

Range queries

Gilles Englebert

Slides at englebert.lu

May 19, 2026

Overview

Range queries

Gilles
Englebert

Range queries

The static
case

The dynamic
case

Lowest
Common
Ancestor

Homework

References

- 1 Range queries
- 2 The static case
- 3 The dynamic case
- 4 Lowest Common Ancestor
- 5 Homework
- 6 References

Task

Range queries

Gilles
Englebert

Range queries

The static
case

The dynamic
case

Lowest
Common
Ancestor

Homework

References

Situation: Let $a[1 \dots n]$ be an array of integers.

Task

Range queries

Gilles
Englebert

Range queries

The static
case

The dynamic
case

Lowest
Common
Ancestor

Homework

References

Situation: Let $a[1 \dots n]$ be an array of integers. Say we need to compute, for $i < j$, **range sum queries** (e.g. running average of temperatures):

$$\text{RSQ}(i, j) := \sum_{k=i}^j a[k] = a[i] + \dots + a[j],$$

Task

Range queries

Gilles
Englebert

Range queries

The static
case

The dynamic
case

Lowest
Common
Ancestor

Homework

References

Situation: Let $a[1 \dots n]$ be an array of integers. Say we need to compute, for $i < j$, **range sum queries** (e.g. running average of temperatures):

$$\text{RSQ}(i, j) := \sum_{k=i}^j a[k] = a[i] + \dots + a[j],$$

or **range minimum queries** (e.g. minimum elevation in part of a map):

$$\text{RMQ}(i, j) := \min(a[i], \dots, a[j]).$$

We need to repeat these computations many times for different choices of i, j .

Range queries

Gilles
Englebert

Range queries

The static
case

The dynamic
case

Lowest
Common
Ancestor

Homework

References

These are multiple different problems, depending on whether we are in the:

These are multiple different problems, depending on whether we are in the:

- **Static problem:** $a[1 \dots n]$ is fixed.

These are multiple different problems, depending on whether we are in the:

- **Static problem:** $a[1 \dots n]$ is fixed.
- **Dynamic problem:** $a[1 \dots n]$ can be updated in between queries.

These are multiple different problems, depending on whether we are in the:

- **Static problem:** $a[1 \dots n]$ is fixed.
- **Dynamic problem:** $a[1 \dots n]$ can be updated in between queries.

It also depends on the binary operation that we consider, $a + b$ is fundamentally different from $\min(a, b)$. For example, addition has an *inverse operation*, namely $-$, which \min does not have.

The *static* case

Range queries

Gilles
Englebert

Range queries

The static
case

The dynamic
case

Lowest
Common
Ancestor

Homework

References

Initial setup time by **precomputation** allows for faster queries. This works differently for RSQ and RMQ. Let's begin with **sums**.

Solution

Calculate $s[k] = \sum_{i=1}^k a[i]$ for $1 \leq k \leq n$ in time $O(n)$. Then

$$\text{RSQ}(i, j) = s[j] - s[i - 1]$$

in $O(1)$!

The *static* case

Range queries

Gilles
Englebert

Range queries

The static
case

The dynamic
case

Lowest
Common
Ancestor

Homework

References

Initial setup time by **precomputation** allows for faster queries. This works differently for RSQ and RMQ. Let's begin with **sums**.

Solution

Calculate $s[k] = \sum_{i=1}^k a[i]$ for $1 \leq k \leq n$ in time $O(n)$. Then

$$\text{RSQ}(i, j) = s[j] - s[i - 1]$$

in $O(1)$!

Here we used the fact that $-$ 'undoes' $+$.

Q: does \oplus (xor) have an inverse operation?

The *static* case

Range queries

Gilles
Englebert

Range queries

The static
case

The dynamic
case

Lowest
Common
Ancestor

Homework

References

Initial setup time by **precomputation** allows for faster queries. This works differently for RSQ and RMQ. Let's begin with **sums**.

