LOWEST COMMON ANCESTOR

Definition: The Range Minimum Problem is the following:

Preprocess: Array $A[1], \dots, A[n]$ of integers such that the following query can be answered in constant time:

Query: Given range $[i,j]$; $1 \leq i \leq j \leq n$

Return an index k , $i \leq k \leq j$

such that $A[k] \leq A[\ell] \quad \forall \ell = i, \dots, j$

Gabow, Bentley, Tarjan 84 - Gave linear-time preprocessing algorithm for problem.

Using LCA to solve Range Minimum:

Construct the Cartesian Tree of A.

Definition: The Cartesian Tree of A is recursively defined as follows:

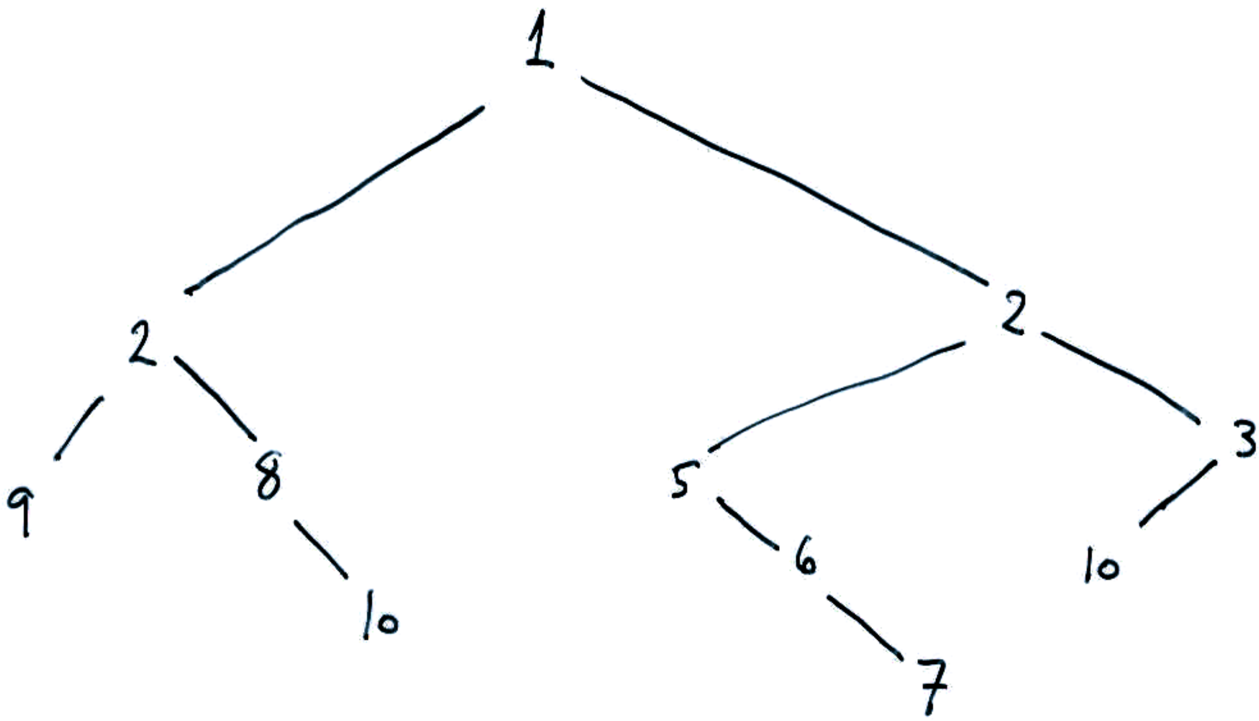
1) The root, r, of the cartesian tree is i, where $A[i]$ is a minimum of A.

2) The left son of r is the root of the cartesian tree of $A[1], \ldots, A[i-1]$.

3) The right son of $A[i]$ is the root of the cartesian tree of $A[i+1], \ldots, A[n]$.

