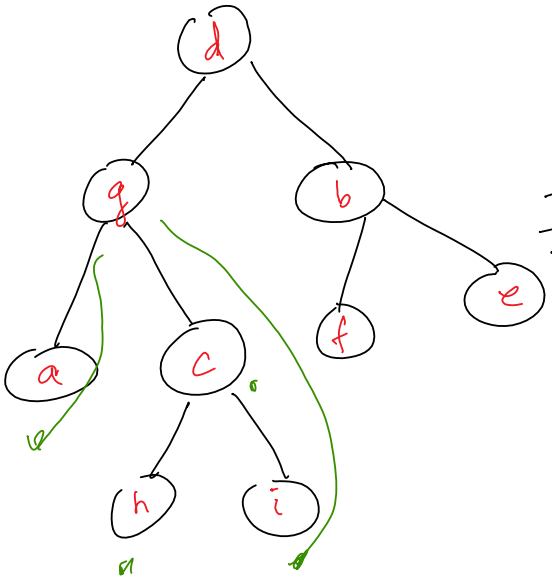
LCA algorithm

Tuesday, March 27, 2018 11:17 AM

The Lowest Common Ancestor problem



Problem: Finding the **lowest common ancestor** of any two given nodes in the tree T.

Input: Tree T with n nodes
Nodes i & j

Output: lca(i, j)

Goal: Answer lca queries of the kind lca(i, j) in O(1) time

Examples:

lca(a, b) = d
lca(i, a) = g
lca(h, c) = c
lca(b, e) = b

For solving the LCA problem, we will visit Bender-Farach's algorithm (from the late '90's).

Bender, Michael A., and Martin Farach-Colton. "The LCA problem revisited." In *Latin American Symposium on Theoretical Informatics*, pp. 88-94. Springer, Berlin, Heidelberg, 2000.

http://spin2013.cs.sunysb.edu/~bender/newpub/BenderFa00-lca.pdf

LCA algorithm

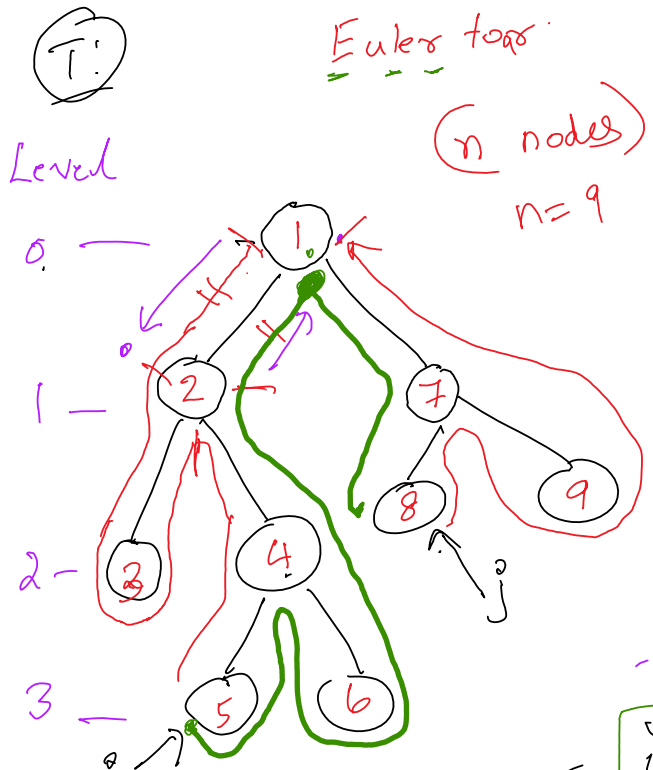
Tuesday, March 27, 2018 11:18 AM

Preprocessing $T \xrightarrow{O(n)} \{E, L, R\} \Rightarrow LCA(i,j)$ $O(1)$ time