Solution

Calculate $s[k] = \sum_{i=1}^k a[i]$ for $1 \leq k \leq n$ in time $O(n)$. Then

$$\text{RSQ}(i, j) = s[j] - s[i - 1]$$

in $O(1)$!

Here we used the fact that $-$ 'undoes' $+$.

Q: does \oplus (xor) have an inverse operation?

This is only faster since the $a[i]$ do not change. If they did and we had to recompute the $s[k]$, at least partially, and querying would take time $O(n)$ after all.

Range minimum queries

Range queries

Gilles
Englebert

Range queries

The static
case

The dynamic
case

Lowest
Common
Ancestor

Homework

References

This does not work for min, since we cannot 'undo' the operation. However, taking minimums is **idempotent**, meaning that $\min(a, b) = \min(a, b, b)$.

Range minimum queries

Range queries

Gilles
Englebert

Range queries

The static
case

The dynamic
case

Lowest
Common
Ancestor

Homework

References

This does not work for min, since we cannot 'undo' the operation. However, taking minimums is **idempotent**, meaning that $\min(a, b) = \min(a, b, b)$.

We can use this idea by splitting $[1, n]$ into large intervals. Then we can take the minimum of elements in the interval $[a, b]$ by taking the minimum of a few (potentially overlapping) large intervals.

Range minimum queries

Range queries

Gilles
Englebert

Range queries

The static
case

The dynamic
case

Lowest
Common
Ancestor

Homework

References

This does not work for min, since we cannot 'undo' the operation. However, taking minimums is **idempotent**, meaning that $\min(a, b) = \min(a, b, b)$.

We can use this idea by splitting $[1, n]$ into large intervals. Then we can take the minimum of elements in the interval $[a, b]$ by taking the minimum of a few (potentially overlapping) large intervals.

There are two ways of going about this:

- **Square root decomposition**
- **Divide and conquer**

Square root decomposition

Range queries

Gilles
Englebert

Range queries

The static
case

The dynamic
case

Lowest
Common
Ancestor

Homework

References

We divide the array into $k = \sqrt{n}$ blocks of size at most k .



Figure: Example with $n = 9$

Square root decomposition

Range queries

Gilles
Englebert

Range queries

The static
case

The dynamic
case

Lowest
Common
Ancestor

Homework

References

We divide the array into $k = \sqrt{n}$ blocks of size at most k .



Figure: Example with $n = 9$

For this example we can precompute in $O(n)$ the minima per block, i.e. $s = \{1, 3, 2\}$. Now, when we want to compute the minimum for a range $[2, 7]$ we decompose:

$$\text{RMQ}(2, 7) = \min(a[2], s[1], a[6], a[7]) = 1.$$

Square root decomposition

Range queries

Gilles
Englebert

Range queries

The static
case

The dynamic
case

Lowest
Common
Ancestor

Homework

References

We divide the array into $k = \sqrt{n}$ blocks of size at most k .



Figure: Example with $n = 9$

For this example we can precompute in $O(n)$ the minima per block, i.e. $s = \{1, 3, 2\}$. Now, when we want to compute the minimum for a range $[2, 7]$ we decompose:

$$\text{RMQ}(2, 7) = \min(a[2], s[1], a[6], a[7]) = 1.$$

We can query in time $O(\sqrt{n})$.

Divide and conquer

Range queries

Gilles
Englebert

Range queries

The static
case

The dynamic
case

Lowest
Common
Ancestor

Homework

References

We can also divide a segment into two large overlapping parts. For this we calculate the minima already for all subsets of sizes $2^k \leq n$ which are contained in $[1, n]$. There are $O(n \log n)$ such segments.

Divide and conquer

Range queries

Gilles
Englebert

Range queries

The static
case

The dynamic
case

Lowest
Common
Ancestor

Homework

References

We can also divide a segment into two large overlapping parts. For this we calculate the minima already for all subsets of sizes $2^k \leq n$ which are contained in $[1, n]$. There are $O(n \log n)$ such segments.