EXAMPLE :

$A = 9 \quad 2 \quad 8 \quad 10 \quad 1 \quad 5 \quad 6 \quad 7 \quad 2 \quad 10 \quad 3$



Range minimum(i,j) = LCA(i,j)

in the

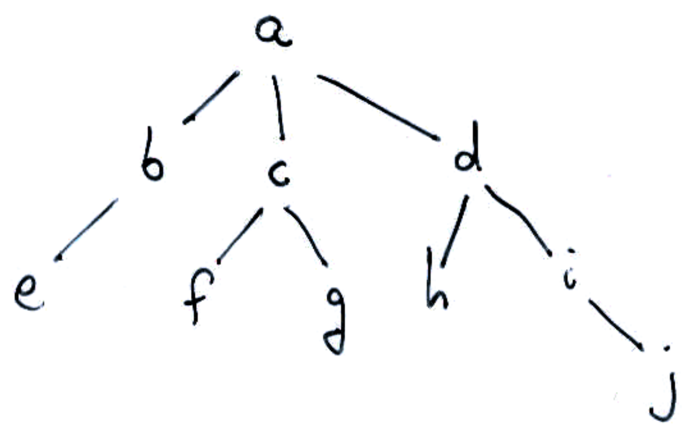
cartesian tree.

Jsing Range Minimum to solve LCA:

Given tree T let A be the euler tour of T

i.e. the elements of T as visited by a DFS of T.

EXAMPLE:



a b e b a c f c g c a d h d i j i d a

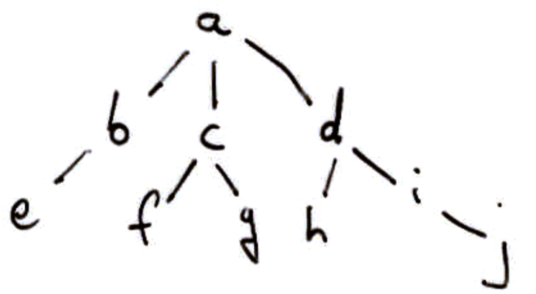
$|A| = 2(\#\text{ of edges of } T) + 1.$

Definitions:

$\ell(v)$ = Index of leftmost appearance of v in A.

$r(v)$ = Index of rightmost appearance of v in A.

EXAMPLE:



1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
a	b	e	b	a	c	f	c	g	c	a	d	h	d	i	j	i	d	a

	a	b	c	d	e	f	g	h	i	j
ℓ	1	2	6	12	3	7	9	13	15	16
r	19	4	10	18	3	7	9	13	17	16

Linear Time Construction of ℓ, r:

Method 1:

Scan A from left to right & write in ℓ first occurrence of each symbol.

Scan A from right to left & write in ℓ first occurrence of each symbol.

Method 2:

Note that

$\ell(v)$ is where $A[\ell(v)] = v$ and

$$\mathrm{LEVEL}(A[\ell(v)-1]) = \mathrm{LEVEL}(v) - 1.$$

$r(v)$ is where $A[r(v)] = v$ and

$$\mathrm{LEVEL}(A[r(v)+1]) = \mathrm{LEVEL}(v) - 1.$$

Key Trait:

Between $l(v)$ and $r(v)$ we have exactly the euler tour of the subtree rooted at v.

This trait leads to the following conclusions:

1. u is an ancestor of v iff $l(u) < l(v) < r(u)$

2. u and v are unrelated iff either $r(u) < l(v)$ or $r(v) < l(u)$.

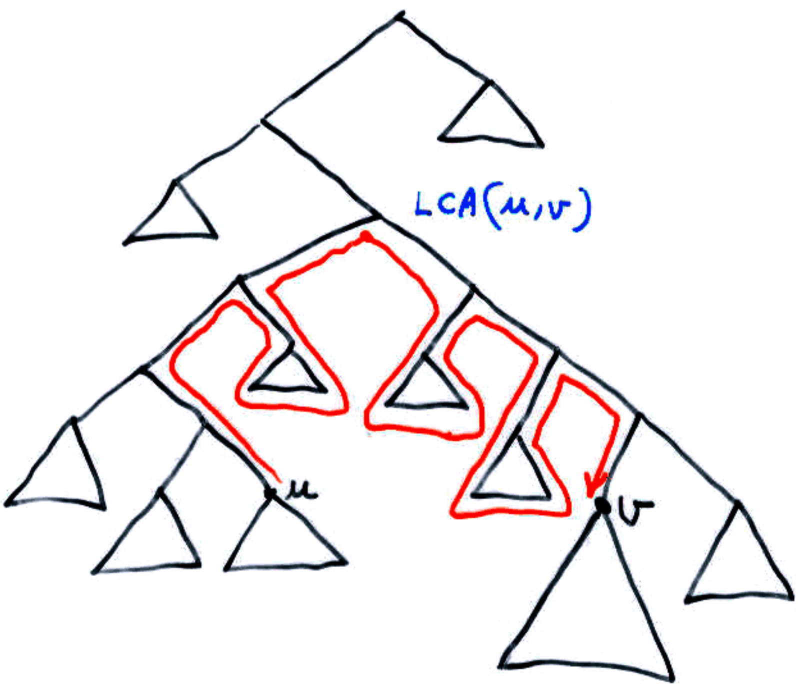
3. It is possible to find in constant time for given u, v if u is an ancestor of v.