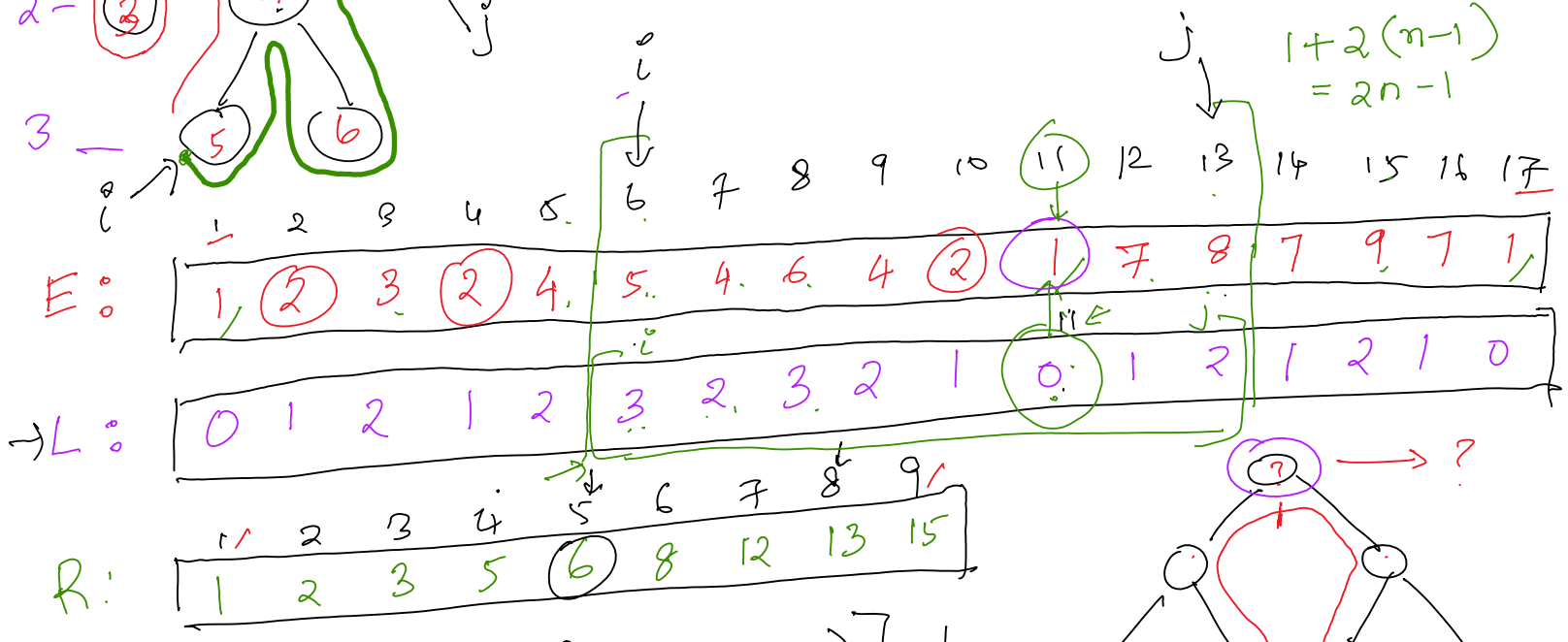
Bender - Farach (2000)

$O(n)$.

1) Euler tour of T
   - → E : Euler tour ← $2n-1$
2) → L : Level array ← $2n-1$
3) → R : Reverse array ← n
   - R[i] ← 1st occurrence of node i in E.



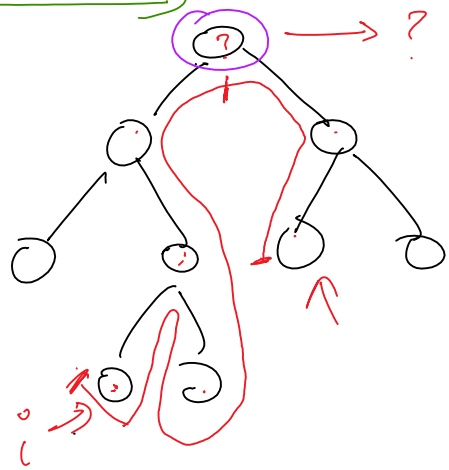
E.g: lca (4, 8)



$lca(i,j) = E[RMQ_L(R[i], R[j])] = 1$

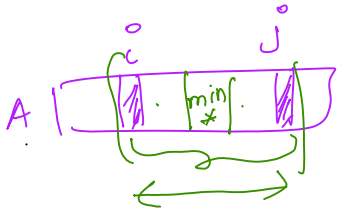
$O(1)$ if $RMQ_L(R[i], R[j])$ is $O(1)$ time

