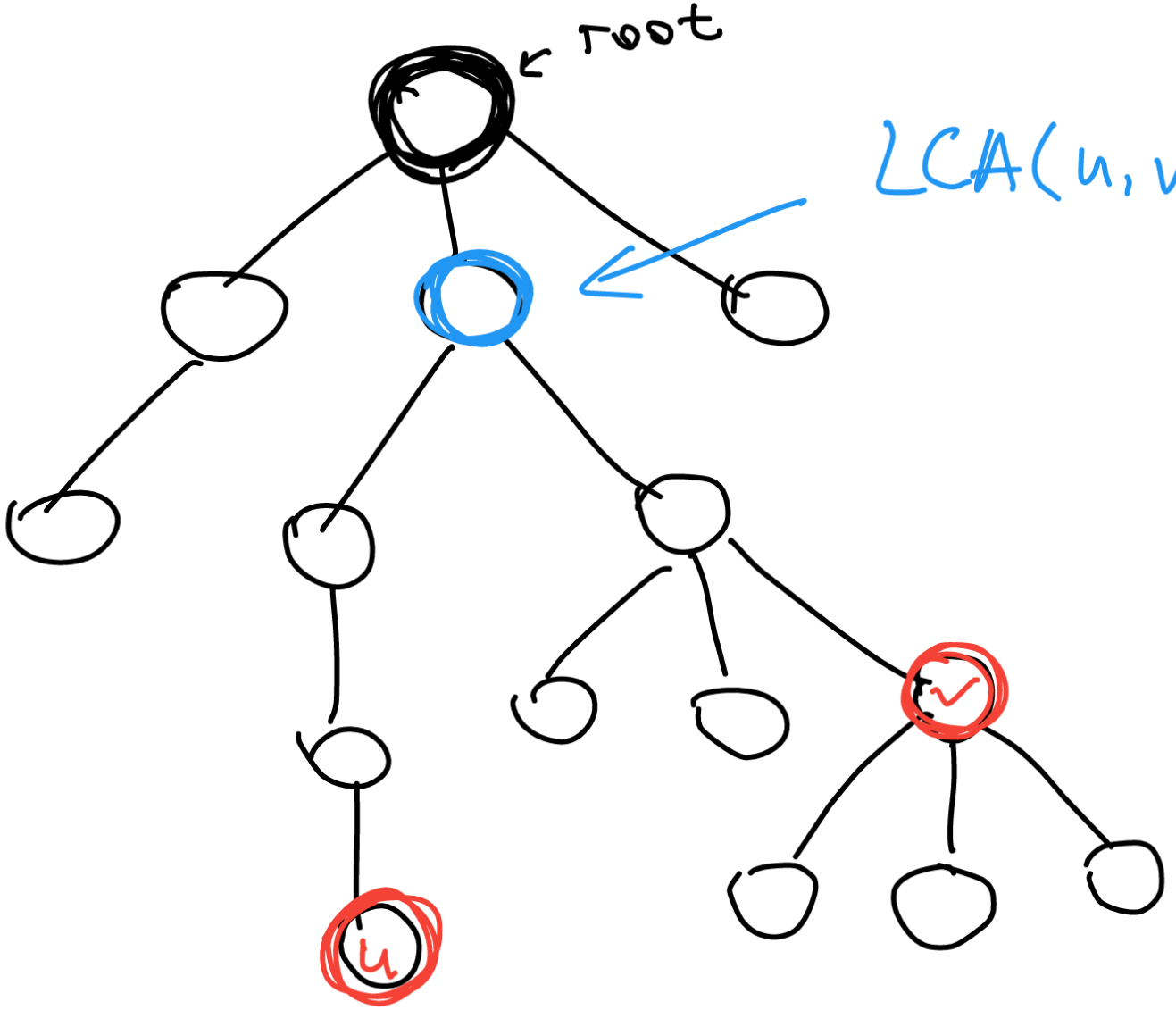
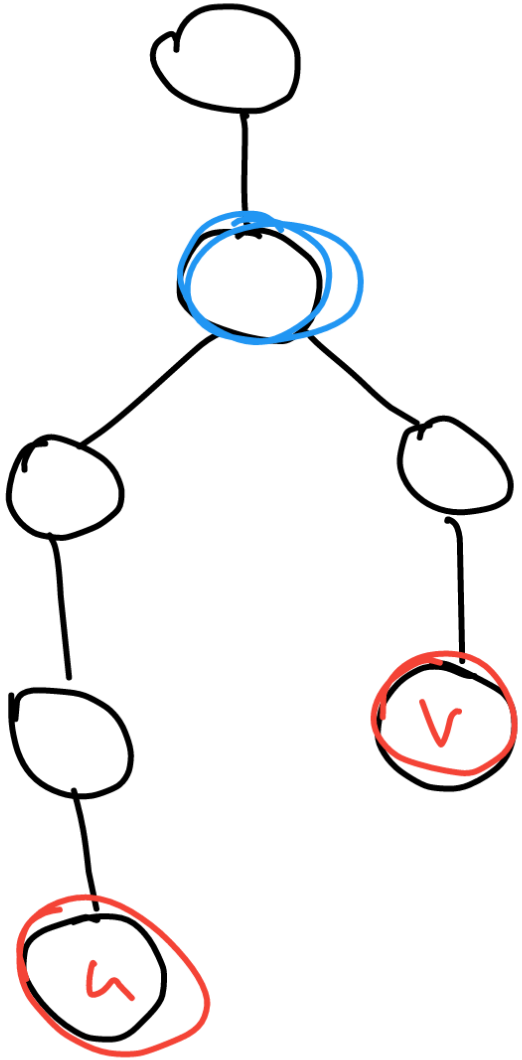
LCA : Lowest Common Ancestor