4. Let u, v be unrelated with $r(u) < \ell(v)$.

then

$LCA(u,v) =$ vertex with minimal level in the interval $[r(u), \ell(v)]$ in A.

EXAMPLE:



Conclude: Construct array

$LEVEL(A) = LEVEL(A[1]), LEVEL(A[2]), \ldots$
$\ldots, LEVEL(A[n])$

and preprocess for range minimum queries

$LCA(u,v) =$ Range Minimum $(r(u), l(v))$
in $LEVEL(A)$.

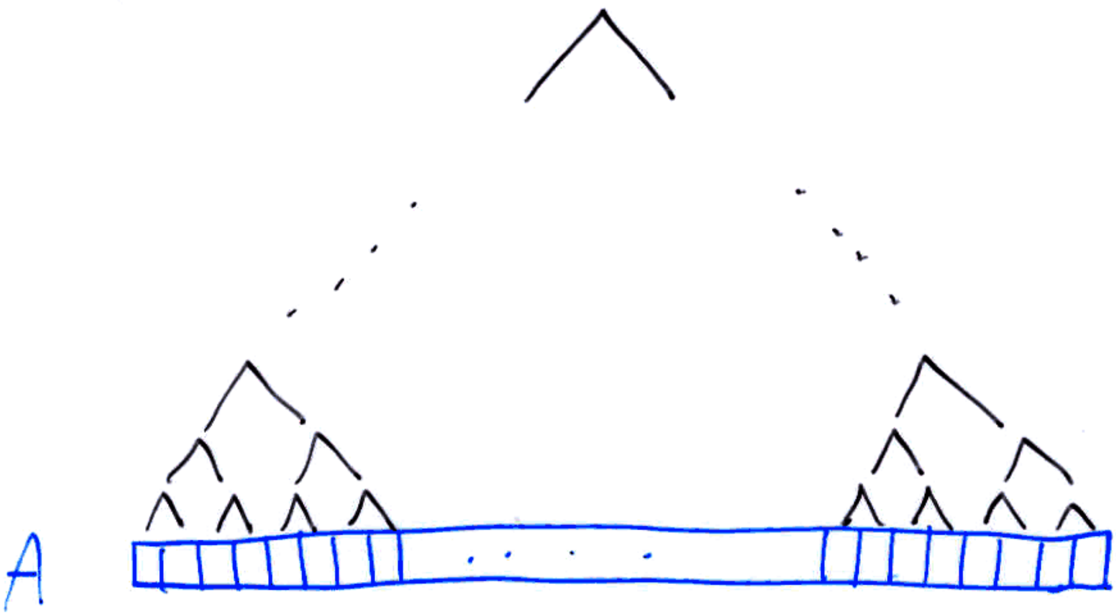
IMPORTANT FACT:

These range minimum queries are restricted in the sense that for any two adjacent numbers, $LEVEL(A[i])$, $LEVEL(A[i+1])$ the difference between them is always ± 1.

Range Minimum Algorithm with $O(n \log n)$ Preprocessing and $O(1)$ Query.