6 13 11



RMQ: Range Minimum Query

$RMQ_A(i,j)$ ← The index of the minimum $A[i \cdots j]$



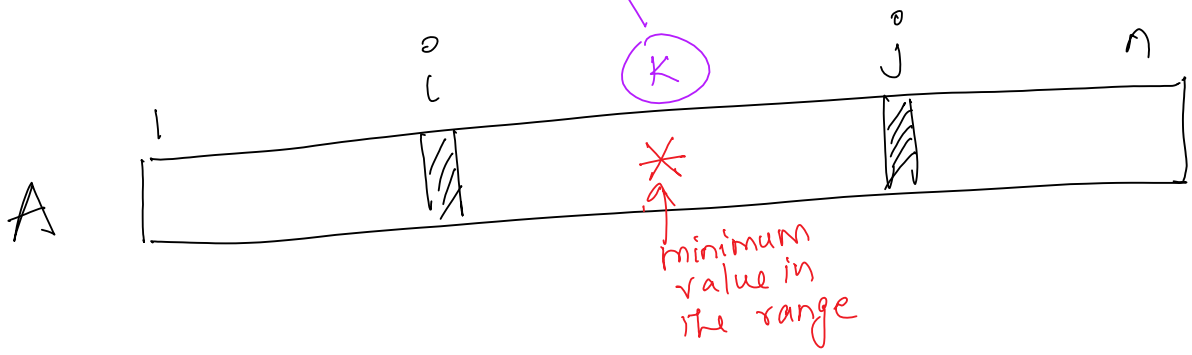
RMQ function

Wednesday, March 25, 2020 10:51 AM

$RMQ_A(i,j)$:

Given as **input**, an array $A[1..n]$ (of ints) and two indices i & j

Return as **output**, the location of the minimum in the array interval $A[i..j]$



Note: It is trivial to be able to answer $RMQ_A(i,j)$ in $O(n)$ time.

Q) Is it possible to answer any RMQ query in $O(1)$ time?

B-prep(A[1..n]): From array A[1..n], generate a set of O(lgn) "B-arrays"

