divide into \( \frac{n}{b} \) blocks of size \( b \)

store prefix/suffix mins inside each block \( \leq O(n) \)

use Method 6 inside each block \( \leq O(\frac{n}{b} \cdot b \log b) \)

use Method 6 for the mins of all blocks \( \leq O(\frac{n}{b} \log \frac{n}{b}) \)

query time \( O(1) \)

Space/preproc time \( O(n + \frac{n}{b} \cdot b \log b + \frac{n}{b} \log \frac{n}{b}) \)

\( b = \log n \) \( \Rightarrow O(n \log \log n) \)

**Method 8:** recursion!

use recursion inside each block
rest same

space/preproc time

\[
S(n) = O(n) + O(\frac{n}{b} \log \frac{n}{b}) + \frac{n}{b} S(b)
\]

Set \( b = \log n \)

\[
S(n) = \frac{n}{\log n} S(\log n) + O(n)
\]

\[\rightarrow n\]
**Method 9** bootstrap again:

\[
S(n) = O(n) + O\left(\frac{n}{b} \log^* \frac{n}{b}\right) + \frac{n}{b} S(b)
\]

Set \( b = \log^* n \)

\[
S(n) = O(n) + \frac{n}{\log^* n} S(\log^* n)
\]

\[
\Rightarrow \text{space/prep time } O(n \log^{**} n)
\]

\[
\Rightarrow \text{query time } O(1)
\]

Bootstrap \( l \) times

\[
\text{space/prep } O(n (\log \cdots \log n))
\]

\[
\text{query time } O(1)
\]

Def. Inverse Ackermann fn
**Def** Inverse Ackermann function
\[ \alpha(n) = \text{smallest } \ell \text{ s.t. } \log^\ell n \leq \text{const} \]

Query time: \( O(\alpha(n)) \)
Space prepare: \( O(n) \)

[optimal for semigroup op]

**Related Problem** (Lowest Common Ancestor (LCA))
Given binary tree \( T \) with \( n \) nodes,
build data structure to answer following query:
find nodes \( u, v \), find LCA

**Thm** RMQ reduces to LCA.

**Pf:** Given \( a_1, \ldots, a_n \), define Cartesian tree:
- root is min
- recurse on all elements to its left
- right

\[ (2) ((1)(1)) ((1)(1)) ((1)) \]
Cartesian tree can be built in \( O(n) \) time:

e.g. insert elements from left to right

maintain rightmost path in stack \( S \)

3 \( \Rightarrow \) 4 \( \Rightarrow \) 5

\( \times \)

To insert \( a_i \):

while \( S \text{.top}() > a_i \) \( S \text{.pop}() \)
del 1 edge, insert 2 edges

\( S \text{.push}(a_i) \).

(\text{amortised analysis!})

Total time \( O(n + \#\text{pops}) \)

\( \leq O(n + \#\text{pushes}) \)

\( = O(n) \)

Then LCA reduces to RMQ.

\text{Pf:}

\( \text{Given tree} \)

\( \)
Pf:

Given tree

look at depth values
take in-order traversal

2 1 5 4 6 5 6 3 2 3

Method 10 (Final) (Harel, Tarjan '84 /
Schieber, Vishkin '88)
Bender, Farach-Costa '00

divide into \( \frac{n}{b} \) blocks of size \( b \) as before

how to solve subproblems of size \( b \) directly?

precompute all answers
for all inputs of size \( b \).

# inputs of size \( b \)

= # binary trees of size \( b \)

\leq 2^{4b}

can encode binary tree as sequence of
\leq 4b bits

query by table lookup

with size
query by table lookup
\[ \text{ans}[T, i, j] \]

table size \( O(2^{ab}b^2) \)

Space/proc time
\[ \leq O(n + \frac{a}{b}(\log \frac{a}{b})) + O(2^{ab}b^2) + O(\frac{n}{b} \cdot b) \]

query time \( O(1) \)

set \( b = \frac{\log a}{8} \)

\( O(n) \) Space/proc time