Consider array $A[1], \ldots, A[n]$

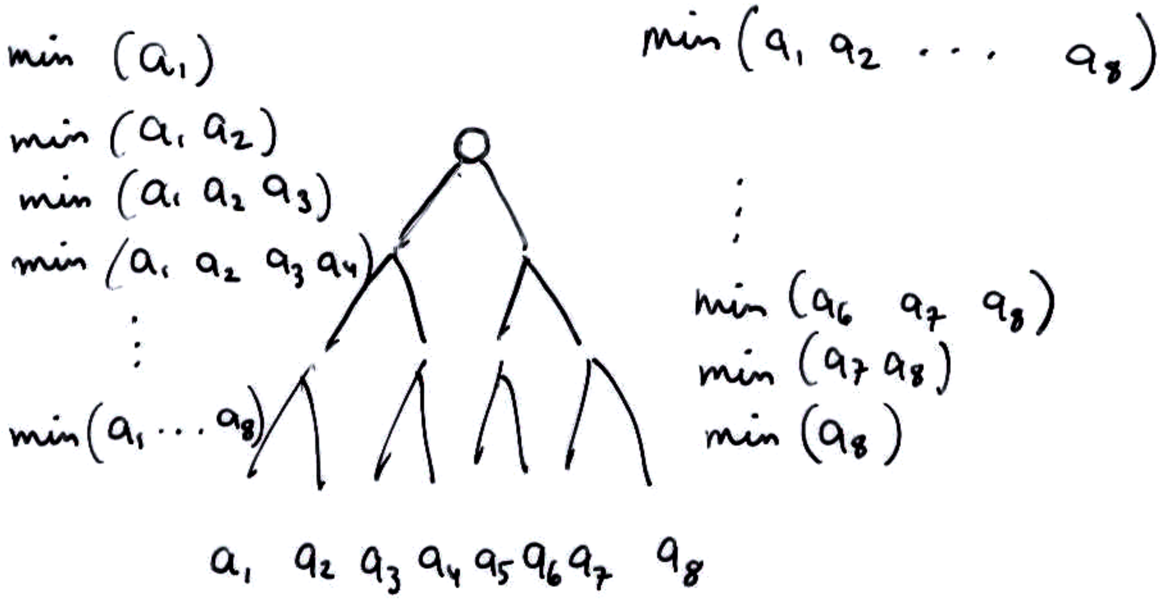
1. Construct on top of A a complete binary tree.



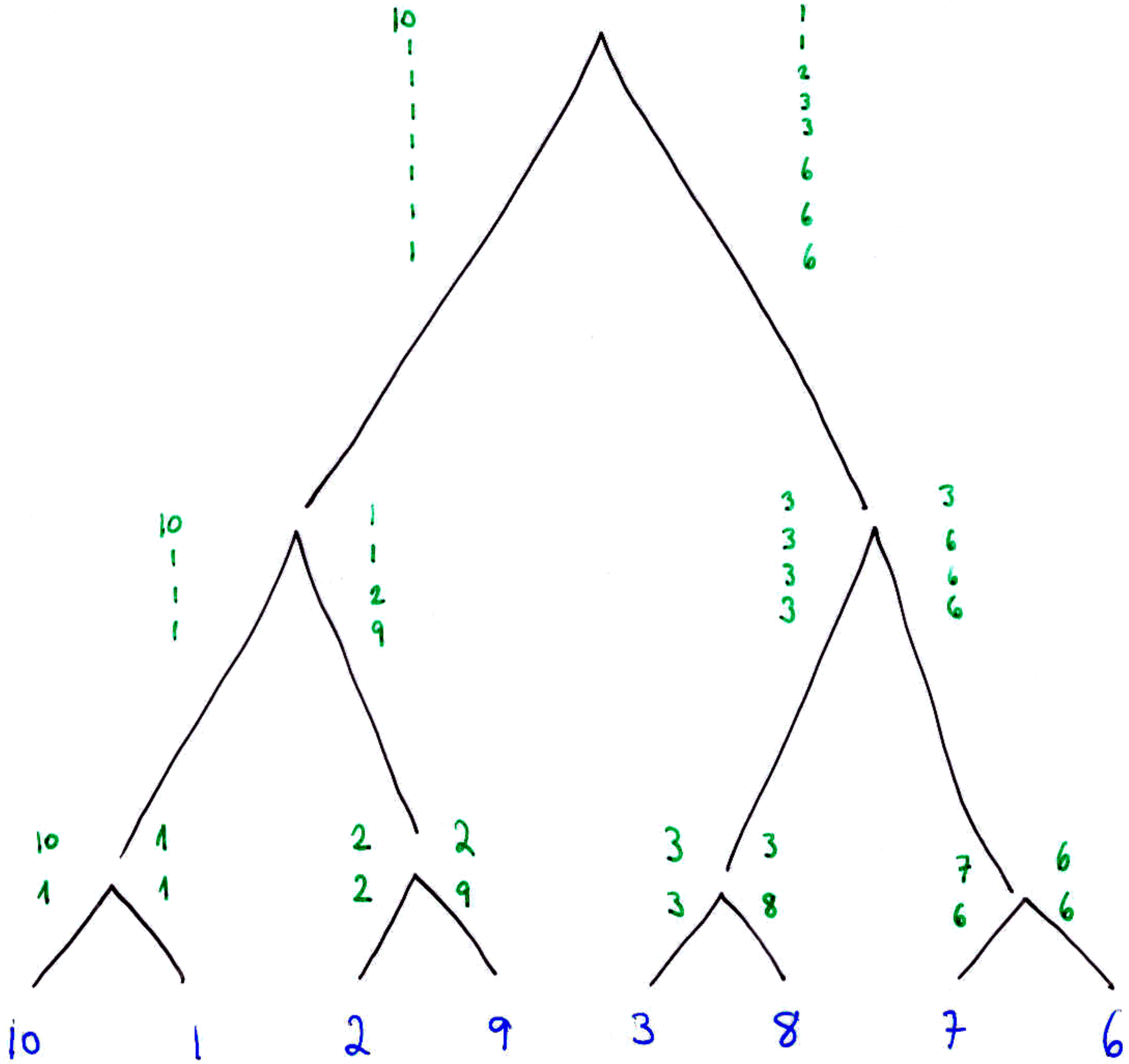
2. For every node in the tree consider all prefixes of its leaves and all suffixes of its leaves.