$\langle O(n), O(depth[u] + depth[v]) \rangle$

precalc depth[u] (1 DFS)

```
while u ≠ v:
    if depth[u] ≥ depth[v]:
        u = par[u]
    else:
        v = par[v]
```



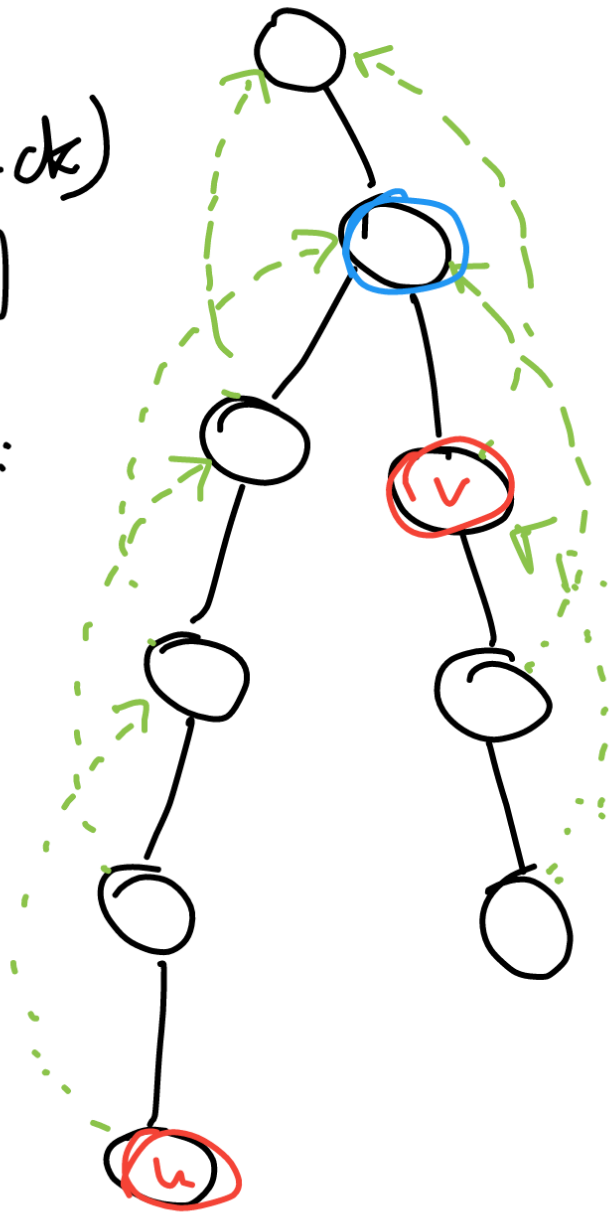
$\langle O(n), O(\sqrt{n}) \rangle$: Square root idea:

