ANIL MAHESHWARI

anil@scs.carleton.ca

School of Computer Science

CARLETON U. OTTAWA CANADA

1982-87 BITS (MATH+EEE)

87-93 TIFR (Ph.D CS)

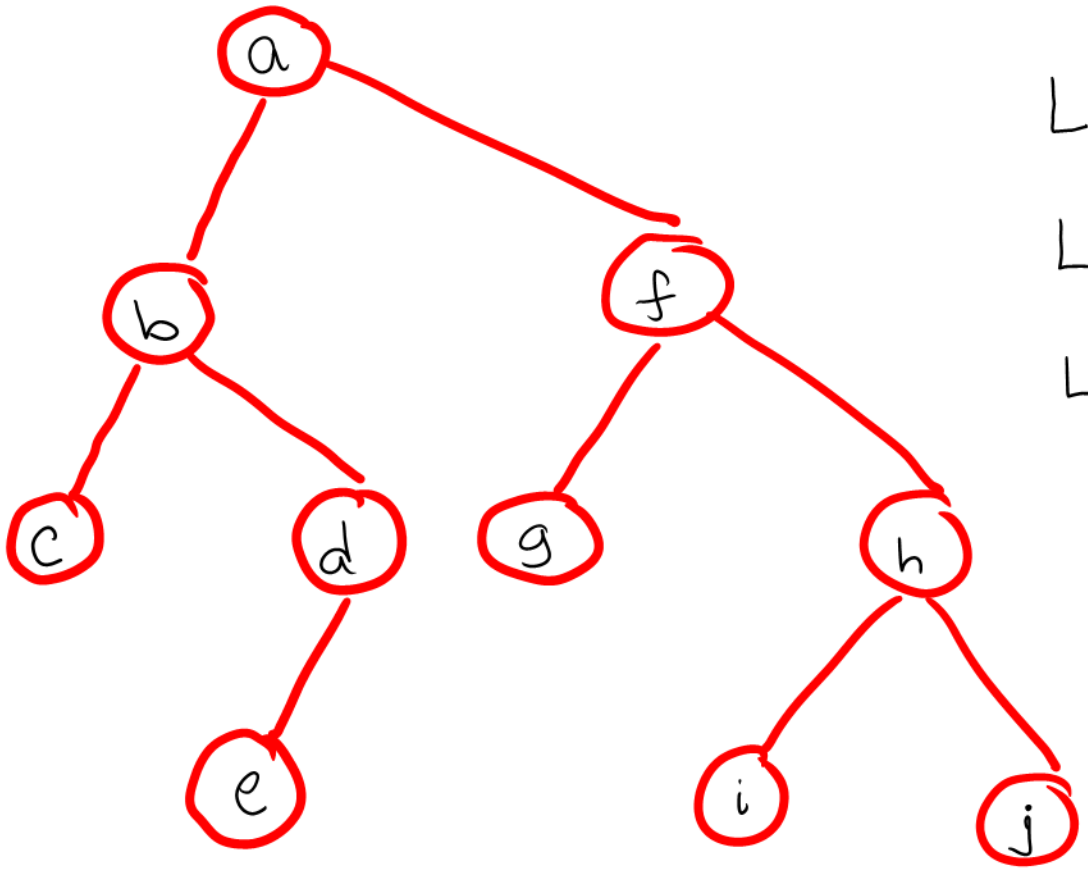
93-95 MPI Germany

>95 C.U.

Lowest Common Ancestor

(Bader + Farach-Colton 2000)

T



$LCA(d,i) = a$

$LCA(b,h) = a$

$LCA(g,j) = f$

Problem

Preprocess T in $O(n)$ time so that **Queries** can be answered in $O(1)$ time.