For each prefix write minimum

For each suffix write minimum.



EXAMPLE: (min prefixes on right
min suffixes on left)



Size of Tree:

level 0 : $2 \cdot n$

level 1 : $2 \cdot \frac{2n}{2}$

level 2 : $2 \cdot \frac{2^2 n}{2^2}$

$\vdots$

level $\log n$: $2 \cdot \frac{2^{\log n} \cdot n}{2^{\log n}}$

Total: $2n \log n$

Time to compute: $O(n \log n)$

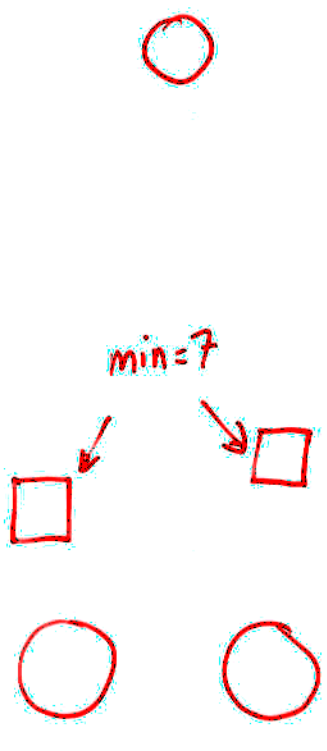
Query Computation: Range Minimum(i,j)