Solution

Let $s[i][k]$ be $\text{RMQ}(i, i + 2^k - 1)$. Then we can compute it as:

$$s[i][0] = a[i],$$

$$s[i][d + 1] = \min(s[i][d], s[\min(n - 2^d, i + 2^d)][d]).$$

The RMQ problem can now be solved as follows:

$$\text{RMQ}(i, j) = \min(s[i][d], s[j - 2^d + 1][d])$$

where d is the smallest integer such that $2^d \leq |i - j|$.

Homework - static range queries

Range queries

Gilles
Englebert

Range queries

The static
case

The dynamic
case

Lowest
Common
Ancestor

Homework

References

- UVa 507 - Jil rides again
- CIL/LIO 2011 DF - Blocky

Dynamic Range Sum

Range queries

Gilles
Englebert

Range queries

The static
case

The dynamic
case

Lowest
Common
Ancestor

Homework

References

When the $a[i]$'s can vary in-between queries, **Segment trees** allow for a solution similar the the DC approach above, but with a more efficient update routine.

Dynamic Range Sum

Range queries

Gilles
Englebert

Range queries

The static
case

The dynamic
case

Lowest
Common
Ancestor

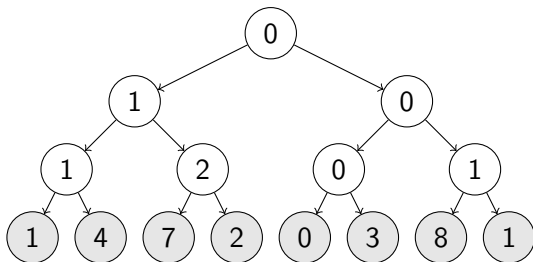
Homework

References

When the $a[i]$'s can vary in-between queries, **Segment trees** allow for a solution similar the the DC approach above, but with a more efficient update routine. Take the following array of size 8:

1	4	7	2	0	3	8	1
---	---	---	---	---	---	---	---

We construct the following binary tree which covers, at different levels, segments of lengths 1, 2, 4 and 8:



Querying

Range queries

Gilles
Englebert

Range queries

The static
case

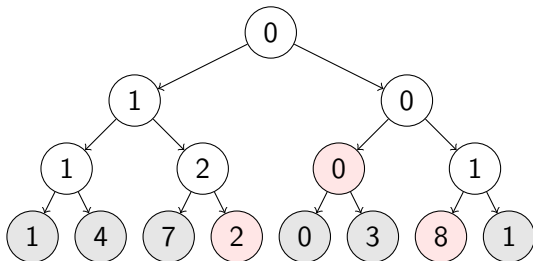
The dynamic
case

Lowest
Common
Ancestor

Homework

References

Say we want to find $\text{RMQ}(3, 6)$. Then we need to inspect the nodes:



This takes time $O(\log n)$, since we can usually stop early in the tree when a large segment is already precomputed.

Updating

Range queries

Gilles
Englebert

Range queries

The static
case

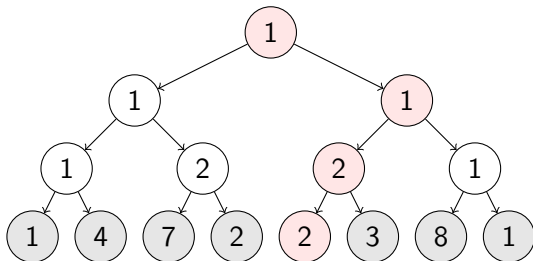
The dynamic
case

Lowest
Common
Ancestor

Homework

References

If we want to change $a[4] = 2$, this will have an impact on the following nodes:



This also takes time $O(\log n)$, since climb up the tree from the affected leaf towards the root, and a full binary tree has depth $\log n$.