History

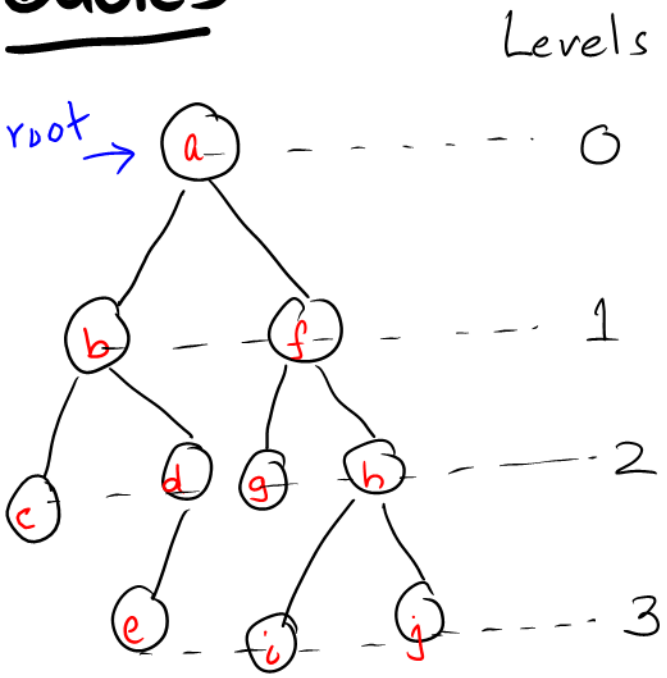
- Fundamental Problem
- Subproblem in several graph algorithms
- Harel + Tarjan 1984 (O(n), O(1))
- Schieber + Vishkin 1993 (O(n), O(1))

Complex, Not Practical

- Bader + Farach-Colton 2000 (O(n), O(1))

Simple + Easy + Textbook Material.

Basics



Depth First Traversal

→ Visit the root.

→ search the left tree, recursively

→ search the right-tree, recursively

Euler Tour

" Keeps the history of the dfs traversal.

E = [a b c b d e d b a f g f h i h j h f a]

L = [0 1 2 1 2 3 2 1 0 1 2 1 2 3 2 3 2 1 0]

LCA(c, i) =

LCA(g, j) =

Problem is reduced to finding the minimum element in L-array !

So far the Steps are

Preprocessing

1. Compute Euler Tour E

   O(n)

2. Compute First Occurence of Nodes in E-array

   O(n)

3. Compute Level array L.

   O(n)

Query (LCA(α, β))

1. Locate the first occurence of nodes α and β in the E-array O(1)

2. Locate the corresponding entries in L-array (say α′, β′) O(1)

3. Answer a Range Minima Query for L[α′, ..., β′].

   RMQ(α′, β′) ?

Naive RMQ

Precompute answers to all possible queries in a table $M[i,j]$.

For $i := 1$ to n do

  For $j := i$ to n do

    Compute $M[i,j]$ = index of the minimum element in $A[i \ldots j]$.

Note that $M[i,j+1] = \min[M[i,j], A[j+1]]$

Each value in the table can be computed in $O(1)$ time $\Rightarrow$ Overall $O(n^2)$ time.

Range Minima Queries

Input : An Array A of n-numbers.

Query : Two Indices (i, j)
where $1 \leq i \leq j \leq n$.

Output : $RMQ(i, j)$ = Minimum Element or its index in $A[i \ldots j]$.

	Naive	Smart	± 1 property
Preprocessing	$O(n^2)$	$O(n \log n)$	$O(n)$
Query Answering	$O(1)$	$O(1)$	$O(1)$

Smart RMQ

In place of computing all entries in the table we compute only $\log_2 n$ entries in each row.