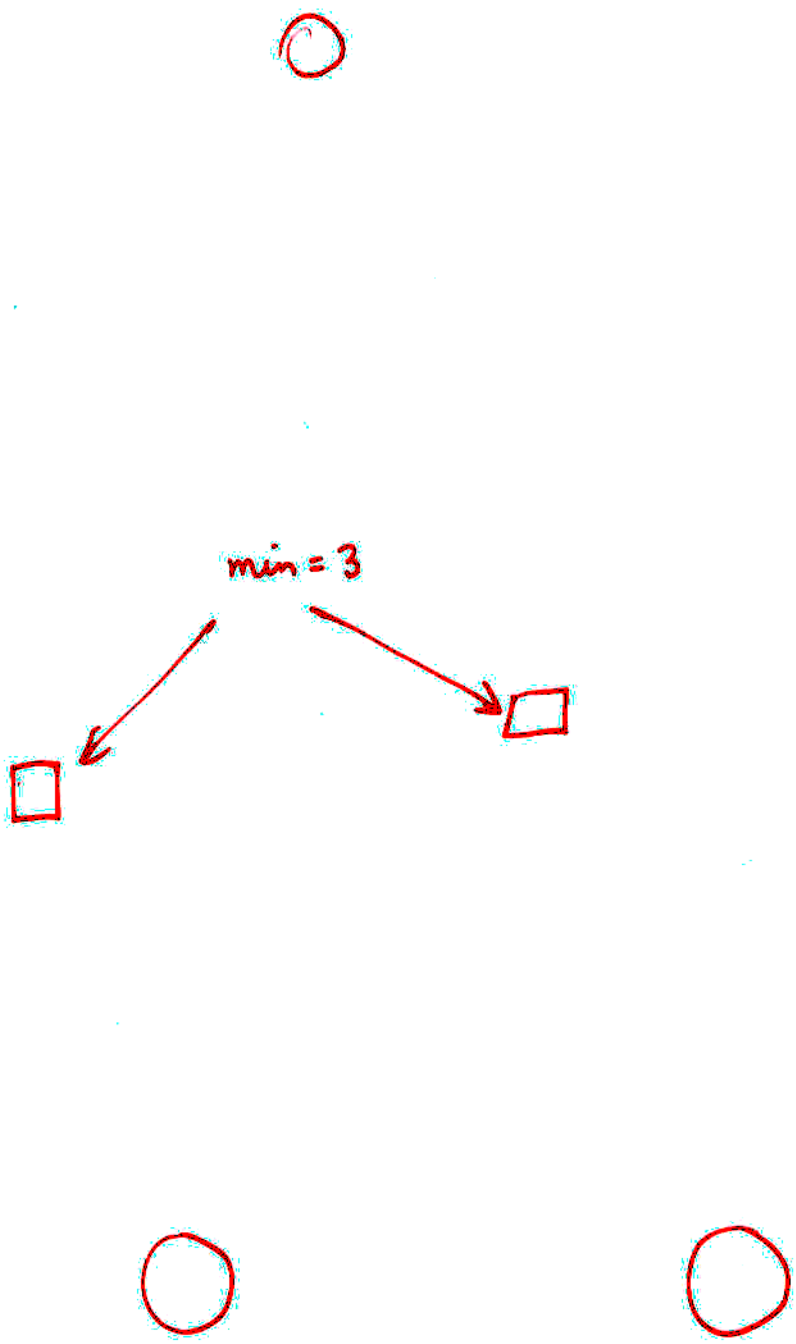
1) Find lowest common ancestor of i and j in binary tree. (we'll soon see how to do it in constant time.)

2) Goto left son and find minimum in the suffix starting at i.

3) Goto right son and find minimum in the prefix ending at j.

4) Take minimum of the two.