LCA algorithm B-prep(A) ⇒ $\{B_0, B_1, B_2, \cdots B_{\lg n}\}$

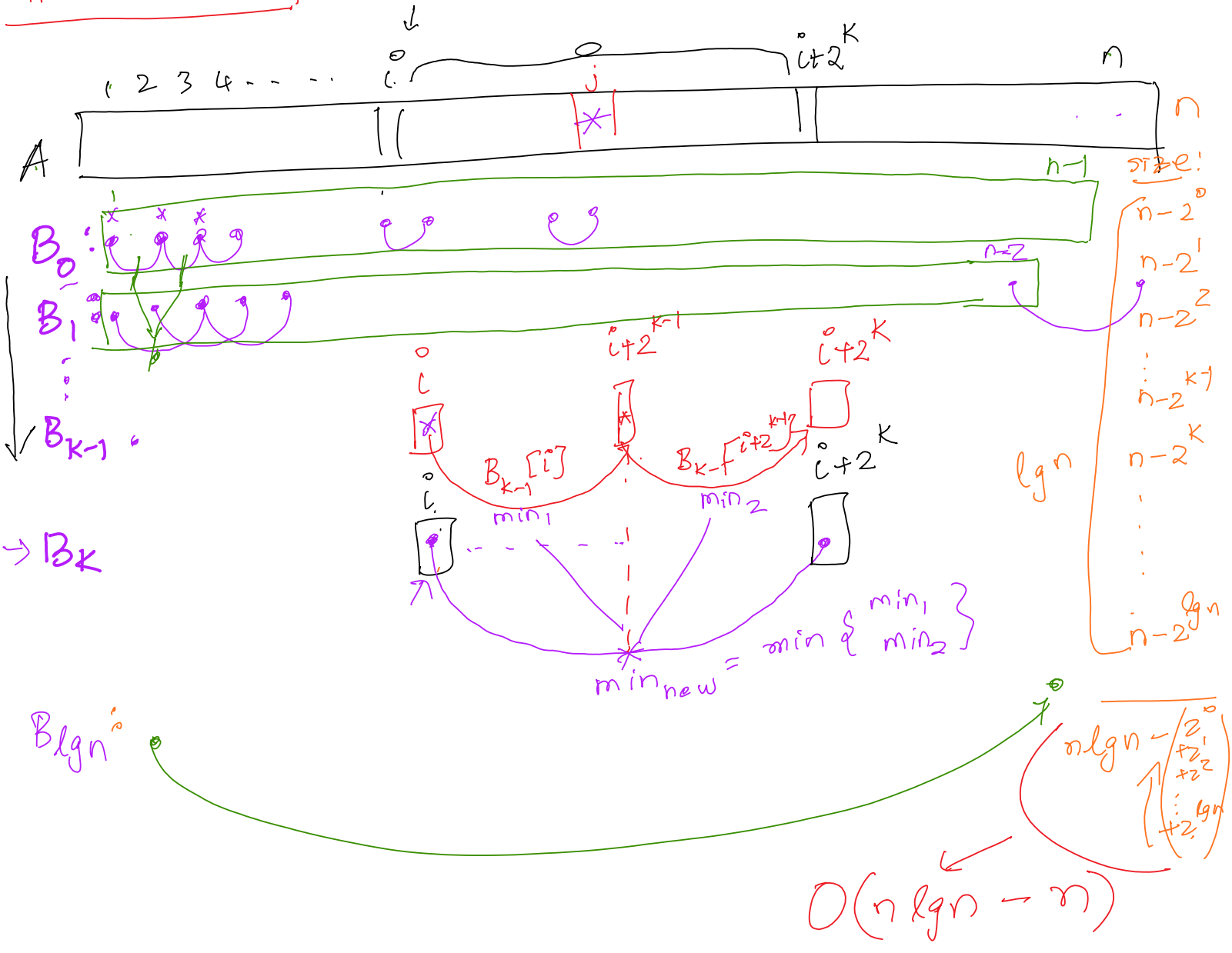
Tuesday, March 27, 2018 11:18 AM

s.t.

$B_K[i] \leftarrow$ minloc(A[i .. $i+2^K$])

$= j$

Let **minloc()**:
→ the location/index of the minimum



B-prep()

Sum of all the sizes of all B-arrays = $O(n \lg n)$

A[1..n] ⇒ $\{B_0, B_1, \cdots B_{\lg n}\}$
$O(n \lg n)$

⇒ Time to construct all B-array = $O(n \lg n)$

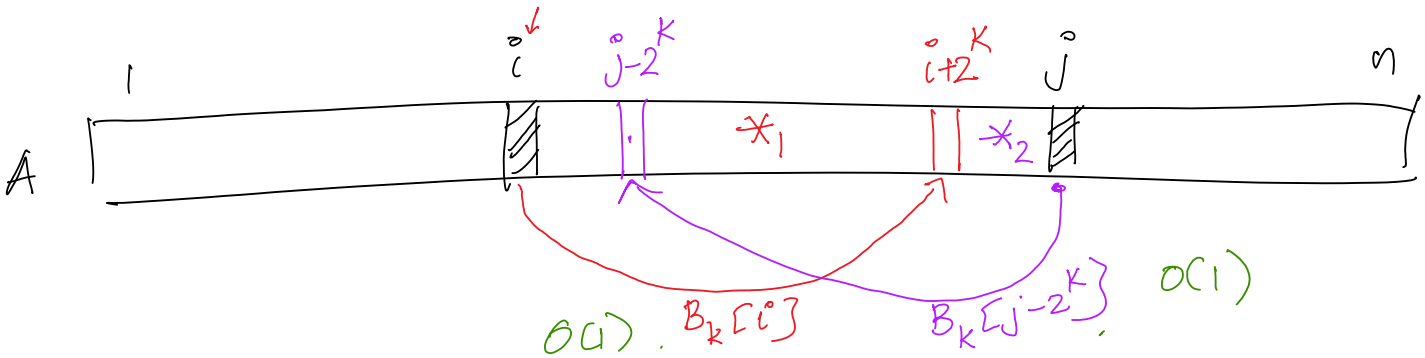
LCA algorithm

Tuesday, March 27, 2018 11:18 AM

Q) How to answer $RMQ_A(i,j)$ using the B-arrays?