In practice

Range queries

Gilles
Englebert

Range queries

The static
case

The dynamic
case

Lowest
Common
Ancestor

Homework

References

So a segment tree consists of three components:

- **construction** in $O(n)$
- **update** of one element in $O(\log(n))$
- **query** in $O(\log(n))$

Let us look at some code!

Link: Visualization of Segment Trees.

Task

Range queries

Gilles
Englebert

Range queries

The static
case

The dynamic
case

Lowest
Common
Ancestor

Homework

References

We finish with an application of range minimum queries.

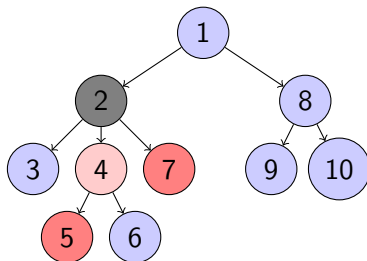


Figure: Tree with $n = 10$ vertices, $\text{LCA}(5, 7) = 2$

Find the **lowest common ancestor** $\text{LCA}(i, j)$ of two nodes i and j .

Another example

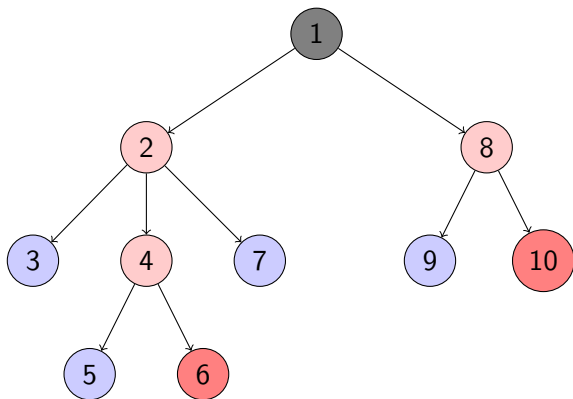


Figure: Tree with $n = 10$ vertices, $\text{LCA}(6, 10) = 1$

Solution

Range queries

Gilles
Englebert

Range queries

The static
case

The dynamic
case

Lowest
Common
Ancestor

Homework

References

We reduce LCA to RMQ by noting that the lowest common ancestor is the node with the lowest depth that is passed in DFS between the two nodes one is concerned about.

Solution

Construct the arrays $E[1 \dots 2n - 1]$ (the sequence of nodes traversed in a DFS), $L[1 \dots 2n - 1]$ (their depths) and $H[1 \dots n]$, the first appearance in the list of the nodes. Notice that:

$$\text{LCA}(i, j) = E[\text{RMQ}_L(H[i], H[j])]$$

Example

Range queries

Gilles
Englebert

Range queries

The static
case

The dynamic
case

Lowest
Common
Ancestor

Homework

References

Let's execute the algorithm for the second example.

Index	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
H	1	2	3	5	6	8	11	14	15	17	-	-	-	-	-	-	-	-	-
E	1	2	3	2	4	5	4	6	4	2	7	2	1	8	9	8	10	8	1
L	0	1	2	1	2	3	2	3	2	1	2	1	0	1	2	1	2	1	0

Indeed, the vertex 1 is the lowest common ancestor of 6 and 10.

Homework

Range queries

Gilles
Englebert

Range queries

The static
case

The dynamic
case

Lowest
Common
Ancestor

Homework

References

- UVa 12532 - Interval Product (Segment Tree, easy)
- UVa 11235 - Frequent Values (Segment Tree)
- UVa 10938 - Flea Circus (LCA)
- IOI 2015 - Horses
- IOI 2019 - Arranging Shoes

References

Range queries

Gilles
Englebert

Range queries

The static
case

The dynamic
case

Lowest
Common
Ancestor

Homework

References

And some references (also clickable!)

Competitive Programming book

USACO

Also, if you have any questions, do not hesitate to write me at
gilles@englebert.lu.