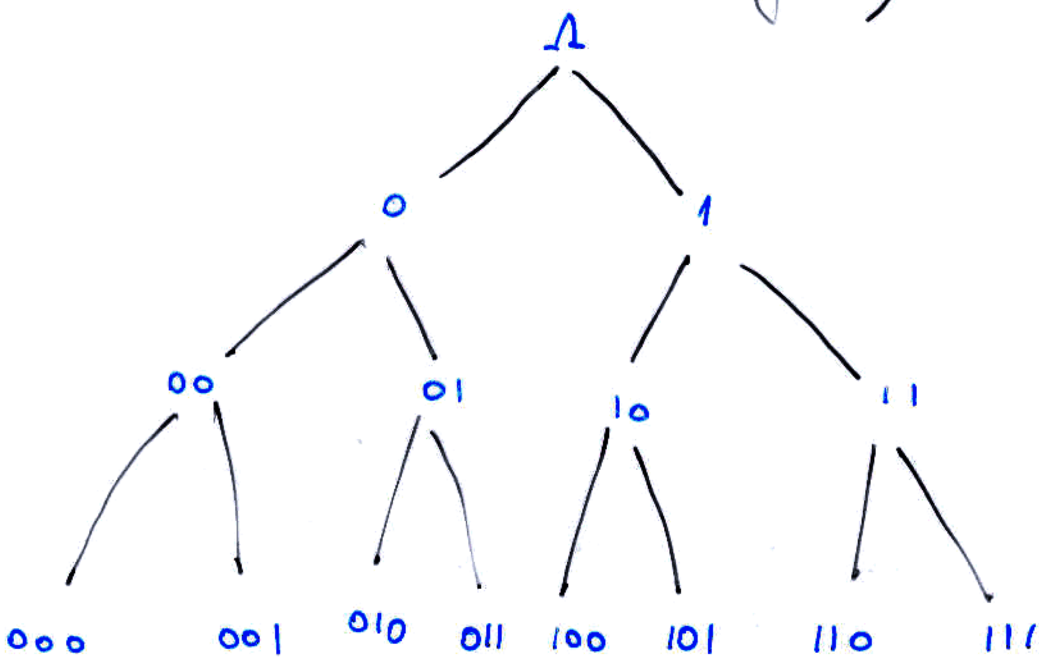

min=7


min = 3

How do we find LCA in binary tree in constant time?

Model: Word arithmetic for word of size $O(\log n)$ in constant time.

Denote nodes of binary tree by their path from root (0 - left, 1 - right)



LCA(i,j) = longest common prefix

Find it by logical & arithmetic operations on i and j:

EXAMPLE:

$$\begin{array}{r} 010 \\ \text{XOR} \quad 000 \\ \hline 010 \end{array}$$

XOR gives us 0's in msbits.
The first 1 denotes msbit that is not equal.

Shift i to right $\lfloor \log(\text{XOR}(i,j)) \rfloor$ bits and truncate $\lfloor \log(\text{XOR}(i,j)) \rfloor$ bits.

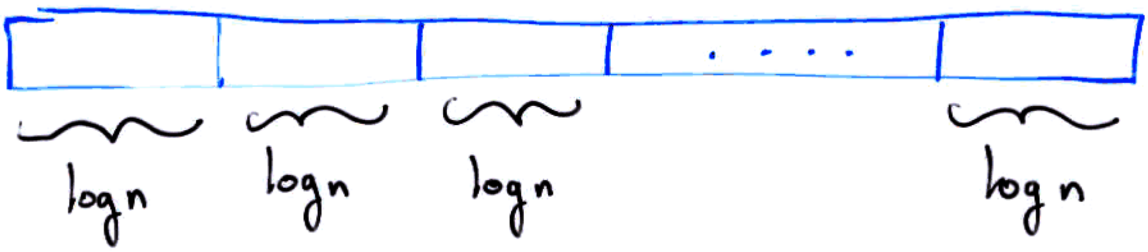
A similar constant time mask allows finding index in the min prefix and min suffix vectors of the sons of the LCA

Remaining Problem:

Getting rid of $\log n$ factor.

IDEA:

- Divide array A into $\frac{n}{\log n}$ substrings of size $\log n$



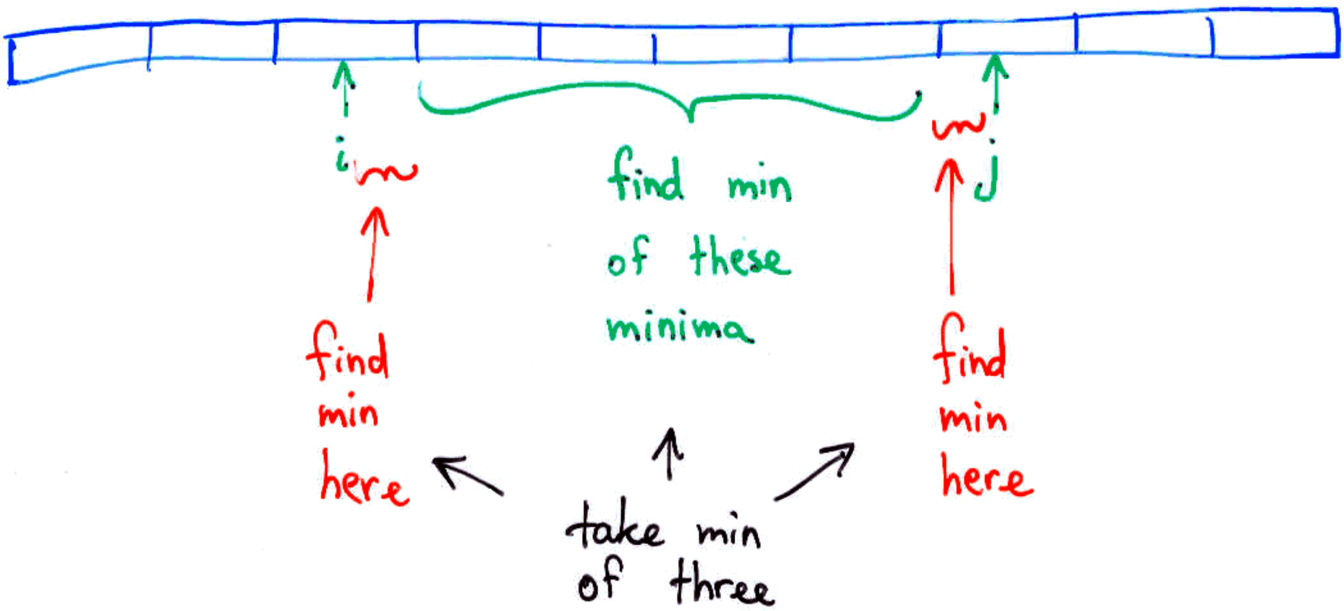
- For each of the $\frac{n}{\log n}$ subarrays find minimum.