A) Step 1) Preprocessing $A[1..n] \xrightarrow[O(n \lg n)]{}$ B-arrays $\{B_0, B_1, \dots B_{\lg n}\}$

Step 2) Querying $RMQ_A(i,j)$:



$O(1)$ $B_k[i]$ $B_k[j-2^k]$ $O(1)$

$*_{(i,j)} \leftarrow \text{minloc}\{*_1, *_2\}$ $O(1)$

$K = ?$ $\quad 2^K \le j - i$

$K \le \lg(j-i)$

$O(1) \rightarrow K = \lfloor \lg(j-i) \rfloor$

$A[1..n] \Rightarrow$ B-prep(A) $\Rightarrow \{B_0, B_1, \dots, B_{\lg n}\} \Rightarrow$ Query time

$RMQ_A(i,j)$ ⇓ lookup B_K values & compare

$O(n \lg n)$ time ↗ ? ↘ $O(n)$ time

$O(1)$ time

$O(1)$ time

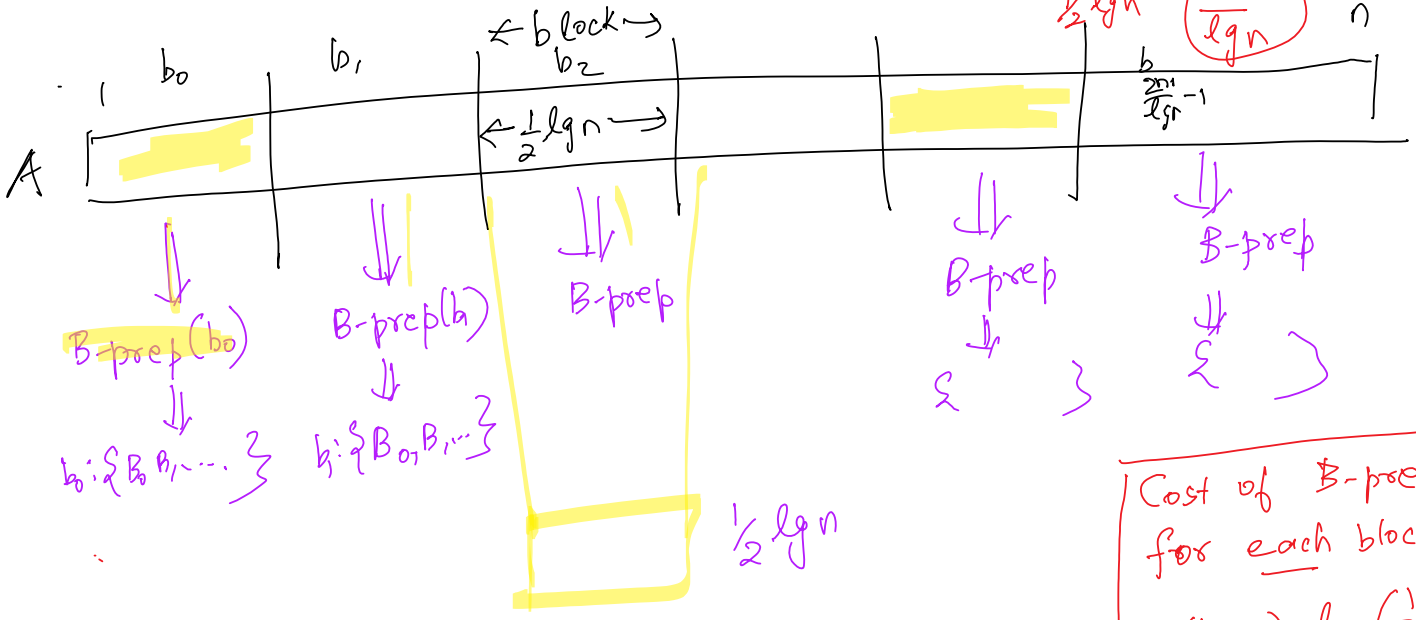
LCA algorithm

Tuesday, March 27, 2018 11:18 AM

Q) Can we bring down the cost of preprocessing from $O(n \lg n) \Longrightarrow O(n)$ possibly?

