On Updating and Querying Submatrices

Jason Yang
Mentor: Jun Wan

MIT PRIMES Computer Science

October 19, 2020
Range update-query problem

- A is an array of $N$ numbers
- A range $R = [l, r]$ is the set of indices $\{i | l \leq i \leq r\}$
- $query(R)$: return $\min_{i \in R} A[i]$

Segment tree + lazy propagation: $O(\log N)$ time updates and queries
Generalizations

Using different operators

- \( \text{update}(R, v) : \forall i \in R, A[i] \leftarrow A[i] \triangleleft v \)
- \( \text{query}(R, v) : \text{return } \bigtriangleup_{i \in R} A[i] \)

If \( \triangleleft \) and \( \bigtriangleup \) are associative, segment tree + lazy propagation usually works (but not always)

Ex. \( (\triangleleft, \bigtriangleup) = \)
- \((+, +)\)
- \((\ast, +)\)
- \((\leftarrow, \text{min})\)

This problem and variants have applications in

- LCA in a tree
- image retrieval
2 dimensions:

- the array becomes a matrix
- ranges \( \{ i | l \leq i \leq r \} \) becomes submatrices

\[
[l_0, r_0][l_1, r_1] = \{ i | l_0 \leq i \leq r_0 \} \times \{ j | l_1 \leq j \leq r_1 \}
\]

We call this the **submatrix update-query problem**.
Generalizing segment tree seems to be very difficult

<table>
<thead>
<tr>
<th></th>
<th>update</th>
<th>query</th>
</tr>
</thead>
<tbody>
<tr>
<td>$d = 1$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Segment Tree</td>
<td>$O(\log N)$</td>
<td>$O(\log N)$</td>
</tr>
<tr>
<td>$d = 2$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2D Segment Tree</td>
<td>$O(N \log N)$</td>
<td>$O(\log^2 N)$</td>
</tr>
<tr>
<td>Quadtree</td>
<td>$O(N)$</td>
<td>$O(N)$</td>
</tr>
<tr>
<td>$d = 2$, special operator pairs ($\nabla$, $\triangle$)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2D Fenwick Tree (Mishra)</td>
<td>$O(16 \log^2 N)$</td>
<td>$O(16 \log^2 N)$</td>
</tr>
<tr>
<td>2D Segment Tree (Ibtehaz)</td>
<td>$O(\log^2 N)$</td>
<td>$O(\log^2 N)$</td>
</tr>
<tr>
<td>2D Segment Tree (ours)</td>
<td>$O(\log^2 N)$</td>
<td>$O(\log^2 N)$</td>
</tr>
</tbody>
</table>
Why is generalizing the segment tree difficult?
A binary tree of nodes:

- each node $n$ covers a range $n_R$ and contains a value $n_V = \min_{i \in n_R} A[i]$.

When querying any range, we only have to look at $O(\log N)$ nodes.

query([2, 12]) = \min(30, 4, 13, 14) = 4
**Segment Tree: Updates**

\[ \text{update}(R, v): \text{ change } n_V \text{ for all } n \text{ that overlap with } R \]

\[ \Rightarrow O(N) \text{ nodes in worst-case} \]

\[ \text{update}([1,10], 20) \]
Segment Tree: Updates

- For all \( n \) s.t. \( n_R \subseteq R \) (shown as green), \( n_V \) simply changes to \( n_V + v \)
- Split green nodes into \( O(\log N) \) subtrees
- Attach a “lazy label” \( t_Z \) to every node \( t \)
  - \( t_Z \) represents the command “\( n_V \leftarrow n_V + t_Z \) \( \forall n \) in subtree at \( t \)”
- For each subtree, increase its root node’s lazy label by \( v \)
Segment Trees: Updates

- For each $n$ s.t. $(n_R \cap R \neq \emptyset) \land (n_R \not\subseteq R)$ (shown as yellow) in greatest-to-lowest depth, do
  \[
  n_V \leftarrow \min((n_l)_V + (n_l)_Z, (n_r)_V + (n_r)_Z)
  \]
- Only $O(\log N)$ many such nodes

