

INF280: Competitive programming

Advanced datastructure algorithms

Louis Jachiet

Sliding windows

Typical examples using a list i_1, \dots, i_N and an integer K

- find $\max_j (i_j + \dots + i_{j+K})$

Typical examples using a list i_1, \dots, i_N and an integer K

- find $\max_j (i_j + \dots + i_{j+K})$
- find $\max_j (\max(i_j, \dots, i_{j+K}) - \min(i_j, \dots, i_{j+K}))$

Typical examples using a list i_1, \dots, i_N and an integer K

- find $\max_j (i_j + \dots + i_{j+K})$
- find $\max_j (\max(i_j, \dots, i_{j+K}) - \min(i_j, \dots, i_{j+K}))$
- find $\max_{j,\ell} (\ell - j \mid \max(i_j, \dots, i_\ell) - \min(i_j, \dots, i_\ell) < K)$

Typical examples using a list i_1, \dots, i_N and an integer K

- find $\max_j (i_j + \dots + i_{j+K})$
- find $\max_j (\max(i_j, \dots, i_{j+K}) - \min(i_j, \dots, i_{j+K}))$
- find $\max_{j,\ell} (\ell - j \mid \max(i_j, \dots, i_\ell) - \min(i_j, \dots, i_\ell) < K)$

Naive algorithm

Two (or three!) nested loops, recomputing from scratch for each position j .

Sliding window techniques to improve efficiency

Sliding window idea

Optimize away nested loops!

Example: fixed width (e.g. maintaining sum of K elements)

17	37	42	5	23	89	45	71	43	2	45	74	28	44	98
----	----	----	---	----	----	----	----	----	---	----	----	----	----	----

Sliding window techniques to improve efficiency

Sliding window idea

Optimize away nested loops!

Example: fixed width (e.g. maintaining sum of K elements)

-17 
+89

17	37	42	5	23	89	45	71	43	2	45	74	28	44	98
----	----	----	---	----	----	----	----	----	---	----	----	----	----	----

Sliding window techniques to improve efficiency

Sliding window idea

Optimize away nested loops!

Example: fixed width (e.g. maintaining sum of K elements)

A diagram illustrating a sliding window. It consists of two horizontal bars. The top bar is blue and is labeled '+45' at its right end. The bottom bar is green and is labeled '-37' at its left end. The blue bar is positioned above the green bar, and they overlap.

17	37	42	5	23	89	45	71	43	2	45	74	28	44	98
----	----	----	---	----	----	----	----	----	---	----	----	----	----	----

Sliding window techniques to improve efficiency

Sliding window idea

Optimize away nested loops!

Example: fixed width (e.g. maintaining sum of K elements)



17	37	42	5	23	89	45	71	43	2	45	74	28	44	98
----	----	----	---	----	----	----	----	----	---	----	----	----	----	----

Sliding window techniques to improve efficiency

Example: variable width (e.g. biggest total less than 100)

17	37	42	5	23	89	45	71	43	2	45	74	28	44	98
----	----	----	---	----	----	----	----	----	---	----	----	----	----	----

Sliding window techniques to improve efficiency

Example: variable width (e.g. biggest total less than 100)

17	37	42	5	23	89	45	71	43	2	45	74	28	44	98
----	----	----	---	----	----	----	----	----	---	----	----	----	----	----

Sliding window techniques to improve efficiency

Example: variable width (e.g. biggest total less than 100)

17	37	42	5	23	89	45	71	43	2	45	74	28	44	98
----	----	----	---	----	----	----	----	----	---	----	----	----	----	----

Sliding window techniques to improve efficiency

Example: variable width (e.g. biggest total less than 100)



17	37	42	5	23	89	45	71	43	2	45	74	28	44	98
----	----	----	---	----	----	----	----	----	---	----	----	----	----	----

Sliding window techniques to improve efficiency

Example: variable width (e.g. biggest total less than 100)

17	37	42	5	23	89	45	71	43	2	45	74	28	44	98
----	----	----	---	----	----	----	----	----	---	----	----	----	----	----

Sliding window techniques to improve efficiency

Example: variable width (e.g. biggest total less than 100)

17	37	42	5	23	89	45	71	43	2	45	74	28	44	98
----	----	----	---	----	----	----	----	----	---	----	----	----	----	----

Sliding window techniques to improve efficiency

Example: variable width (e.g. biggest total less than 100)

17	37	42	5	23	89	45	71	43	2	45	74	28	44	98
----	----	----	---	----	----	----	----	----	---	----	----	----	----	----

Sliding window techniques to improve efficiency

Example: variable width (e.g. biggest total less than 100)

17	37	42	5	23	89	45	71	43	2	45	74	28	44	98
----	----	----	---	----	----	----	----	----	---	----	----	----	----	----



Sliding window techniques to improve efficiency

Example: variable width (e.g. biggest total less than 100)

17	37	42	5	23	89	45	71	43	2	45	74	28	44	98
----	----	----	---	----	----	----	----	----	---	----	----	----	----	----

Sliding window techniques to improve efficiency

Example: variable width (e.g. biggest total less than 100)

17	37	42	5	23	89	45	71	43	2	45	74	28	44	98
----	----	----	---	----	----	----	----	----	---	----	----	----	----	----

Sliding window techniques to improve efficiency

Example: variable width (e.g. biggest total less than 100)

17	37	42	5	23	89	45	71	43	2	45	74	28	44	98
----	----	----	---	----	----	----	----	----	---	----	----	----	----	----

General idea

Maintain a double ended queue where:

- you add on the right to grow
- remove on the left to shrink
- maintain some computation over the window content

Works with **monotone** criteria for windows!