up[u]: $\sqrt{n}$ steps up from u

precalc during DFS (using stack)

up[u] = stack[sz(stack) $-\sqrt{n}$]

```
while depth[up[u]] > depth[v]:
    u = up[u]
while depth[u] > depth[v]:
    u = par[u]
while up[u] ≠ up[v]:
    u = up[u]
    v = up[v]
while u ≠ v:
    u = par[u], v = par[v]
```



This can be improved to $\langle O(n), O(kn) \rangle$ and $\langle O(N \log N), O(\log N) \rangle$

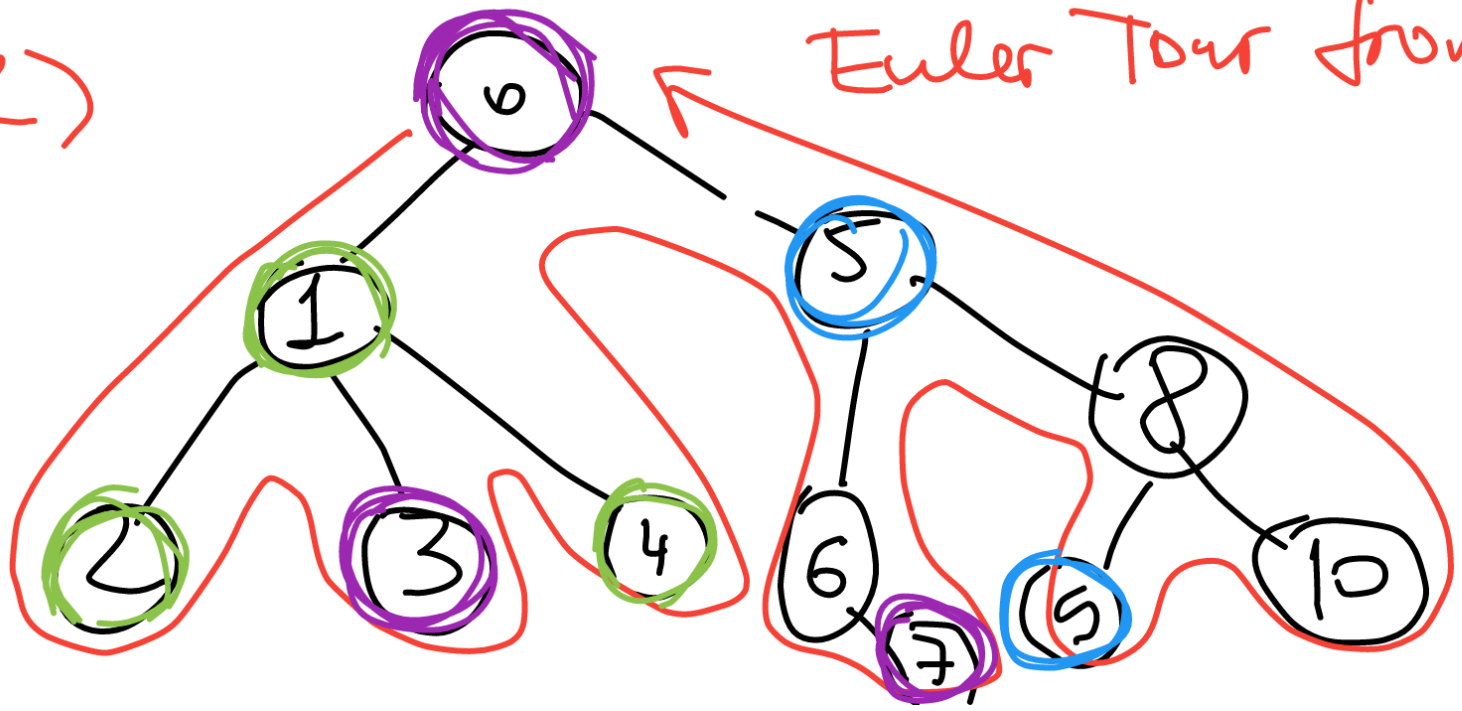
$up[u, k] = stack[sz(stack) - 2^k]$

jump up to max possible power of 2

Reduction to RMQ:

1) order vertices in DFS order

2) Euler Tour from root



Euler Tour:

0 1 2 1 3 2 4 1 0 5 6 7 6 5 8 9 ...

$LCA(u, v) = RMQ_{ET}(F(u), F(v))$

$F(u)$ - first occurence of u in Euler Tour.

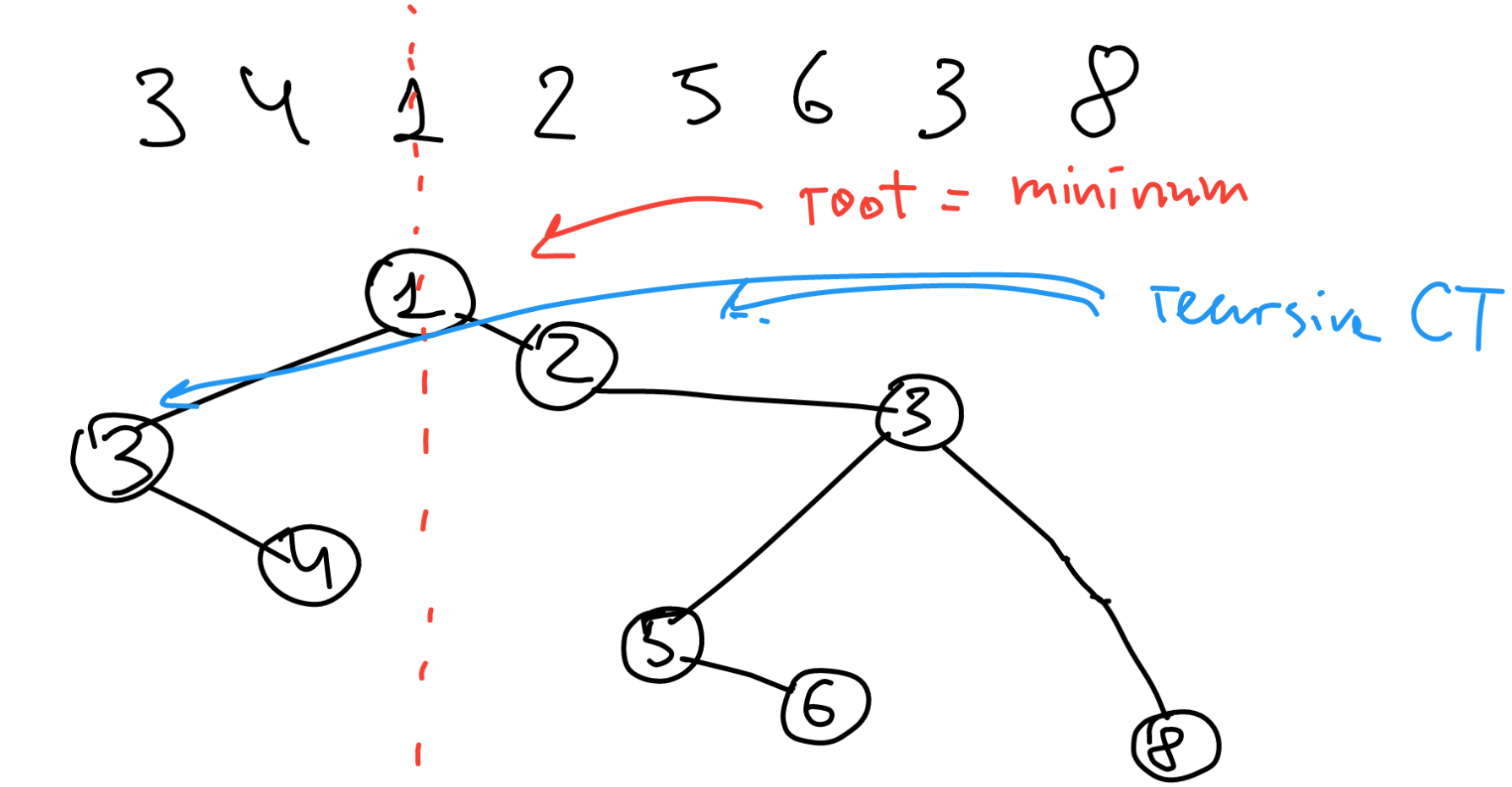
Euler Tour = $2n - 1$ numbers (+DFS)