A) Let us divide $A[1..n]$ into "blocks" of size $\frac{1}{2}\lg n$ each.

$\Rightarrow$ #blocks $= \frac{n}{\frac{1}{2}\lg n} = \frac{2n}{\lg n}$



Cost of B-prep() for each block

$= \left(\frac{1}{2}\lg n\right) \lg\left(\frac{1}{2}\lg n\right)$

$= O\left(\frac{1}{2}\lg n \lg(\lg \sqrt{n})\right)$

#blocks $= \frac{2n}{\lg n}$

Total cost

$= \frac{2n}{\lg n} \times \frac{1}{2}\lg n \lg \lg \sqrt{n}$

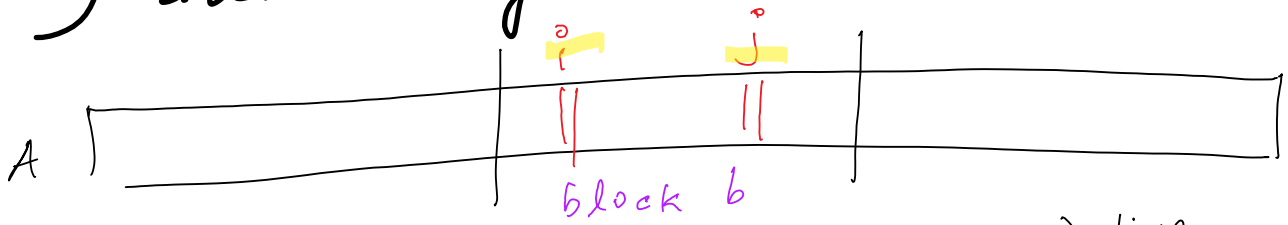
$= O(n \lg \lg \sqrt{n})$

LCA algorithm

Tuesday, March 27, 2018 11:18 AM

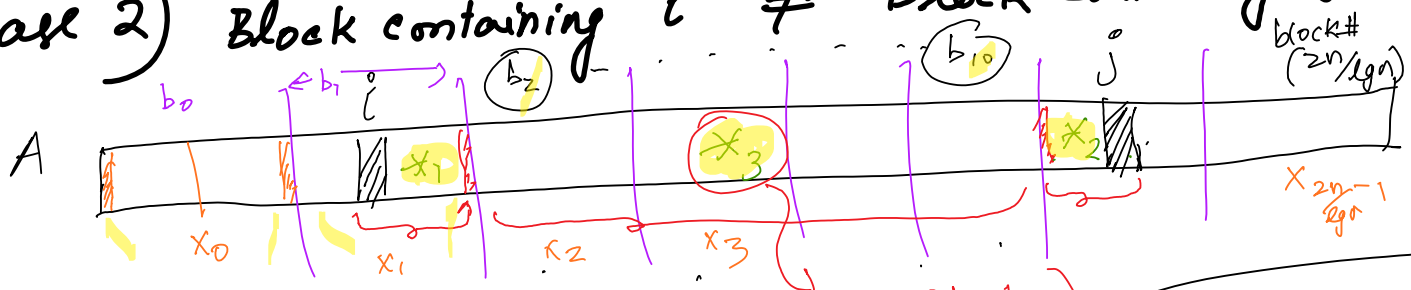
Two cases for RMQ(i,j) queries:

Case 1) Block containing i° = Block containing j°



$\Rightarrow$ RMQ(i°,j°) can be answered in O(1) time using B-arrays for block b

Case 2) Block containing i° ≠ Block containing j°

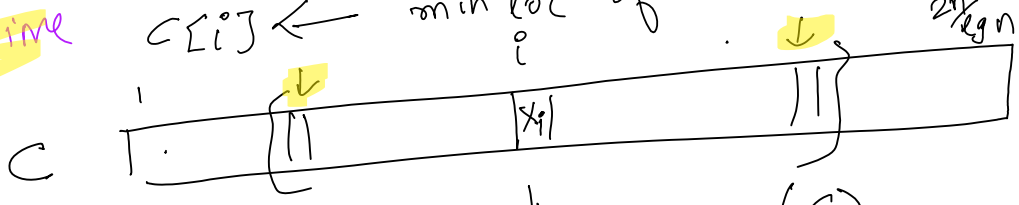


$\Rightarrow$ To compute $*_3$:

1) Build an array $C[1 \ldots \frac{2n}{\lg n}]$ s.t

$C[i] \leftarrow$ min loc of block b_i

$O(\frac{2n}{\lg n})$ time



2) B-prep(C):

$\Downarrow$ B-prep(C)

generate all B-arrays for C in O(1) time.