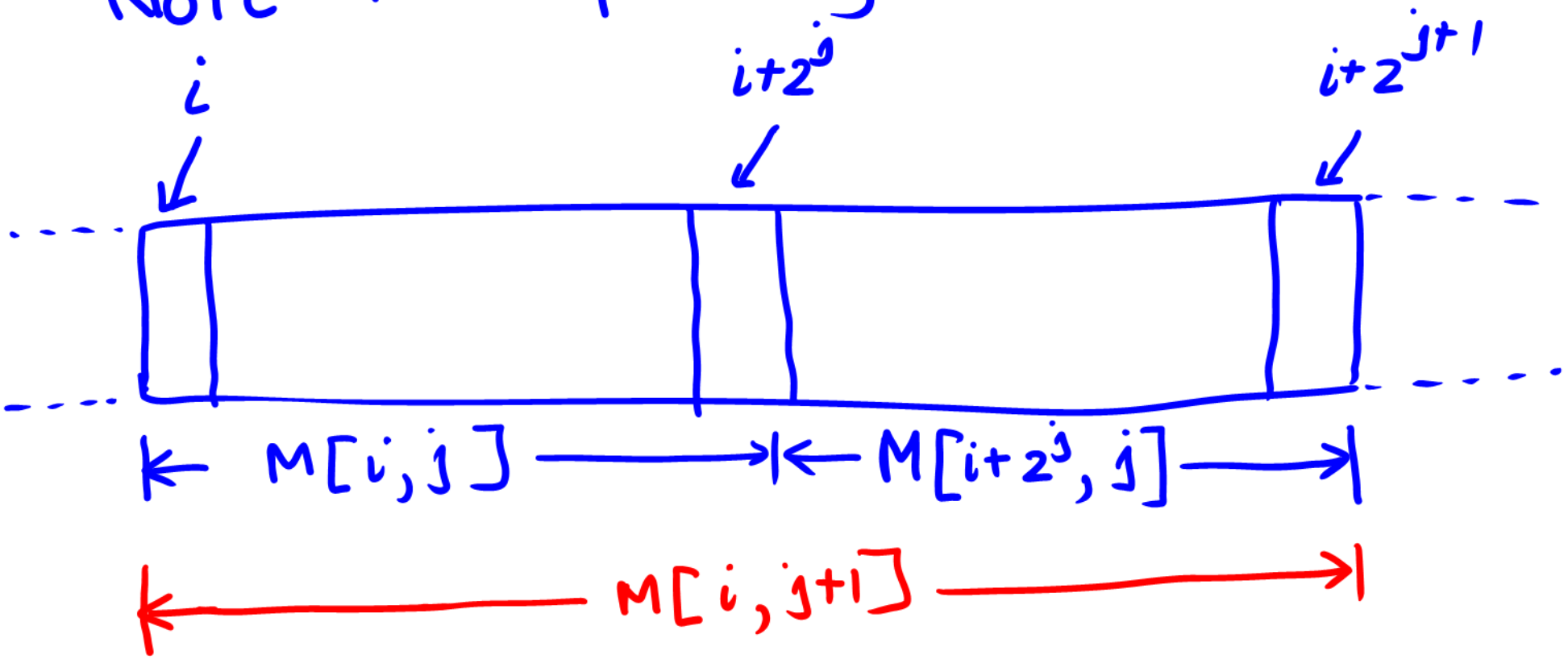
Entries Computed in a row are

$$M[i, i+0],\ M[i, i+2],\ M[i, i+2^2],\ M[i, i+2^3],\ \ldots,\ M[i, i+2^{\log_2 n}]$$

How ?

```
FOR i := 1 to n do
    For j := 0 to log2 n do
        COMPUTE M[i, j]
```

Note the following



Therefore $M[i, j+1] = \text{MIN}\left[M[i, j], M[i+2^j, j]\right]$

Each entry in the table can be computed in $O(1)$ time $\Rightarrow$ $O(n \log n)$ time overall since table consists of $O(n \log n)$ entries.

How to answer RMQ queries using SMART.