$\Rightarrow$ LCA in $\langle O(N \log N), O(1) \rangle$ by RMQ

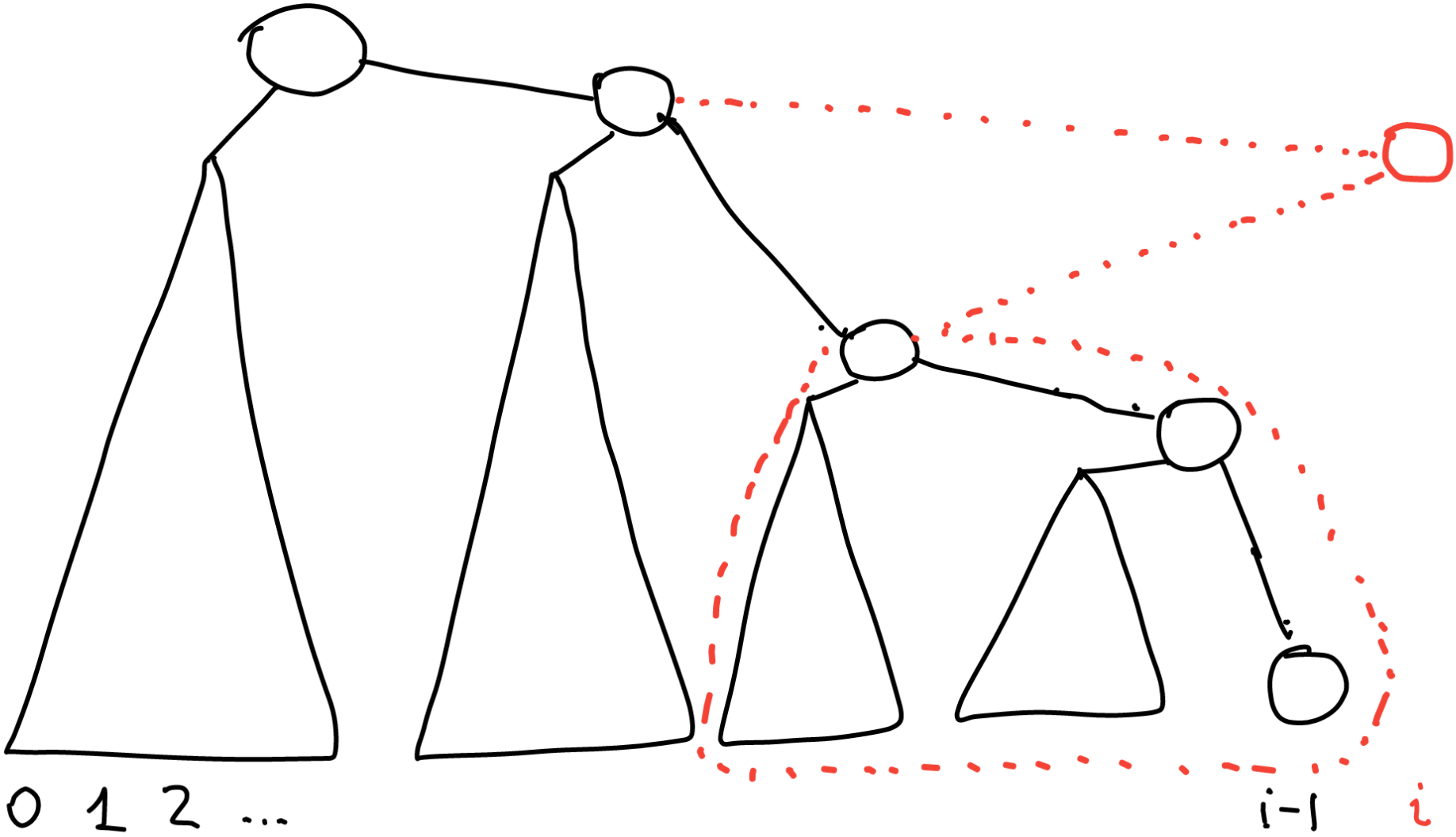
Can we do the opposite?