![Diagram showing segment tree update]
Segment Trees: Queries revised

- When looking at $n_V$ from $n$, we must add all lazy values that affect it
  - We must use $n_V + \sum_{m \supseteq n} m_Z$ instead of just $n_V$
- $\Rightarrow O(\log^2 N)$ time queries (because we look at $O(\log N)$ nodes)
  - Can be improved to $O(\log N)$ time

```
query([4,5]) = 17 + 20 + 0 + 0 = 37
```
A segment tree of segment trees:

- Construct segment tree across rows of $N \times M$ matrix $A$
- Each node $n$ in this segment tree contains a segment tree $n_T$ constructed over the array $B = \text{eltwise-min}_{i \in n_R} A[i]$

\[
(\nabla, \triangle) = (+, \min)
\]

We can do queries in $O(\log N \log M)$ time:

\[
\text{query}(R_X \times R_Y) = \min_{n \in S(R_X)} n_T.\text{query}(R_Y)
\]
2D segment tree

But updates are difficult...

Another problem: lazy propagation
2D Segment tree

By only using lazy propagation in inner segment trees, we do updates in $O(N \log M + M \log N)$ time and queries in $O(\log N \log M)$ time.
Impossible?

Perhaps it is impossible to get $O(\text{polylog}(N))$ time updates and queries?
Min-plus matrix multiplication

Given $N \times N$ matrices $A, B$, min-plus product is

$$C_{i,j} = \min_{0 \leq k < N} (A_{i,k} + B_{k,j})$$

Min-plus matrix multiplication is known to be equivalent to all-pairs shortest paths
Reducing min-plus matrix multiplication to submatrix update-query

\[ C_{0,0} = \min(A_{0,0} + B_{0,0}, A_{0,1} + B_{1,0}, \cdots, A_{0,N-1} + B_{N-1,0}) \]
\[ C_{1,0} = \min(A_{1,0} + B_{0,0}, A_{1,1} + B_{1,0}, \cdots, A_{1,N-1} + B_{N-1,0}) \]
\[ \cdots \]
\[ C_{N-1,0} = \min(A_{N-1,0} + B_{0,0}, A_{N-1,1} + B_{1,0}, \cdots, A_{N-1,N-1} + B_{N-1,0}) \]
Reducing min-plus matrix multiplication to submatrix update-query

<table>
<thead>
<tr>
<th>$C_{0,0}$</th>
<th>$A_{0,0} + B_{0,0}$</th>
<th>$A_{0,1} + B_{1,0}$</th>
<th>\cdots</th>
<th>$A_{0,N-1} + B_{N-1,0}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$C_{1,0}$</td>
<td>$A_{1,0} + B_{0,0}$</td>
<td>$A_{1,1} + B_{1,0}$</td>
<td>\cdots</td>
<td>$A_{1,N-1} + B_{N-1,0}$</td>
</tr>
<tr>
<td>$\vdots$</td>
<td>$\vdots$</td>
<td>$\vdots$</td>
<td></td>
<td>$\vdots$</td>
</tr>
<tr>
<td>$C_{N-1,0}$</td>
<td>$A_{N-1,0} + B_{0,0}$</td>
<td>$A_{N-1,1} + B_{1,0}$</td>
<td>\cdots</td>
<td>$A_{N-1,N-1} + B_{N-1,0}$</td>
</tr>
</tbody>
</table>
Reducing min-plus matrix multiplication to submatrix update-query

<table>
<thead>
<tr>
<th>$C_{0,0}$</th>
<th>$A_{0,0} + B_{0,0}$</th>
<th>$A_{0,1} + B_{1,0}$</th>
<th>\cdots</th>
<th>$A_{0,N-1} + B_{N-1,0}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$C_{1,0}$</td>
<td>$A_{1,0} + B_{0,0}$</td>
<td>$A_{1,1} + B_{1,0}$</td>
<td>\cdots</td>
<td>$A_{1,N-1} + B_{N-1,0}$</td>
</tr>
<tr>
<td>\vdots</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$C_{N-1,0}$</td>
<td>$A_{N-1,0} + B_{0,0}$</td>
<td>$A_{N-1,1} + B_{1,0}$</td>
<td>\cdots</td>
<td>$A_{N-1,N-1} + B_{N-1,0}$</td>
</tr>
</tbody>
</table>
Reducing min-plus matrix multiplication to submatrix update-query