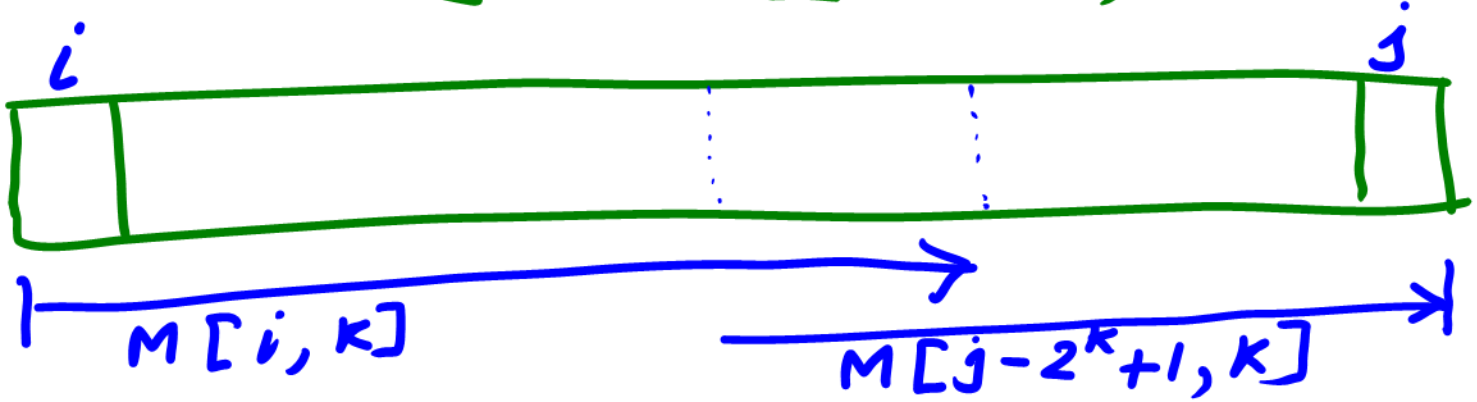
RMQ[i, j]

Step 1: Find the largest power of 2 that can fit in the interval from i..j.

Let $k = \lfloor \log_2(j-i) \rfloor$

Then 2^k is the largest interval.

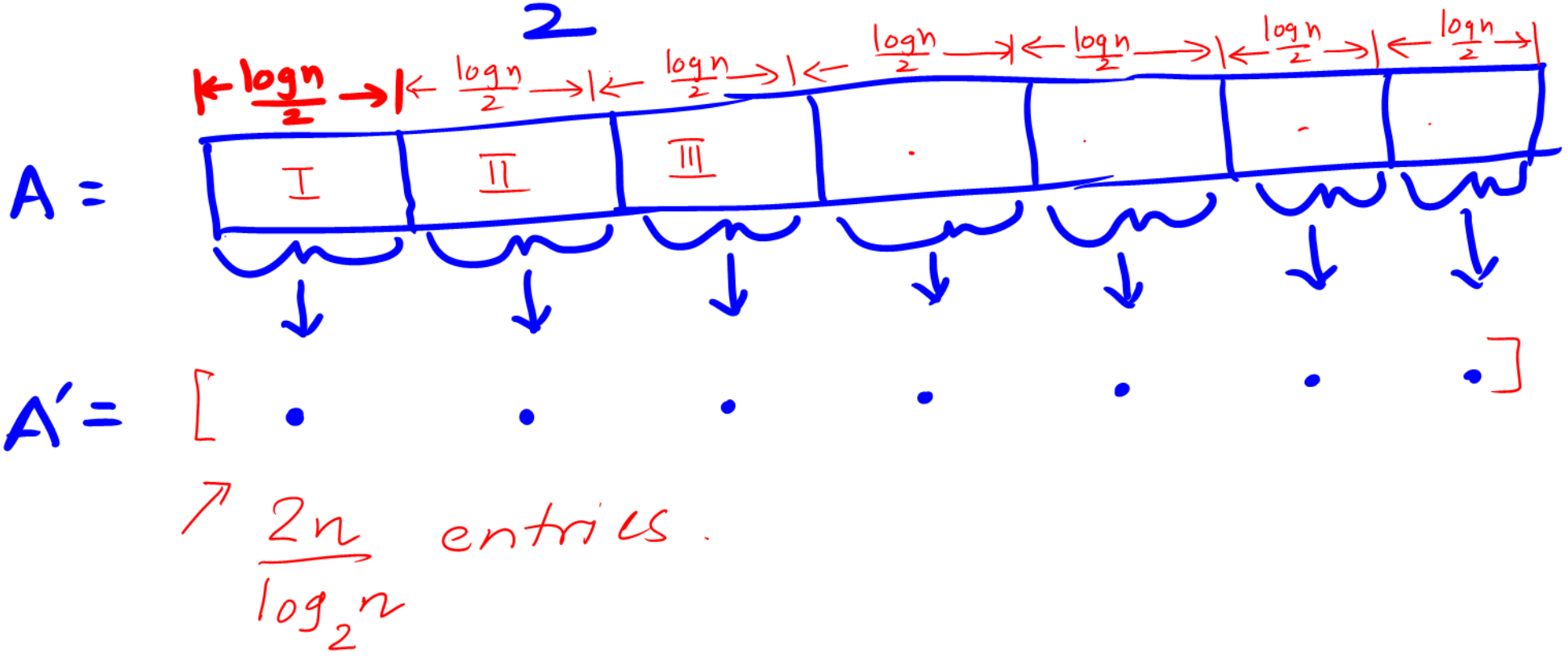
Step2: $RMQ[i,j] = MIN[M[i,k], M[j-2^k+1, k]]$