- Construct full binary tree for finding range minima in the $\frac{n}{\log n}$ numbers (minima of the subarrays).

Time & Space: $O\left(\frac{n}{\log n} \log\left(\frac{n}{\log n}\right)\right) = O(n$

Query Processing:

If i and j in different subarrays:



How do we find

?

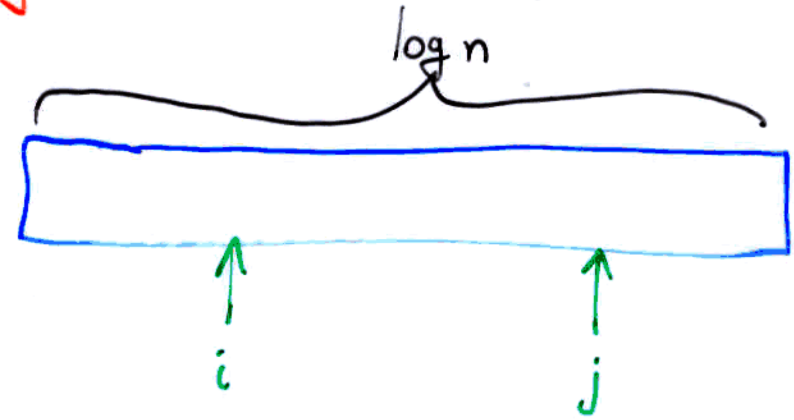
In preprocessing write min of every prefix and suffix of each subarray.

Put in table.

Time: $O(n)$ preprocessing.

constant time table lookup.

Remaining Case: i & j in same subarray



1) Use same technique as for n, by dividing into $\frac{\log n}{\log \log n}$ pieces, each of size $\log \log n$.

2) Remaining Problem: Finding Range Minimum (i, j) within subarray of size $\log \log n$.

Now use important fact that our range minimum queries are on a restricted array.

We can normalize every $\log\log n$ subarray to one where the first level is 0.

We can consider the array to be:

e_1	e_2	$\cdots$	e_i	$\cdots$	$e_{\log\log n - 1}$	$e_{\log\log n}$

where $e_i \in \{-1, 1\}$

$i = 1, \ldots, \log\log n.$

Two subarrays with an equal normalized template have same results to range minimum queries (in index).

Given a template and initial level, we can construct initial subarray.

CRUCIAL FACT:

While there are $\frac{n}{\log\log n}$ subarrays,

there are only $2^{\log\log n} = \log n$

different templates.

- For each template there are

$$(\log\log n)(\log\log n - 1)$$

different possibilities of (i,j).

- For each template construct a table of size $O((\log\log n)^2)$ giving the minimum in the range (i,j) for that template, $\forall\, i,j$.

Total Table size: $O\left(\log n \, (\log \log n)^2\right)$

For every (i,j) in subarray s,
Look up at the (i,j) location
in the table of s's template.
This gives the index of the
minimum in the range (i,j) of
the subarray in constant time.