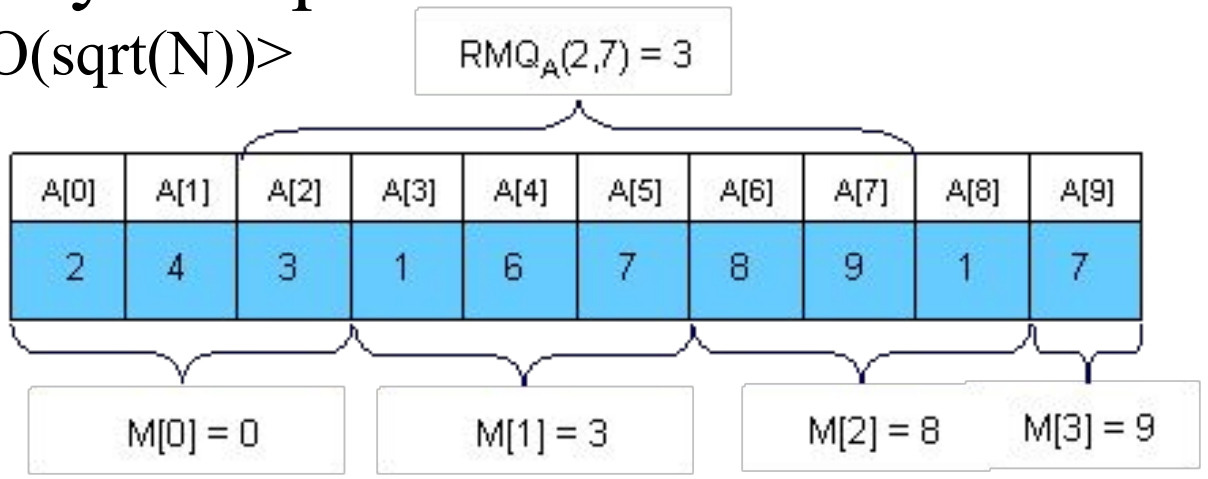
Src: https://www.topcoder.com/community/data-science/data-science-tutorials/range-minimum-query-and-lowest-common-ancestor/#A%20O(N),%20O(1)%20algorithm%20for%20the%20restricted%20RMQ

Querying problems

- Instead of answering every query directly
- Preprocess the input first say in O(X)
- Then answer each query in O(Y)
- So for Q Queries order: O(X + QY)
- Different **preprocessing** times/styles => Different algorithms

Range Minimum Query (RMQ)

- We can think of many preprocessing styles
- E.g. In O(N^2): Preprocess and compute all pairs answers.
  - O(N^2) pre-processing and O(1) query
- Divide array to Sqrt blocks
  - <O(N), O(sqrt(N))>