Reduce RMQ to LCA somehow?

Cartesian Tree:



Naïve construction of Cartesian Tree:
$O(N^2)$ for $1, 2, 3, \ldots, N-1, N$
Can be done in $O(n)$:



root, right[root], right[right[root]], ..., i-1

Stack

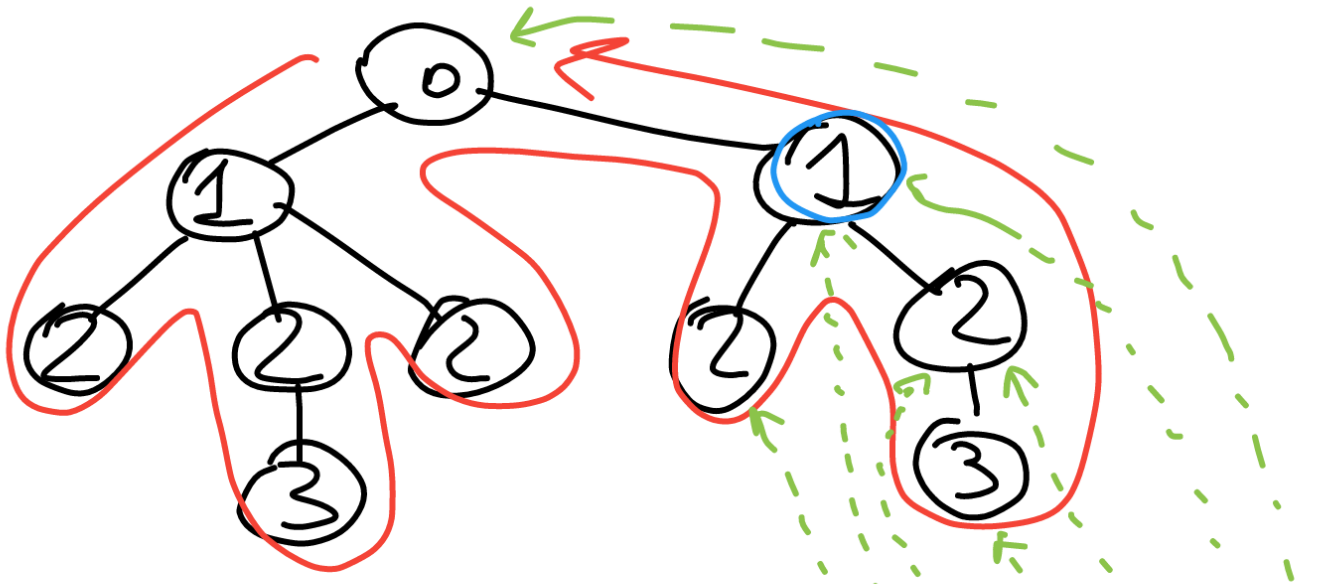
head

each vertex is added once and removed no more than once → $O(n)$ total time

⇒ we can reduce RMQ of N numbers to LCA of tree size N in $O(n)$.

How does it help?

Another Euler Tour : depth[n]



0 1 2 1 2 3 2 1 2 1 0 1 2 1 2 3 2 1 0

argmin RMQ : $RMQ[l, r) = k$

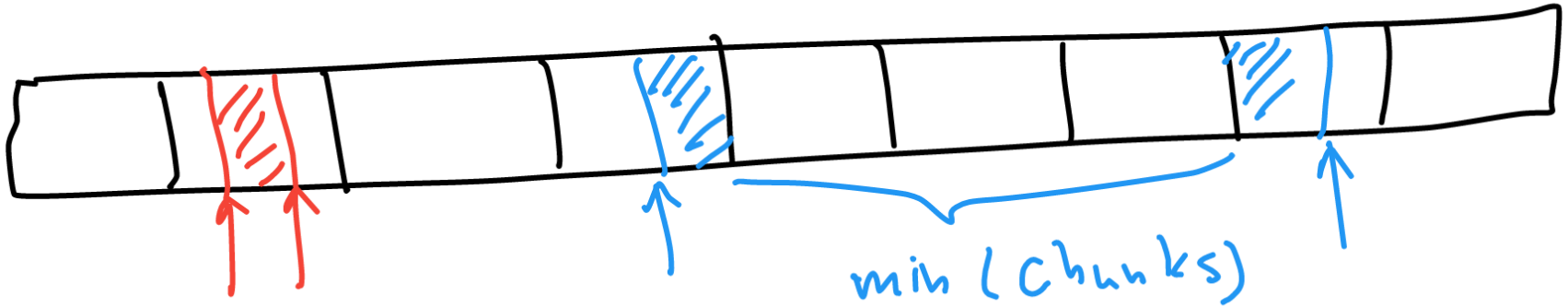
$a_k = \min\{a_l, a_{l+1}, \dots, a_{r-1}\}$

This gives reduction LCA → RMQ±1.

$|a_i - a_{i+1}| = 1$.

Split N into chunks of size $b = \frac{\log_2 N}{2}$

Build sparse table on min(chunks)



Need to solve for queries $\leq b$

"4 Russians trick":

$a_0, a_1, \dots, a_{b-1}$ Subtract a_0:

$0, \pm 1, \pm 1 \pm 1, \dots, \underbrace{\pm 1 \pm 1 \dots \pm 1}_{b-1 \text{ times}}$

encode -1 with "0", $+1$ with "1"

$\underbrace{0\,0\,1\,0\,1\,0 \dots 0}_{b \text{ bits}}$

$2^b = 2^{\frac{\log_2 w}{2}} \in O(\sqrt{w})$

different masks

position of min doesn't change!

Precale all queries:

$$O(\sqrt{n}) \cdot O(b^2) = O(\sqrt{n} \log^2 N) \in o(N)$$

Sparse table on chunks:

$$O\left(\frac{N}{b} \log \frac{N}{b}\right) = O\left(\frac{2N}{\log N} \cdot (\log N - \log b)\right)$$

$= O(N)$. Thus, RMQ±1 can be solved in $\langle O(n), O(1) \rangle$ time.

$$RMQ \xrightarrow{O(n)} LCA \xrightarrow{O(n)} RMQ \pm 1$$

PROFIT!!!