$\Rightarrow$ RMQ queries on C array

cost of B-prep(C) = $\left(\frac{2n}{\lg n}\right) \lg\left(\frac{2n}{\lg n}\right) = O\left(\frac{2n}{\lg n} \lg n\right)$

$= O(n)$ time

$RMQ_A(i,j) = \min_{\text{loc}} \{*_1, *_2, *_3\}$

$\Rightarrow O(1)$ time

LCA algorithm

Tuesday, March 27, 2018 11:18 AM

Overall Preprocessing Algorithm so far:

Input: $A[1..n]$

Time Complex:

PREPROCESSING

1) Partition into $\frac{2n}{\lg n}$ blocks

1a) B-prep (each block separately) } $O(n \lg \lg n)$

2) Generate C-array

2b) B-prep(C) } $O(n)$

3) Answer $RMQ_A(i,j)$ queries } $O(1)$ time

$\Rightarrow$ $A[1..n]$ (input) $\Rightarrow$ [Preprocessing] $\Rightarrow$ [RMQ queries] $\circlearrowright (i,j)$

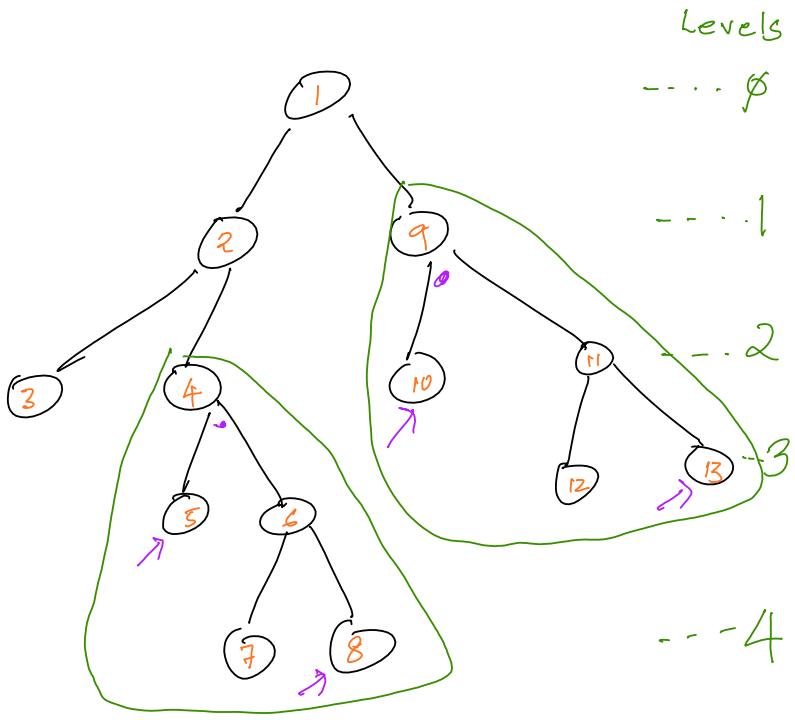
Preprocessing: $O(n \lg \lg n)$

RMQ queries: $O(1)$ time per query

Q) Can we make preprocessing time $O(n)$?