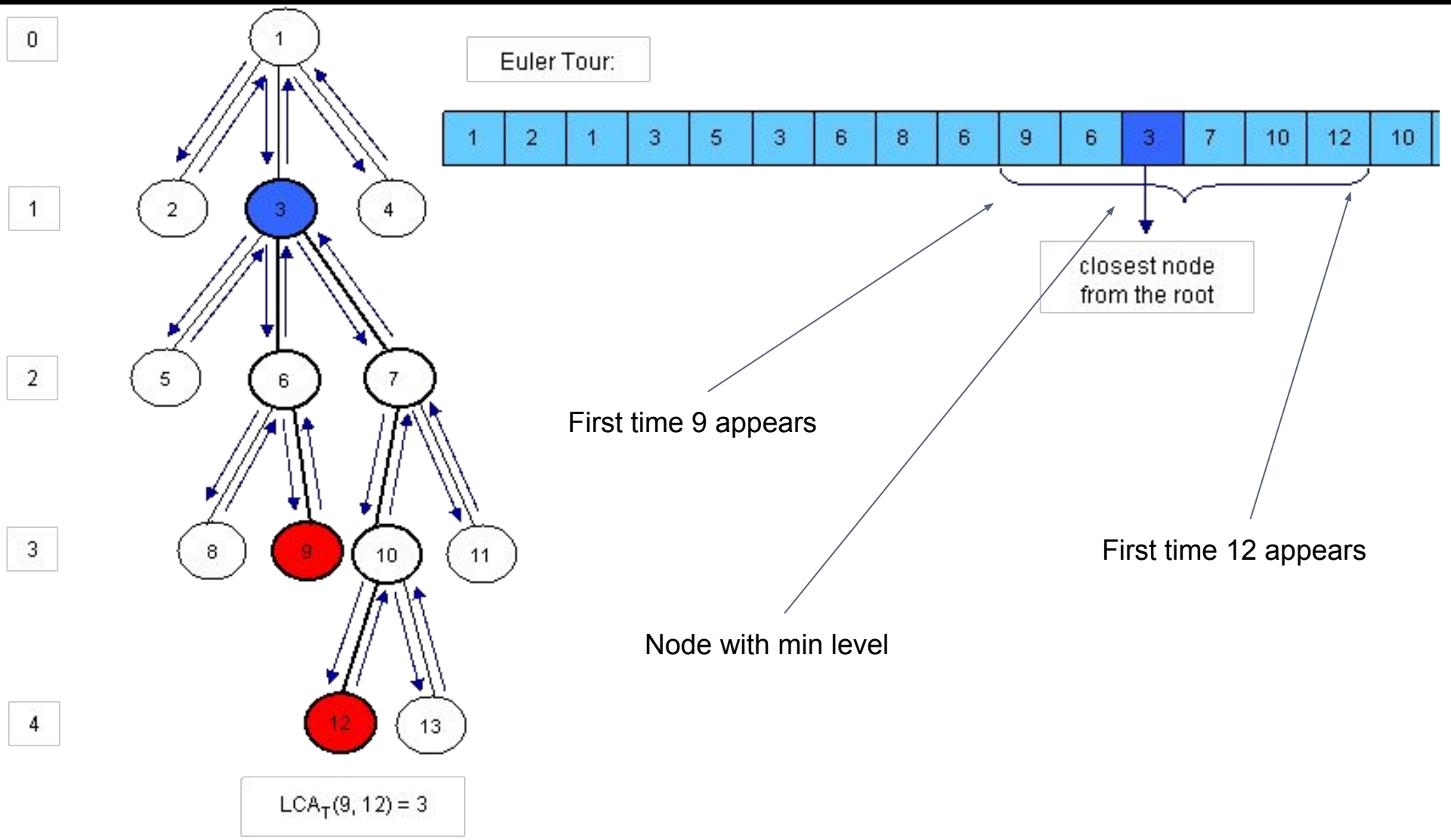
Range Minimum Query (RMQ)

- Using Dynamic Programming
  - <O(N logN), O(1)>.
- Using Segment Tree (revise channel)
  - <O(N), O(logN)>
- Overall
  - Several pre-processing styles and different querying styles
  - From problem to another, we select the best
  - Sqrt idea is efficient and can be applied to other problems

Euler Tour on Tree

- Euler tour is the path along the tree that begins at the root and ends at the root, traversing each edge exactly twice
- Once to enter the subtree at the other endpoint and once to leave it.
- You can think of an Euler tour as just being a **depth first traversal** where we return to the root at the end.
- Src: https://courses.csail.mit.edu/6.851/spring07/scribe/lec05.pdf

