Competitive Programming and Mathematics Society

# Programming Workshop #6

## Euler Tours

**Patrick and Ryan**

CPMSOC

Arc CLUBS SUPPORTED BY Arc INDEPENDENTLY RUN

# Euler Tours

The Euler Tour Technique (ETT) is a method of turning a labeled rooted tree into an array that preserves the structure of the tree. The array represents the order in which nodes are visited when a depth-first search is applied to the tree. It's similar to an in-order traversal, but since nodes can be visited more than once (for example once when recursing down from the root, and once when returning from a subtree).

# Euler Tours

The Euler Tour Technique (ETT) is a method of turning a labeled rooted tree into an array that preserves the structure of the tree. The array represents the order in which nodes are visited when a depth-first search is applied to the tree. It's similar to an in-order traversal, but since nodes can be visited more than once (for example once when recursing down from the root, and once when returning from a subtree).

These arrays are produced by a depth-first search, where every time we visit a node, we add it to the tour.

# Properties of Euler Tours

In addition to the tour itself, it is often useful to store the indexes at which the first and last occurrences of each node occur, which we can denote $first[nodeNumber]$ and $last[nodeNumber]$.

# Properties of Euler Tours

In addition to the tour itself, it is often useful to store the indexes at which the first and last occurrences of each node occur, which we can denote $first[nodeNumber]$ and $last[nodeNumber]$.

Euler Tours have a couple nice properties which we can apply:

- The subarray from $first[nodeNumber]$ to $last[nodeNumber]$ represents the subtree rooted at $nodeNumber$
- The subarray from $first[u]$ to $first[v]$ can go no higher than the LCA of $u$ and $v$
- Each node $v$ is inside the subarray of all of its ancestors, and no other nodes.

# Variations on Euler Tours

There are many ways to implement a Euler Tours each of which produces slightly different results, and has slightly different applications.

# Variations on Euler Tours

There are many ways to implement a Euler Tours each of which produces slightly different results, and has slightly different applications.

- 1. Track the edges taken when doing the DFS, and add each of those edges into the array (the array becomes an array of edges). This is useful when operating and answering queries on edges or paths between nodes where the edges are labelled.
- 2. Add the vertex into array once when you descend into it, and once when you ascend out of it. This can be used when answering queries on the paths between nodes where the nodes are labelled.
- 3. Add the vertex into the array every time you touch it, either descending into it or returning from a child. This is the preferred ordering for answering LCA Queries.

# LCA with Euler Tour

Recall the problem of answering LCA Queries: Given two nodes, $u$ and $v$, return the Lowest Common Ancestor (ancestor of both $u$ and $v$ which is closest to the root) of the two nodes.

# LCA with Euler Tour

Recall the problem of answering LCA Queries: Given two nodes, $u$ and $v$, return the Lowest Common Ancestor (ancestor of both $u$ and $v$ which is closest to the root) of the two nodes.

The nodes in the subarray from $first[u]$ to $first[v]$ inclusive will include the LCA, but will not include any nodes above the LCA. Hence, the LCA is the node in the subarray which has the smallest depth.

# LCA with Euler Tour

Recall the problem of answering LCA Queries: Given two nodes, $u$ and $v$, return the Lowest Common Ancestor (ancestor of both $u$ and $v$ which is closest to the root) of the two nodes.

The nodes in the subarray from $first[u]$ to $first[v]$ inclusive will include the LCA, but will not include any nodes above the LCA. Hence, the LCA is the node in the subarray which has the smallest depth.

Thus, in order to find the LCA, we store the height of each node as we create the Euler Tour. This reduces the problem into finding the minimum value in a subarrray. This can be solved with $O(n * log(n))$ preprocessing and $O(1)$ query with a sparse table or $O(n)$ preprocessing and $O(log(n))$ query with your favourite range querying tree.

# Subtree Queries

Given a rooted tree, $T$ with $N$ nodes, support three operations:

- Set the value of node $i$ to $x$
- Query the sum of values in nodes in the subtree rooted at $i$
- Query the maximum value in a node in the subtree rooted at $i$

# Advance and Retreat Edges

When completing the Euler Tour, edges will be taken twice each, once on the descent and once on the ascent. When using an edge-based Euler Tour, labelling these edges as advance or retreat can be somewhat helpful. For example, we can calculate:

- The level of a node

# Advance and Retreat Edges

When completing the Euler Tour, edges will be taken twice each, once on the descent and once on the ascent. When using an edge-based Euler Tour, labelling these edges as advance or retreat can be somewhat helpful. For example, we can calculate:

- The level of a node: Advance = 1, Retreat = -1 -> Take prefix sum

# Advance and Retreat Edges

When completing the Euler Tour, edges will be taken twice each, once on the descent and once on the ascent. When using an edge-based Euler Tour, labelling these edges as advance or retreat can be somewhat helpful. For example, we can calculate:

- The level of a node: Advance = 1, Retreat = -1 -> Take prefix sum
- The number of nodes in a subtree

# Advance and Retreat Edges

When completing the Euler Tour, edges will be taken twice each, once on the descent and once on the ascent. When using an edge-based Euler Tour, labelling these edges as advance or retreat can be somewhat helpful. For example, we can calculate:

- The level of a node: Advance = 1, Retreat = -1 -> Take prefix sum
- The number of nodes in a subtree : Advance = 1, Retreat = 0 -> Take sum between advance and retreat of root

