Prefix Sum

Arnar Bjarni Arnarson

September 21, 2025

School of Computer Science
Reykjavík University

Range queries

- We have an array A of size n.

Range queries

- We have an array A of size n.
- Given i, j, we want to answer:

Range queries

- We have an array A of size n.
- Given i, j, we want to answer:
  - $\max(A[i], A[i+1], \ldots, A[j-1], A[j])$

Range queries

- We have an array A of size n.
- Given i, j, we want to answer:
    - $\max(A[i], A[i+1], \ldots, A[j-1], A[j])$
    - $\min(A[i], A[i+1], \ldots, A[j-1], A[j])$

Range queries

- We have an array A of size n.
- Given i, j, we want to answer:
  - $\max(A[i], A[i+1], \ldots, A[j-1], A[j])$
  - $\min(A[i], A[i+1], \ldots, A[j-1], A[j])$
  - $\mathrm{sum}(A[i], A[i+1], \ldots, A[j-1], A[j])$

Range queries

- We have an array A of size n.
- Given i, j, we want to answer:
  - $\max(A[i], A[i+1], \ldots, A[j-1], A[j])$
  - $\min(A[i], A[i+1], \ldots, A[j-1], A[j])$
  - $\text{sum}(A[i], A[i+1], \ldots, A[j-1], A[j])$
- We want to answer these queries efficiently, or in other words, without looking through all elements.

Range queries

- We have an array A of size n.
- Given i, j, we want to answer:
  - $\max(A[i], A[i+1], \ldots, A[j-1], A[j])$
  - $\min(A[i], A[i+1], \ldots, A[j-1], A[j])$
  - $\text{sum}(A[i], A[i+1], \ldots, A[j-1], A[j])$
- We want to answer these queries efficiently, or in other words, without looking through all elements.
- Sometimes we also want to update elements.

Range sum on constant array

- Let's look at range sums on a constant array

Range sum on constant array

- Let's look at range sums on a constant array
- How do we support these queries efficiently?

Range sum on constant array

- Let's look at range sums on a constant array
- How do we support these queries efficiently?
- Simplification: only support queries of the form $\mathrm{sum}(0, j)$

Range sum on constant array

- Let's look at range sums on a constant array
- How do we support these queries efficiently?
- Simplification: only support queries of the form $\mathrm{sum}(0, j)$
- Notice that $\mathrm{sum}(i, j) = \mathrm{sum}(0, j) - \mathrm{sum}(0, i-1)$

Range sum on constant array

- So we're only interested in prefix sums

Range sum on constant array

- So we're only interested in prefix sums
- But there are only n of them...

Range sum on constant array

- So we're only interested in prefix sums
- But there are only n of them...
- Just compute them all once in the beginning

Range sum on constant array

- So we're only interested in prefix sums
- But there are only n of them...
- Just compute them all once in the beginning

1	0	7	8	5	9	3

Range sum on constant array

- So we're only interested in prefix sums
- But there are only n of them...
- Just compute them all once in the beginning

1	0	7	8	5	9	3
1						

Range sum on constant array

- So we're only interested in prefix sums
- But there are only n of them...
- Just compute them all once in the beginning

1	0	7	8	5	9	3
1	1					

Range sum on constant array

- So we're only interested in prefix sums
- But there are only n of them...
- Just compute them all once in the beginning

1	0	7	8	5	9	3
1	1	8				

Range sum on constant array

- So we're only interested in prefix sums
- But there are only n of them...
- Just compute them all once in the beginning

1	0	7	8	5	9	3
1	1	8	16			

Range sum on constant array

- So we're only interested in prefix sums
- But there are only n of them...
- Just compute them all once in the beginning

1	0	7	8	5	9	3
1	1	8	16	21		

Range sum on constant array

- So we're only interested in prefix sums
- But there are only n of them...
- Just compute them all once in the beginning

1	0	7	8	5	9	3
1	1	8	16	21	30	

Range sum on constant array

- So we're only interested in prefix sums
- But there are only n of them...
- Just compute them all once in the beginning

1	0	7	8	5	9	3
1	1	8	16	21	30	33

Range sum on constant array

- So we're only interested in prefix sums
- But there are only n of them...
- Just compute them all once in the beginning

1	0	7	8	5	9	3
1	1	8	16	21	30	33

- $O(n)$ time to preprocess

Range sum on constant array

- So we're only interested in prefix sums
- But there are only n of them...
- Just compute them all once in the beginning

1	0	7	8	5	9	3
1	1	8	16	21	30	33

- $O(n)$ time to preprocess
- $O(1)$ time each query

Range sum on constant array

- So we're only interested in prefix sums
- But there are only n of them...
- Just compute them all once in the beginning

1	0	7	8	5	9	3
1	1	8	16	21	30	33

- $O(n)$ time to preprocess
- $O(1)$ time each query

- Can we support updating efficiently?

Range sum on constant array

- So we're only interested in prefix sums
- But there are only n of them...
- Just compute them all once in the beginning

1	0	7	8	5	9	3
1	1	8	16	21	30	33

- $O(n)$ time to preprocess
- $O(1)$ time each query

- Can we support updating efficiently? No, at least not without modification

Generalizing

- This works on any invertible function.

Generalizing

- This works on any invertible function.
- If we want the product we can store the products and use $\text{mul}(i, j) = \text{mul}(0, j)/\text{mul}(0, i-1)$.

Generalizing

- This works on any invertible function.
- If we want the product we can store the products and use $\mathrm{mul}(i, j) = \mathrm{mul}(0, j)/\mathrm{mul}(0, i - 1)$.
- This also works for multidimensional arrays, but the math is more involved.