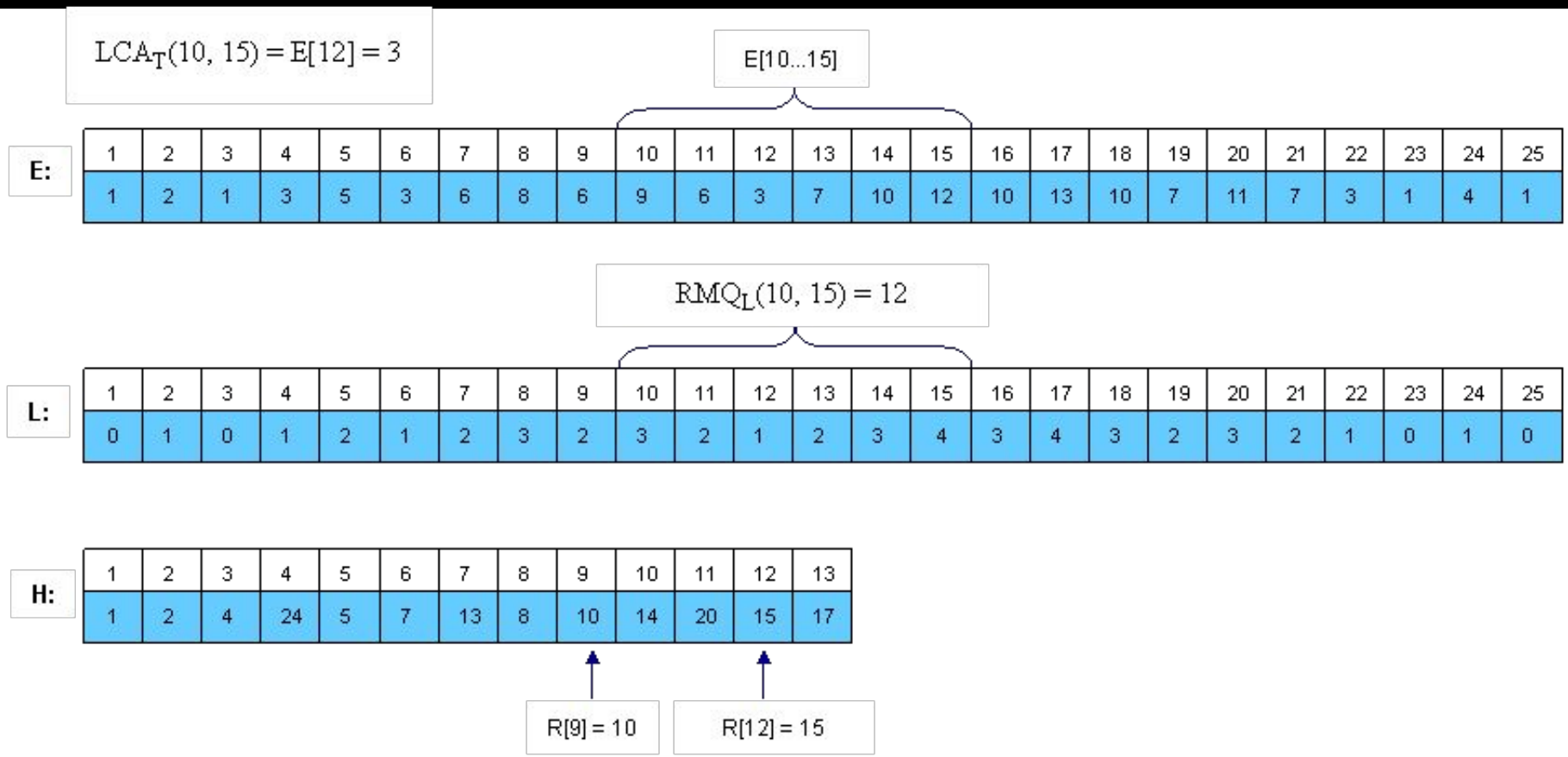
Euler Tour on Tree



LCA

- Let's define the following arrays
- **E[1, 2*N-1]** – the nodes visited in an Euler Tour of T;
- **L[1, 2*N-1]** – the levels of the nodes visited in the Euler Tour; L[i] is the level of node E[i]
- **H[1, N]** – H[i] is the index of the first occurrence of node i in E (or any occurrence)

LCA(9, 12)



- From H: Get occurences of 9, 12 in Euler Tour ⇒ (10, 15)
- From L: get the index of the minimum tree level ⇒ index 12 for tree level 1
- From E, get value of index 12 ⇒ 3

