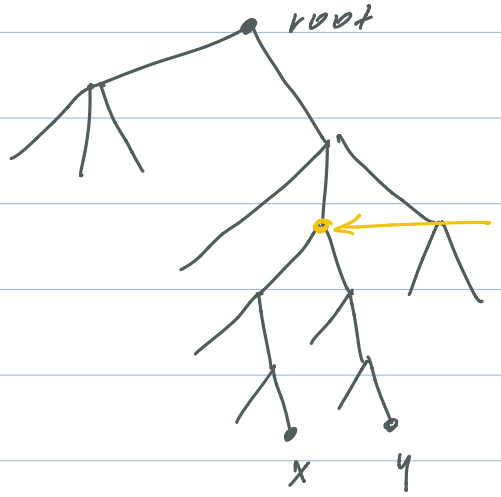
CS124 Lecture 23 Wed, April 22, 2020

~ Least Common Ancestor and Min Range Query ~

Least Common Ancestor



O(depth of tree)

create data structure to answer queries? O(1) time

Costs: preprocessing time, query time, space

no preware	O(1)	O(depth)	O(n)
precompute all	$O(n^2 \cdot depth)$	O(1)	$O(n^2)$
	O(n)	O(1)	O(n)

LCA → Range Minimum Query

RMA: array A of n numbers — naive solution: O(j-i) query

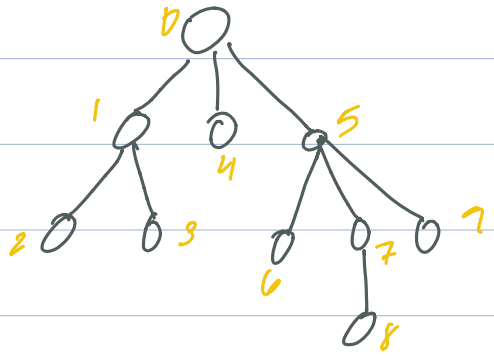
input indexes i, j

return index of min A[i], A[i+1], ..., A[j]

DP solution: $RMQ(i,j) = A^{-1}[\min[RMQ(i,j-1), A[j]]]$

$O(n^2)$ preprocessing O(1) query $O(n^2)$ space

Need linear-time reduction:



DFS from root and label

V_i: 0 1 2 1 3 1 0 4 0 5 6 5 7 5 8 7 5 9 5 0

→ $O(n)$ since visit each edge twice

n nodes → $n-1$ edges

so V is $2n-1$ numbers

RMQ structure over L [L_i: level array

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

R_i: first time I see that node

0 1 2 4 7 9 10 12 13 16

$LCA(u,v) = V[RMQ(R(u), R(v))]$

key idea: during DFS, must traverse LCA before hitting second node

Preprocessing, Space → $O(n \log n)$

keep track of subset of answers

$O(n^2)$ space → store $RMQ(i,j)$ $\forall i,j$

$O(n \log n)$ space → store $RMQ[i, i+2^k)$

$k = 0, 1, 2, \ldots, \log n$

$$RMQ(i, i+2^k) = A^{-1}\left[\min\left(A\left(RMQ[i, i+2^{k-1})\right), A\left(RMQ[i+2^{k-1}+1, i+2^k)\right)\right]$$

$RMQ[i, i+313)$ = min value of two overlapping ranges

Recursion: PP, space $\rightarrow O(n)$

Split A:

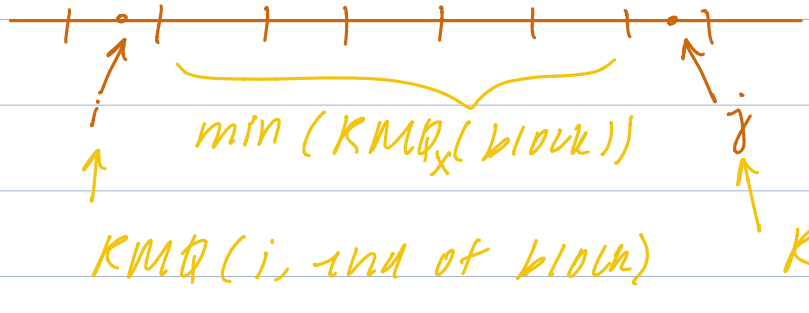
X[1, ... 2n/logn], where X[i] = min element in ith block of A

Y[1, ... 2n/logn], position where X[i] occurs

Preprocessing: RMQ structures on each block and on X

RMQ (i, j)

- if same block, easy!
- if not same block



minimum = minimum of these 3 possibilities

X is of size $2n/\log n$

Time/span: $\frac{2n}{\log n} \cdot \frac{2n}{\log n} \rightarrow O(n)$

Splitting A: $\frac{2n}{\log n}$ blocks of size $\log n / 2$

RMQ on one block $\rightarrow \frac{\log n}{2} \cdot \log\left(\frac{\log n}{2}\right) =$

$O(\log n \times \log \log n)$

Total: $O(n \times \log \log n)$

Keep recursing to $O(n \times \log\log\log\ldots\log n)$

Down to $O(n)$:

level array: elements always differ by 1

LCA $\xrightarrow{+1}$ RMQ

+1 −1 +1 −1 −1 +1 −1 ... $\log n/2$

only $2^{\log n/2} = \sqrt{n}$ number of total sequences

since max $\sqrt{n}$ sequences → time is $O(\sqrt{n} \times \log n \times \log\log n)$

* limited # of sequences → store in lookup table

RMQ → LCA → +/− RMQ

linear time reduction to each other