Idea

Friday, March 27, 2020 10:34 AM



E array:

1 2 3 2 4 5 4 6 7 6 8 6 4 2 1 9 10 9 11 12 11 13 11 9 1

L array:

0 1 2 1 2 3 2 3 4 3 4 3 2 1 0 1 2 1 2 3 2 3 2 1 0

B-prep

0 +1 -1 +1 +1 -1 +1 -1 -1

B-prep X

0 +1 -1 +1 +1 -1 +1 -1 -1

$(\frac{1}{2}\lg n - 1)$ bits

block

$\frac{1}{2}\lg n - 1$

0 | ±1 | ±1 | ±1 | . . . . | ±1

$\frac{2n}{\lg n}$ blocks $\geq \sqrt{n}$

#total distinct fingerprints $= 2^{(\frac{1}{2}\lg n - 1)} = 2^{\lg_2 \sqrt{n} - 1}$

$\leq \sqrt{n} \times (\frac{1}{2}\lg n)$ lglg$\sqrt{n}$ $= \frac{2^{\lg \sqrt{n}}}{2} = \frac{\sqrt{n}}{2}$

$= O(\sqrt{n})$

Cost of B-prep all distinct fingerprints $= O(n)$ time $= O(\sqrt{n} \times \sqrt{n})$

LCA algorithm

Tuesday, March 27, 2018 11:19 AM

Key Observation: Since the target array for preprocessing is the **level (L)** array, we can take advantage of the following property of L:

⇒ Every pair of consecutive elements in L differs by 1. // because of the Euler tour property

⇒ Each block generated from the L-array can be represented as a +1/-1 array (offset by the first element)

e.g., block X:

4	5	6	5	4	3	4	5	4	5

⇒

0	+1	+1	-1	-1	-1	+1	+1	-1	+1

⇒ let us call this pattern the block's "fingerprint"

Now, two blocks may look different but may have the same fingerprint.

e.g., block Y:

8	9	10	9	8	7	8	9	8	9

⇒

0	+1	+1	-1	-1	-1	+1	+1	-1	+1

⇒ same as block X's fingerprint

Two blocks with the same fingerprint will have their minimums located at the same index. (as shown highlighted above).

LCA algorithm

Tuesday, March 27, 2018 11:19 AM

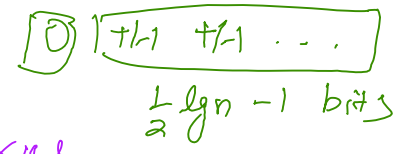
Idea: Instead of B-prep() each block of L separately, (a) Identify and bin blocks by their fingerprints

(b) B-prep() all the fingerprints.

Q) What is the maximum number of distinct fingerprints possible for the blocks derived from an array of size n?

Each block is of size $\left(\frac{1}{2}\lg n\right)$

$\Rightarrow$ Each block represents a binary pattern $(+/-1)$ of length $\left(\left(\frac{1}{2}\lg n\right)-1\right)$



$\Rightarrow$ Max # distinct binary patterns

$$= 2^{\frac{1}{2}\lg n - 1} = \frac{2^{\lg \sqrt{n}}}{2} = \frac{\sqrt{n}}{2}$$

$\therefore$ Cost of B-prep all distinct block fingerprints

$$= O\left(\frac{\sqrt{n}}{2} \times \left(\frac{1}{2}\lg n\right) \log\left(\frac{1}{2}\lg n\right)\right)$$

$$= O\left(\sqrt{n} \lg n \lg(\lg \sqrt{n})\right)$$

$$= O(n)$$

LCA algorithm

Tuesday, March 27, 2018 11:18 AM

Overall Algorithm :

Input: $A[1..n]$

Time Complex:

PREPROCESSING

1) Partition into $\frac{2n}{\lg n}$ blocks

1a) B-prep (each block fingerprint separately) } $O(n)$

2) Generate C-array

2b) B-prep(C) } $O(n)$

3) Answer $RMQ_A(i,j)$ queries } $O(1)$ time

$\Rightarrow$ $A[1..n]$ (input) $\Rightarrow$ [Preprocessing] $\Rightarrow$ [RMQ queries] $y(i,j)$

$\Theta(n)$

$O(1)$ time per query

LCA algorithm

Saturday, March 31, 2018 10:57 PM

Summary of the main algorithmic results:

	Time complexity: Preprocessing	Querying
$RMQ_A(i,j)$ for an arbitrary array of size n	$O(n \lg \lg \sqrt{n})$	$O(1)$
$RMQ_A(i,j)$ for a binary array of size n	$O(n)$	$O(1)$
$lca(i,j)$ for a tree with n nodes	$O(n)$	$O(1)$
$lcp(suff_i, suff_j)$ for a string of size n	$O(n)$	$O(1)$