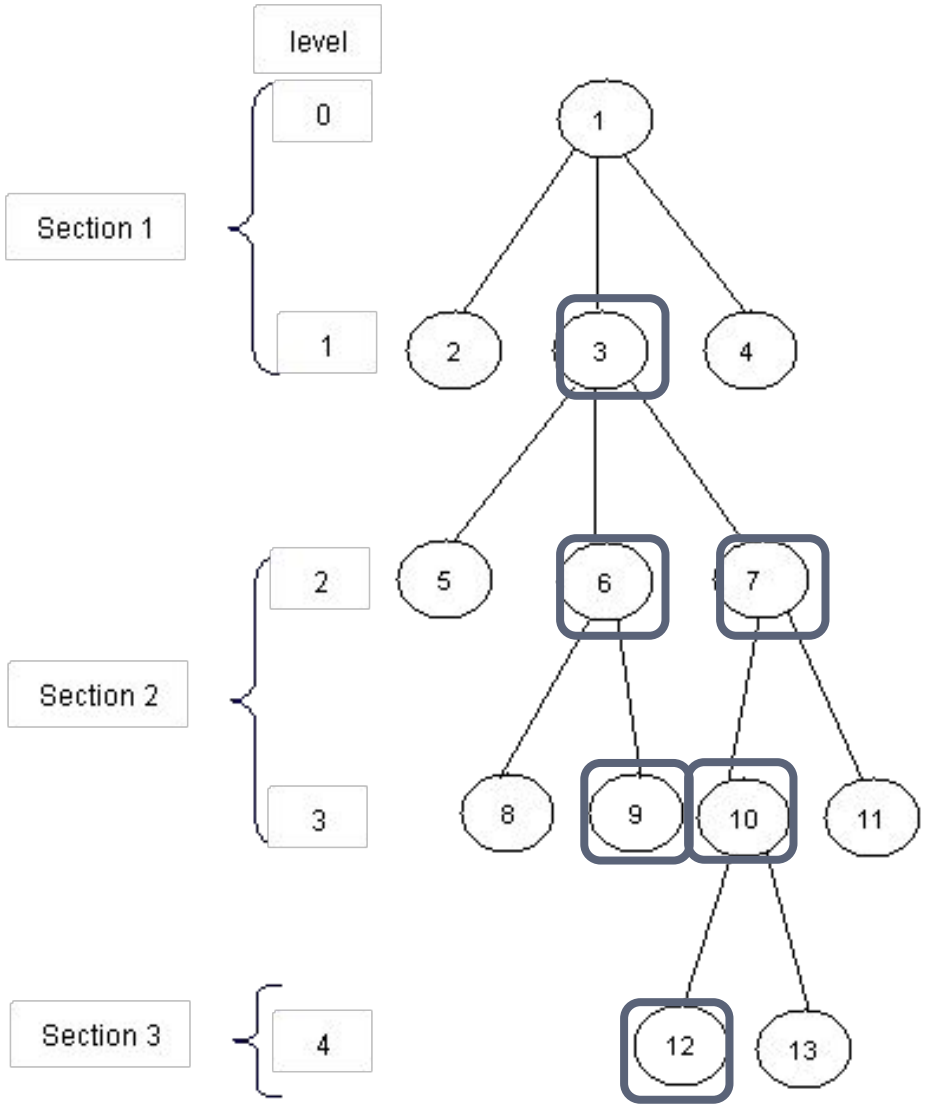
LCA Algorithms

- With Overall <O(N), O(logN)>, very efficient algorithm
- Easy logic..but much code, code
- With dynamic programming, we can implement <O(N logN), O(logN)>
  - and it is easy to code once understood it

Slow bottom up approach

- A slight different version of the bottom approach
- Find level of p and q
- If different, keep moving up the longer one so that be on same level
- Keep moving both together, till find same common parent

Slow bottom up approach



9 on level 3
12 on level 4

move 12 up one level to be SAME level

Move up together till common node

2^j Processing style

- **Walking** up step by step is very slow
- Sometimes we don't need every element
- What about **jumping** instead of walking?
- Using DP, compute 2^j ancestor of a node
- Use that to jump up as possible
  - Recursively, we can keep jumping from new positions