Generalizing

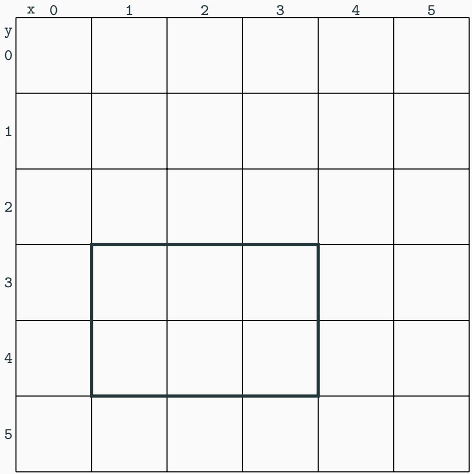
- This works on any invertible function.
- If we want the product we can store the products and use $\mathrm{mul}(i,j) = \mathrm{mul}(0,j)/\mathrm{mul}(0,i-1)$.
- This also works for multidimensional arrays, but the math is more involved.
- We let $\mathrm{sum}(x_i, x_j, y_i, y_j)$ denote the query for the sum from x_i to x_j along the x-dimension, and the same for y.

Generalizing

- This works on any invertible function.
- If we want the product we can store the products and use $\mathrm{mul}(i, j) = \mathrm{mul}(0, j)/\mathrm{mul}(0, i-1)$.
- This also works for multidimensional arrays, but the math is more involved.
- We let $\mathrm{sum}(x_i, x_j, y_i, y_j)$ denote the query for the sum from x_i to x_j along the x-dimension, and the same for y.
- Then the formula becomes

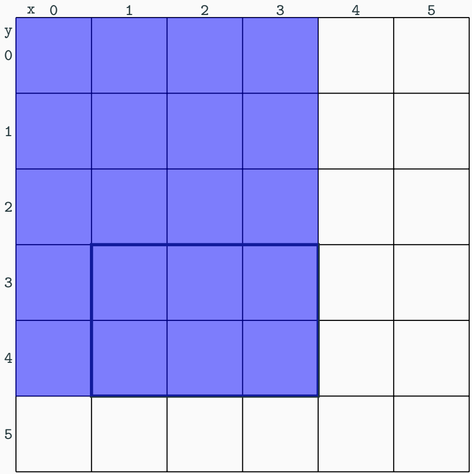
$$\begin{aligned}\mathrm{sum}(x_i, x_j, y_i, y_j) &= \mathrm{sum}(0, x_j, 0, y_j)\\ &- \mathrm{sum}(0, x_{i-1}, 0, y_j)\\ &- \mathrm{sum}(0, x_j, 0, y_{i-1})\\ &+ \mathrm{sum}(0, x_{i-1}, 0, y_{i-1})\end{aligned}$$

2D sum



`query(1, 3, 3, 4)`

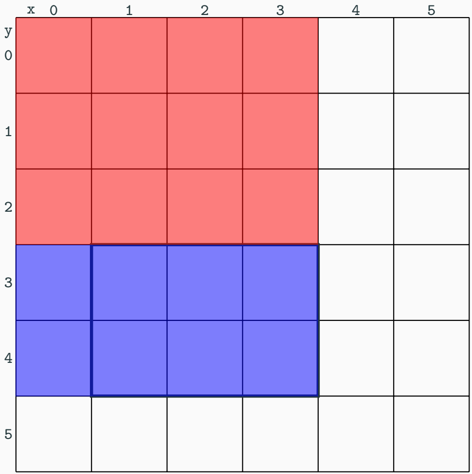
2D sum



`query(1, 3, 3, 4)`

`query(0, 3, 0, 4)`

2D sum

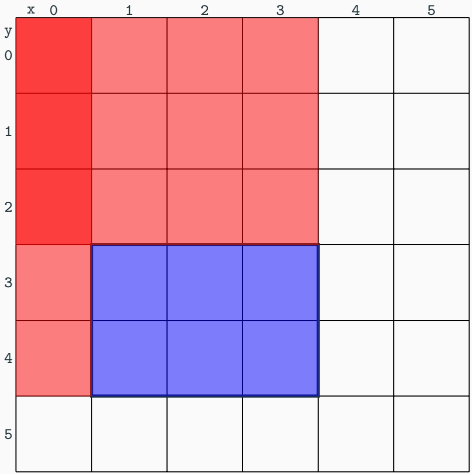


`query(1, 3, 3, 4)`

`query(0, 3, 0, 4)`

`query(0, 4, 0, 2)`

2D sum



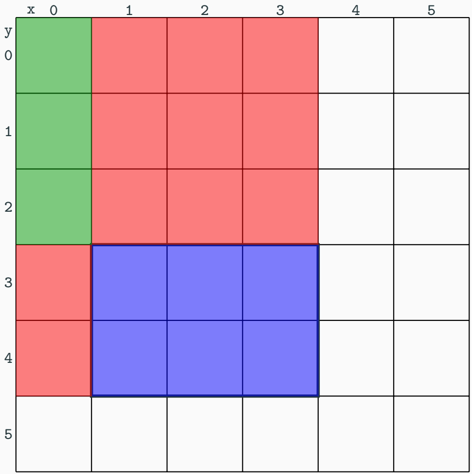
query(1, 3, 3, 4)

query(0, 3, 0, 4)

query(0, 4, 0, 2)

query(0, 0, 0, 4)

2D sum




```
query(1, 3, 3, 4)

query(0, 3, 0, 4)

query(0, 4, 0, 2)

query(0, 0, 0, 4)

query(0, 0, 0, 2)
```