# Advance and Retreat Edges

When completing the Euler Tour, edges will be taken twice each, once on the descent and once on the ascent. When using an edge-based Euler Tour, labelling these edges as advance or retreat can be somewhat helpful. For example, we can calculate:

- The level of a node: Advance = 1, Retreat = -1 -> Take prefix sum
- The number of nodes in a subtree : Advance = 1, Retreat = 0 -> Take sum between advance and retreat of root
- The DFS index of a node

# Advance and Retreat Edges

When completing the Euler Tour, edges will be taken twice each, once on the descent and once on the ascent. When using an edge-based Euler Tour, labelling these edges as advance or retreat can be somewhat helpful. For example, we can calculate:

- The level of a node: Advance = 1, Retreat = -1 -> Take prefix sum
- The number of nodes in a subtree : Advance = 1, Retreat = 0 -> Take sum between advance and retreat of root
- The DFS index of a node : Advance = 1, Retreat = 0; Take prefix sum up to advance

# USACO Gold Milk Visits

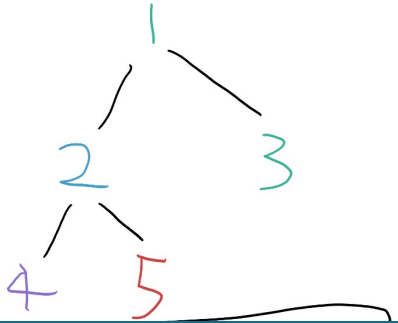
http://www.usaco.org/index.php?page=viewproblem2&cpid=970

Given a tree with $N$ coloured nodes, is there a node of colour $C_i$ on the path from $X_i$ to $Y_i$? Answer these types of queries $Q$ times. Subtask 1: $N, Q \leq 10^3$

Subtask 2: number of colours $\leq 10$ and $N, Q \leq 10^5$

Subtask 3: $N, Q \leq 10^5$

Think about: example, reductions, precomputation



# USACO Gold Milk Visits - Subtask 1

Subtask 1: $N, Q \leq 10^3$
Calculate LCA using Euler tour
traverse the path $X \to \text{lca}(X, Y) \to Y$
$\mathcal{O}(NM)$

# USACO Gold Milk Visits - Subtask 2

Subtask 2: number of colours $\leq 10$
Tree prefix for each colour:

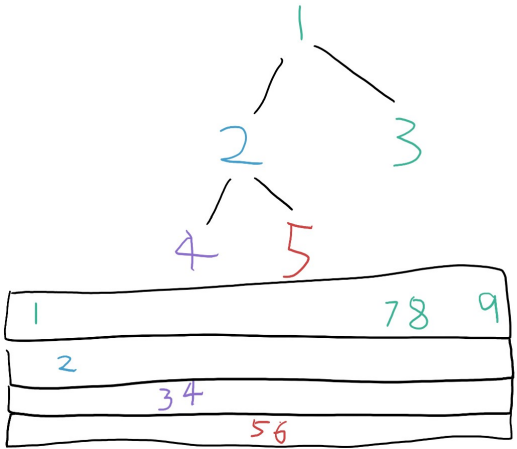
$$\max(prefix[C_i][X], prefix[C_i][Y]) > prefix[C_i][\text{parent}(\text{lca}(X, Y))]$$

$\mathcal{O}(10N + M \log N)$

# USACO Gold Milk Visits - Subtask 3

Subtask 3: $N, Q \leq 10^5$

Euler tour for each colour.




$O(N + M \log N)$

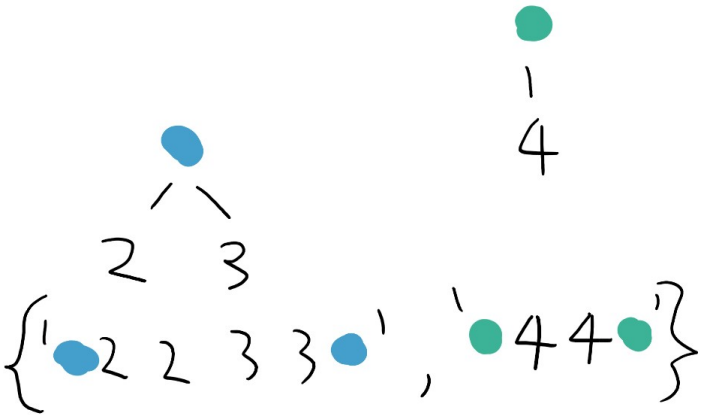
# Better Disjoint Set Unions

Disjoint set unions are kinda boring...
Make one which supports these operations:

- Check whether 2 items are in the same set
- Merge 2 sets
- *Undo the previous merge operation*

# Better Disjoint Set Unions - Solution

Euler tour trees!



# Attendance form :D

https://forms.gle/VDNqRSQyNA3xdzug6

# Wrap Up

- Last workshop for the term in week 9
- Next weeks Maths workshop! (Tuesday 12-2)
- Programming weekly blog posts!

# Additional questions

I recommend that you try implementing the example problems first.

- CSES Substring queries https://cses.fi/problemset/task/1137
- CSES Distinct Colors https://cses.fi/problemset/task/1139
- CSES Path Queries https://cses.fi/problemset/task/1138
- USACO Promotion Counting http://www.usaco.org/index.php?page=viewproblem2&cpid=696