General idea

Maintain a double ended queue where:

- you add on the right to grow
- remove on the left to shrink
- maintain some **computation** over the window content

Works with **monotone** criteria for windows!

Advanced technique for sliding window

The deque trick to maintain min and max

Maintain the (ordered) list of elements that might become min/max.

Example: maintaining min of 5 elements)

17	37	42	5	23	89	45	71	43	2	35	74	28	44	98
----	----	----	---	----	----	----	----	----	---	----	----	----	----	----

Updates

- Add on the right: remove everything bigger
- Remove on the left: remove when min

Advanced technique for sliding window

The deque trick to maintain min and max

Maintain the (ordered) list of elements that might become min/max.

Example: maintaining min of 5 elements)

Candidate mins: 5, 23

17	37	42	5	23	89	45	71	43	2	35	74	28	44	98
----	----	----	---	----	----	----	----	----	---	----	----	----	----	----

Updates

- Add on the right: remove everything bigger
- Remove on the left: remove when min

Advanced technique for sliding window

The deque trick to maintain min and max

Maintain the (ordered) list of elements that might become min/max.

Example: maintaining min of 5 elements)

Candidate mins: 5, 23, 89

17	37	42	5	23	89	45	71	43	2	35	74	28	44	98
----	----	----	---	----	----	----	----	----	---	----	----	----	----	----

Updates

- Add on the right: remove everything bigger
- Remove on the left: remove when min

Advanced technique for sliding window

The deque trick to maintain min and max

Maintain the (ordered) list of elements that might become min/max.

Example: maintaining min of 5 elements)

Candidate mins: 5, 23, 45

17	37	42	5	23	89	45	71	43	2	35	74	28	44	98
----	----	----	---	----	----	----	----	----	---	----	----	----	----	----

Updates

- Add on the right: remove everything bigger
- Remove on the left: remove when min

Advanced technique for sliding window

The deque trick to maintain min and max

Maintain the (ordered) list of elements that might become min/max.

Example: maintaining min of 5 elements)

Candidate mins: 5, 23, 45, 71

17	37	42	5	23	89	45	71	43	2	35	74	28	44	98
----	----	----	---	----	----	----	----	----	---	----	----	----	----	----

Updates

- Add on the right: remove everything bigger
- Remove on the left: remove when min

Advanced technique for sliding window

The deque trick to maintain min and max

Maintain the (ordered) list of elements that might become min/max.

Example: maintaining min of 5 elements)

Candidate mins: 23, 43

17	37	42	5	23	89	45	71	43	2	35	74	28	44	98
----	----	----	---	----	----	----	----	----	---	----	----	----	----	----

Updates

- Add on the right: remove everything bigger
- Remove on the left: remove when min

Advanced technique for sliding window

The deque trick to maintain min and max

Maintain the (ordered) list of elements that might become min/max.

Example: maintaining min of 5 elements)

Candidate mins: 2

17	37	42	5	23	89	45	71	43	2	35	74	28	44	98
----	----	----	---	----	----	----	----	----	---	----	----	----	----	----

Updates

- Add on the right: remove everything bigger
- Remove on the left: remove when min

Advanced technique for sliding window

The deque trick to maintain min and max

Maintain the (ordered) list of elements that might become min/max.

Example: maintaining min of 5 elements)

Candidate mins: 2, 35

17	37	42	5	23	89	45	71	43	2	35	74	28	44	98
----	----	----	---	----	----	----	----	----	---	----	----	----	----	----

Updates

- Add on the right: remove everything bigger
- Remove on the left: remove when min

Advanced technique for sliding window

The deque trick to maintain min and max

Maintain the (ordered) list of elements that might become min/max.

Example: maintaining min of 5 elements)

Candidate mins: 2, 35, 74

17	37	42	5	23	89	45	71	43	2	35	74	28	44	98
----	----	----	---	----	----	----	----	----	---	----	----	----	----	----

Updates

- Add on the right: remove everything bigger
- Remove on the left: remove when min

Advanced technique for sliding window

The deque trick to maintain min and max

Maintain the (ordered) list of elements that might become min/max.

Example: maintaining min of 5 elements)

17	37	42	5	23	89	45	71	43	2	35	74	28	44	98
----	----	----	---	----	----	----	----	----	---	----	----	----	----	----

Updates

- Add on the right: remove everything bigger
- Remove on the left: remove when min

Also works with a set... with a $O(\log(n))$ penalty.

Prefix sums

Structure to compute sums in $O(1)$

Input

A list of elements $v_1 \dots v_n$ over a group

Query

Compute $q(i, j) = \sum_{i \leq l < j} v_l$

Structure to compute sums in $O(1)$

Input

A list of elements $v_1 \dots v_n$ over a group

Query

Compute $q(i, j) = \sum_{i \leq l < j} v_l$

Solution

Precompute $T[i] = \sum_{l < i} v_l$, $q(i, j) = T[j] - T[i]$

Structure to compute sums in $O(1)$

Input

A list of elements $v_1 \dots v_n$ over a group

Query

Compute $q(i, j) = \sum_{i \leq \ell < j} v_\ell$

Solution

Precompute $T[i] = \sum_{\ell < i} v_\ell$, $q(i, j) = T[j] - T[i]$

Works in d dimensions!