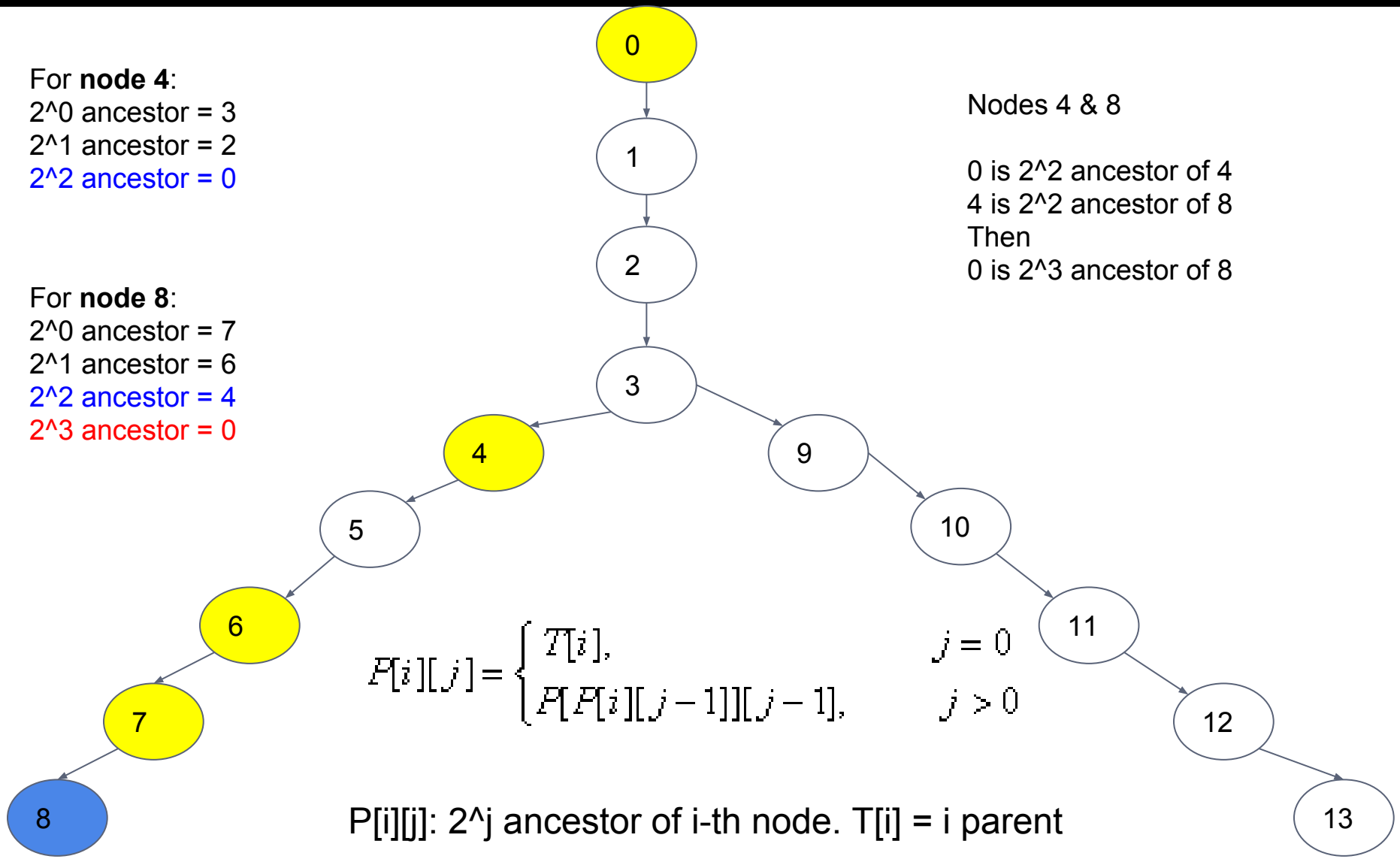
2^j Ancestor of ith node

For **node 4**:
2^0 ancestor = 3
2^1 ancestor = 2
2^2 ancestor = 0

For **node 8**:
2^0 ancestor = 7
2^1 ancestor = 6
2^2 ancestor = 4
2^3 ancestor = 0

