CS 598 TMC Advanced Data Structures (F'23)

courses engr illinois edu/cs598-tmc

Course Work:
- 4 HWS 45%
- Presentation 15% (may work in groups of ≤3)
- Project 40%

Prerequisite: strong background in algorithms (CS374)

This is a theory course!

No textbook

Course Topics:

1. Basics (BST, heaps union-find, ...)
2. Integers (hashing, VEB trees, fusion trees...)
3. Geometry (orthogonal range search, point location, ANN/LSH, ...)
4. Graphs (dynamic connectivity, distance oracles, ...)
5. Strings (suffix trees/arrays, ...)
6. Other models (succinct DS, higher memory DS)
5. Other models (succinct DS, external memory DS, streaming/sketching, ...)

Problem (Range Min Queries (RMQ))
Given sequence of n numbers \(a_1, \ldots, a_n\), build a data structure to answer following query:

\[ \text{given } i, j, \text{ find min of } a_i, \ldots, a_j. \]

5, 3, 8, 9, 4, 10, 7

[static problem]
To bound: space, preprocessing time, query time

[Other variants: dynamic (update time), range median/mode, 2D, ...]

Method 1:
Preproc time \(O(1)\)
Space \(O(n)\)
Query time \(O(n)\)

Method 2:
Precompute all answers in table
Space \(O(n^2)\)
Query \(O(1)\)
Preproc time \(O(n^2)\)
Method 3:
Divide into $\sqrt{n}$ blocks of size $\sqrt{n}$, precompute min of each block.

Space/Preproc time: $O(n)$
Query time: $O(\sqrt{n})$

Method 4:
Tree

At each node, store min of subtree.

Preproc($a_L, a_R)$:
1. left. preproc($a_L, a_{\frac{L+R}{2}}$)
2. right. preproc($a_{\frac{L+R}{2}+1}, a_R$)
3. $m^* = \min(\text{left.} m^*, \text{right.} m^*)$

query($i, j$):
- if $[i, j] \cap [L, R] = \emptyset$ return $\infty$
- if $[i, j] \supseteq [L, R]$ return $m^*$
- return $\min(\text{left. query}(i, j), \text{right. query}(i, j))$

Space: $O(n)$
Preproc time: $O(n)$
Method 5

Divide into $\sqrt{n}$ blocks of size $\sqrt{n}$, precompute answers for $(i,j)$ within each block.

For each block boundary $l$, precompute ans for $(l,i)$ & for $(i,l)$.

Space/proc time $O(n^{3/2})$

Query time $O(1)$
**Method 6:** tree

At each node, store all prefix mins & suffix mins.

\[
\text{preproc}(a, l, r):
\]
\[
\text{left: } \text{preproc}(a, l, a[l])
\]
\[
\text{right: } \text{preproc}(a[r+1], r)
\]
\[
\text{for } j = l \text{ to } r, \quad m^+(j) = \min(a_l, \ldots, a_j)
\]
\[
\text{for } i = l \text{ to } r, \quad m^-(i) = \min(a_i, \ldots, a_r)
\]

Space/preproc time \( \mathcal{O}(n \log n) \),

\( S(n) = 2S(\frac{n}{2}) + \mathcal{O}(n) \)

\[
\text{Query}(i, j):
\]
\[
\text{if } j \leq \frac{l + r}{2}, \quad \text{return left. query}(i, j)
\]
\[
\text{if } i > \frac{l + r}{2}, \quad \text{return right. query}(i, j)
\]
\[
\text{return } \min(\text{left. } m^-(i), \text{right. } m^+(j))
\]

Query time \( \mathcal{O}(\log n + 1) = \mathcal{O}(1) \).

**Method 7:** bootstrap

Divide into \( \frac{n}{b} \) blocks of size \( b \).
Divide into \( \frac{n}{b} \) blocks of size \( b \).

Store prefix/suffix mins inside each block \( \leq O(n) \).

Use Method 6 inside each block \( \leq O\left(\frac{n}{b} \cdot \log b\right) \).

Use Method 6 for the mins of all blocks \( \leq O\left(\frac{n}{b^2} \log \frac{n}{b}\right) \).

Query time \( O(1) \).

Space/prep time \( O(n + \frac{n}{b} \log b + \frac{n}{b^2} \log \frac{n}{b}) \).

Set \( b = \log n \) \( \Rightarrow \) \( O(n \log \log n) \).