<table>
<thead>
<tr>
<th></th>
<th>$+B_{0,0}$</th>
<th>$+B_{1,0}$</th>
<th>$\cdots$</th>
<th>$+B_{N-1,0}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$C_{0,0}$</td>
<td>$A_{0,0}$</td>
<td>$A_{0,1}$</td>
<td></td>
<td>$A_{0,N-1}$</td>
</tr>
<tr>
<td>$C_{1,0}$</td>
<td>$A_{1,0}$</td>
<td>$A_{1,1}$</td>
<td></td>
<td>$A_{1,N-1}$</td>
</tr>
<tr>
<td>$\vdots$</td>
<td></td>
<td></td>
<td>$\vdots$</td>
<td></td>
</tr>
<tr>
<td>$C_{N-1,0}$</td>
<td>$A_{N-1,0}$</td>
<td>$A_{N-1,1}$</td>
<td></td>
<td>$A_{N-1,N-1}$</td>
</tr>
</tbody>
</table>

- $N$ elements of $C$ can be found with $N$ submatrix updates and $N$ submatrix queries.
- We can then undo all updates and use different elements of $B$ to get $N$ other elements of $C$, and then repeat this.
Reducing min-plus matrix multiplication to submatrix update-query

1: Initialize $(+, \min)$ update-query DS with $A$
2: \textbf{for} $j = 0$ to $N - 1$ \textbf{do}
3: \hspace{1em} update$([0, N - 1][k, k], B[k][j]) \forall 0 \leq k < N$
4: \hspace{1em} $C[i][j] \leftarrow \text{query}([i, i][0, N - 1]) \forall 0 \leq i < N$
5: \hspace{1em} update$([0, N - 1][k, k], -B[k][j]) \forall 0 \leq k < N$

Runs in $O(P(N) + N^2(U(N) + Q(N))$ time, where $P(N), U(N), Q(N)$ are worst-case preprocessing, update, and query times resp. over a $N \times N$ matrix
We can replace matrix of an update-query data structure to $A_1$ by doing:

$$update([i, i][j, j], -Q([i, i][j, j]) + A_1[i][j]) \forall 0 \leq i, j < N$$

$\Rightarrow$ We can find many matrix multiplications while initializing only once.
Lower bounds

- Product of two $KN \times KN$ matrices
  - $\Rightarrow$ block matrix product of two $K \times K$ matrices where each element is a $N \times N$ matrix instead of a number
  - $\Rightarrow O(K^3)$ many $N \times N$ matrix multiplications using schoolbook algorithm
  - $\Rightarrow KN \times KN$ min-plus matrix product in $O(P(N) + K^3 N^2(U(N) + Q(N)))$ time
Main theorem

- $N \times N$ min-plus matrix multiplication widely believed to not have $O(N^{3-\varepsilon})$ time solution
- If true, then
  \[ O(P(N) + K^3 N^2 (U(N) + Q(N))) > O((KN)^{3-\varepsilon}) \quad \forall \varepsilon > 0 \]

Theorem

If min-plus matrix multiplication cannot be done in $O(N^{3-\varepsilon})$ time, then either $U(N)$ or $Q(N) > O(N^{1-\varepsilon})$ for any $\varepsilon > 0$, or $P(N)$ is superpolynomial.

- A quadtree has $O(N)$ time updates and queries and $O(N^2)$ time preprocessing.
- Thus, our lower bound is tight up to $o(N^{\varepsilon})$ factors.
For submatrix updates and queries:

- Is sublinear (ex. $O\left(\frac{N}{\log N}\right)$) update and query time w/ polynomial preprocessing time possible for $([\nabla, \triangle] = (+, \min)$?

- Are $O(\log N \log M)$ time updates and queries possible for more operator pairs?
  - i.e. beyond cases where $\nabla = \triangle$ and $\nabla$ is commutative and associative (ex. min, +, *, AND)

- If $\nabla$ is noncommutative, are $O(\text{poly}(N, M))$ updates and queries possible at all?
  - 1D case solved with segment tree + lazy propagation (but lazy part is more complex)
Acknowledgments

I would like to thank

- Jun Wan for his mentorship
- Dr. Gerovitch and MIT PRIMES for making this project possible
- My parents for their support
- You for listening
References


Stabbing Queries.

References


Segment Tree - Competitive Programming Algorithms
https://cp-algorithms.com/data_structures/segment_tree.html

Animesh Fatehpuria. 2DRangeSumQuerywithUpdates.cpp