Nodes 4 & 8

0 is 2^2 ancestor of 4
4 is 2^2 ancestor of 8
Then
0 is 2^3 ancestor of 8



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

P[i][j]: 2^j ancestor of i-th node. T[i] = i parent

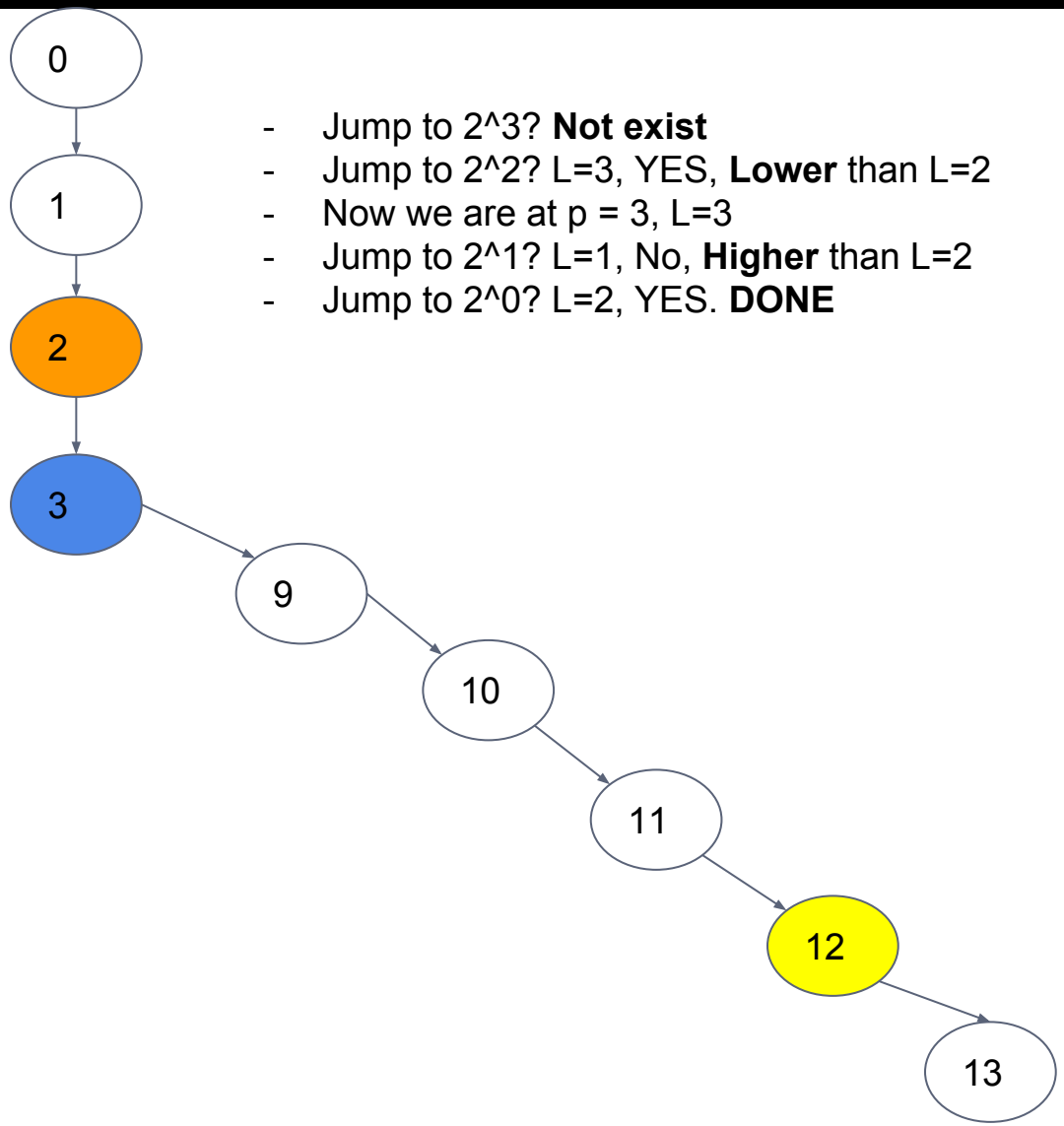
Find kth ancestor

For **node 12**:
L[12] = 7
2^0 ancestor = 11
2^1 ancestor = 10
2^2 ancestor = 3

5th ancestor? 7-5 in L=2

Find ancestor in L=2?

- Let p current node
- Move to highest possible ancestor below L=2 using 2^j
- Let that be q
- Again...till reach L=2



- Jump to 2^3? **Not exist**
- Jump to 2^2? L=3, YES, **Lower** than L=2
- Now we are at p = 3, L=3
- Jump to 2^1? L=1, No, **Higher** than L=2
- Jump to 2^0? L=2, YES. **DONE**