RMQ with ±1 property.

* In Level array entries differ from previous entry by a +1 or a -1.

* Partition A into subarrays of size $\frac{\log_2 n}{2}$.



Preprocessing:

1. Compute A' $O(n)$
2. Preprocess A' using SMART-RMQ $O(n)$
3. NORMALIZE EACH SUBARRAY $O(n)$
4. Preprocess all normalized subarrays $O(\sqrt{n}\log^2 n)$

What is a NORMALIZED SUBARRAY?

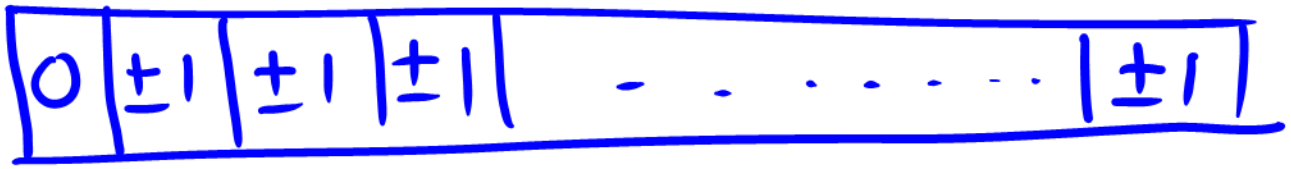
$S =$ 2 3 4 3 2 1 0 1 2 3

$S_N =$ 0 1 2 1 0 -1 -2 -1 0 1

$\pm 1 =$ 0 +1 +1 -1 -1 -1 -1 +1 +1 +1

KEY PROPERTY OF SUBARRAYS!

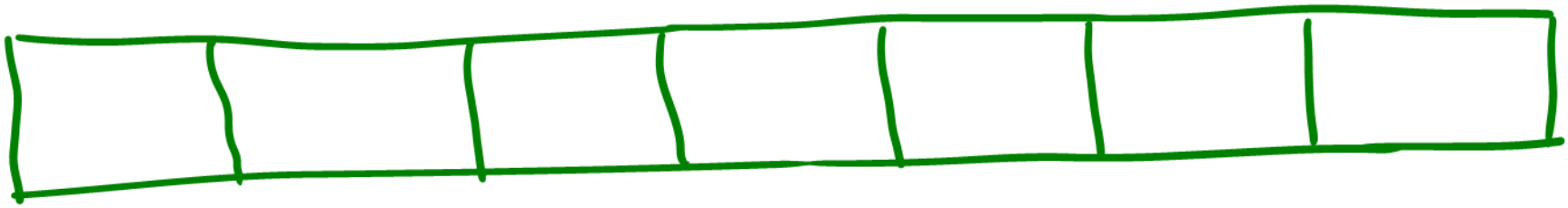
How many normalized subarrays of length $\frac{\log n}{2}$ are there?



$$= 2^{\frac{\log_2 n}{2} - 1} \approx O(\sqrt{n})$$

For each of them we use brute-force algorithm and preprocess them for RMQs. in $O(\sqrt{n} \log^2 n)$ time.

How To Answer ±1 RMQ queries



OPTION 1: i and j are within the same subarray

$\Rightarrow$ USE NORMALIZED SUBARRAY PREPROCESSING.

OPTION 2: i and j are in different subarrays.

CLAIM

LCA queries can be answered in $O(1)$ time after $O(n)$ preprocessing in a rooted binary tree on n nodes.

[Bader + Farach-Colton 2000]