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)

LCA(a, 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 $O(n), O(1)$ [COMPLEX NOT PRACTICAL]
  - Schieber + Vishkin 1993 $O(n), O(1)$
  - 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 processing

1. Compute Euler Tour \( E \) \( \mathcal{O}(n) \)
2. Compute First Occurrence \( \mathcal{O}(n) \)
3. Compute Level Array \( L \) \( \mathcal{O}(n) \)

**Query** \( \text{LCA}(\alpha, \beta) \)

1. Locate the first occurrence of nodes \( \alpha \) and \( \beta \) in the \( E \)-array \( \mathcal{O}(1) \)
2. Locate the corresponding entries in \( L \)-array (say \( \alpha', \beta' \)) \( \mathcal{O}(1) \)
3. Answer a RANGE MINIMA Query for \( L[\alpha', \ldots, \beta'] \)
   \( \text{RMQ}(\alpha', \beta') \) ?
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 $A[i...j]$.

<table>
<thead>
<tr>
<th>Preprocessing</th>
<th>Naive</th>
<th>Smart</th>
<th>$i+1$ Property</th>
</tr>
</thead>
<tbody>
<tr>
<td>Query Answering</td>
<td>$O(n^2)$</td>
<td>$O(n \log n)$</td>
<td>$O(n)$</td>
</tr>
<tr>
<td></td>
<td>$O(1)$</td>
<td>$O(1)$</td>
<td>$O(1)$</td>
</tr>
</tbody>
</table>
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

$$M[i, i+0], M[i, i+2], M[i, i+4], M[i, i+8], \ldots, M[i, i+2^{\log_2 n}]$$

How?

```
FOR i := 1 TO n do
  FOR j := 0 TO $\log_2 n$ do
    COMPUTE M[i, j]
```
Each entry in the table can be computed in $O(\log n)$ time. Overall, the time to compute all entries is $O(n \log n)$.

Therefore, $M[i,j+1] = \min \{ M[i,j], M[i+1,j], M[i+2,j] \}$.
How to answer RMQ queries using SMART.

\[ \text{RMQ}[i, j] \]

Step 1: Find the largest power \( k \) of 2 that can fit in the interval from \( i \) to \( j \).

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

Then \( 2^k \) is the largest interval.

Step 2: \( \text{RMQ}[i, j] = \min \left[ \text{M}[i, k], \text{M}[j - 2^k + 1, k] \right] \)
RMQ with \( \pm 1 \) property.

- In Level array entries differ from previous entry by a \( +1 \) or a \( -1 \).

- Partition \( A \) into subarrays of size \( \frac{\log n}{2} \).

\[
A = \begin{bmatrix}
\text{I} & \text{II} & \text{III} & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot
\end{bmatrix}
\]

\[
A' = \begin{bmatrix}
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot
\end{bmatrix}
\]

\( \frac{2n}{\log_2 n} \) entries.
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 $O(n \log^2 n)$

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 $\log_2 n$ are there?

$$
0 \pm 1 \pm 1 \pm 1 \ldots \pm 1
$$

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

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 \( \pm 1RMQ \) Queries

**OPTION 1:** \( i \) and \( j \) are within the same subarray

\[ \Rightarrow \text{USE NORMALIZED SUBARRAY PREPROCESSING.} \]

**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]