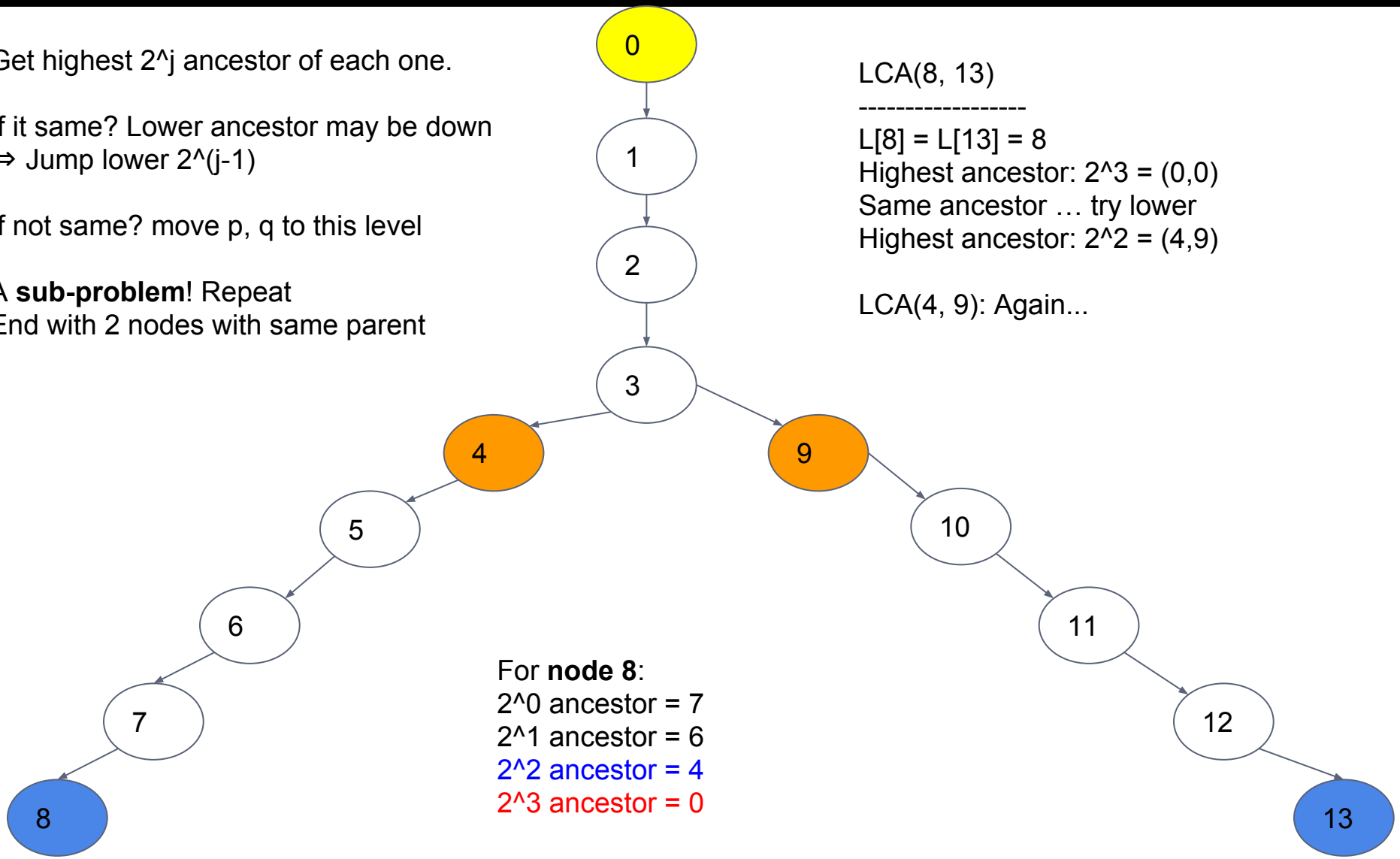
LCA(8, 13): Same initial level

Get highest 2^j ancestor of each one.

If it same? Lower ancestor may be down
⇒ Jump lower 2^(j-1)

If not same? move p, q to this level

A **sub-problem**! Repeat
End with 2 nodes with same parent



LCA(8, 13)

L[8] = L[13] = 8
Highest ancestor: 2^3 = (0,0)
Same ancestor … try lower
Highest ancestor: 2^2 = (4,9)

LCA(4, 9): Again...

For **node 8**:
2^0 ancestor = 7
2^1 ancestor = 6
2^2 ancestor = 4
2^3 ancestor = 0

LCA(p, q)

- If they are on same level, do previous steps
- Otherwise
  - Assume p is in lower level
  - Assume q in level L[q]
  - Move p to Level[q]
  - If p = q, then q is ancestor of p
  - Else, do normal steps of same level processing

تم بحمد الله

علمكم الله ما ينفعكم

ونفعكم بما تعلمتم

وزادكم علماً

Problems

- SRM 310 -> Floating Median
- http://acm.pku.edu.cn/JudgeOnline/problem?id=1986
- http://acm.pku.edu.cn/JudgeOnline/problem?id=2374
- http://acmicpc-live-archive.uva.es/nuevoportal/data/problem.php?p=2045
- http://acm.pku.edu.cn/JudgeOnline/problem?id=2763
- http://www.spoj.pl/problems/QTREE2/
- http://acm.uva.es/p/v109/10938.html
- http://acm.sgu.ru/problem.php?contest=0&problem=155