Segment Trees

RARES CRISTIAN

Introductory Problem: Range Sum Queries

- Given:
  - Array $A = \{a_1, a_2, \dots, a_n\}$
  - Queries $f(j,k) = \sum_{i=j}^{k} a_i$

- Naïve Approach:
  - For each query, iterate from index j to k, computing the sum.
  - Time Complexity of Query: $O(n)$

- Solution: Prefix Sums
  - Let $P = \{p_1, p_2, \dots, p_n\}$ where
  - $p_k = \sum_{i=1}^{k} a_i$
  - Efficiently compute P iteratively:
    - $p_k = p_{k-1} + a_k$
    - $p_0 = 0$
    - Time Complexity Preprocessing: $O(n)$
  - Answer Queries: $f(j,k) = p_k - p_{j-1}$
    - Time Complexity Query: $O(1)$

Adding Updates

New query type: Updates

- $U(i, v) \rightarrow a_i \coloneqq v$

Solving with Prefix Sums

- Answering queries remains the same
- Handling Updates:
  - Update value of p_j for all j $\geq i$
  - $O(n)$

What we Want

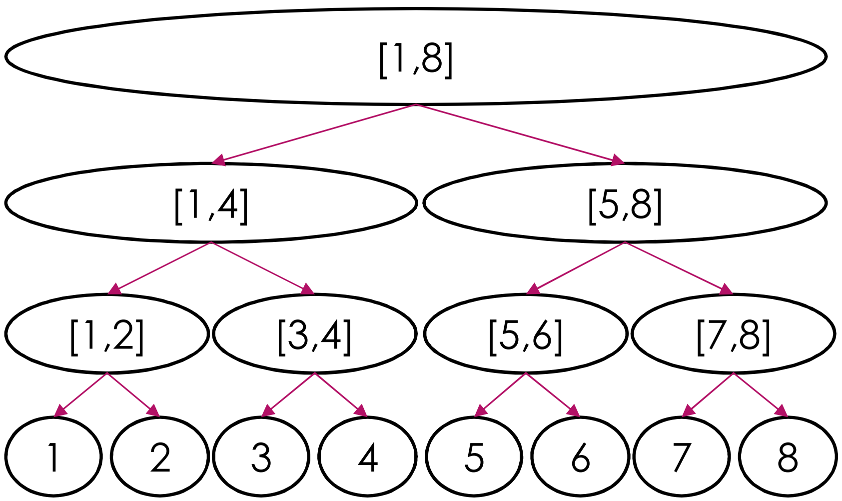
- For all possible queries, Q, minimize the number of ranges in P needed to compute $f(Q)$.
  - (Moving forward, P is the set of ranges we precompute the answer to)
- For any given element $a \in A$, minimize the number of elements in P which contain a.

Possible Functions

- Prefix Approach:
  - For sum queries, $f(j,k) = p_k - p_{j-1}$
  - Or, $f(j,k) = f(0,k) - f(0,j-1)$
  - There needs to be some notion of an inverse.
- Many problems to do not have an easily computable inverse.
  - Consider Maximum of range.

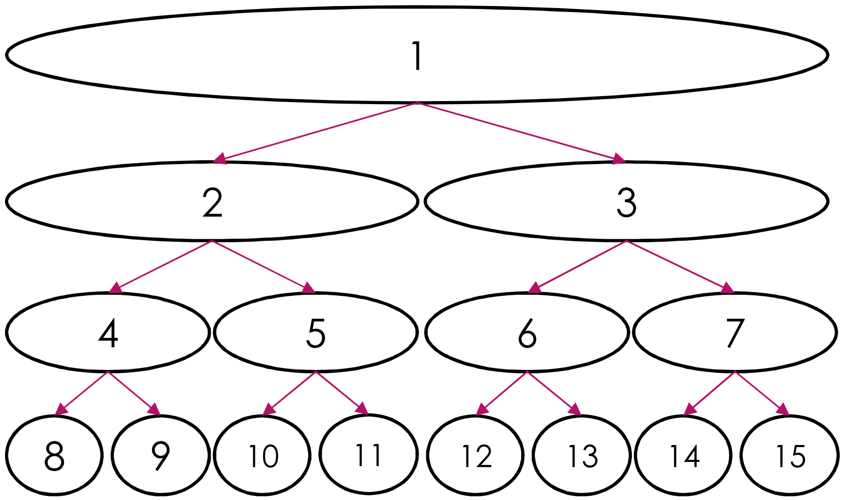
- Our New Approach
  - Say we partitioned $[i,k]$ into $[i,j]$ and $[j+1,k]$
  - We require $f(i,k) = M(f(i,j), f(j+1,k))$ for some function M.
  - M represents merging two ranges together.

A Segment Tree



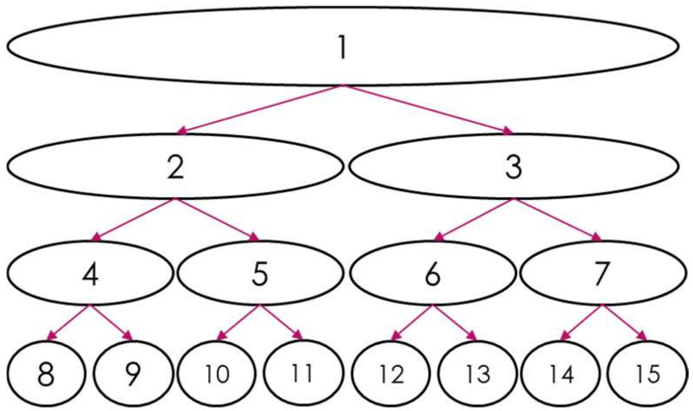
Some Properties

- Height: $\log(N)$
  - Every $a_i \in A$ is contained in $\log(N)$ elements of P
- Total of $2n - 1$ elements in the Tree
- Node $2i$ is the left child of i
- Node $2i + 1$ is the right child of i



Updating Elements

- Algorithm $U(p \in P, i, v)$
  - Want $U(p, i, v)$ to return $f(p)$ given $a_i = v$
- Initial Call to $U(p = root, i, v)$
  - If $p = \{a_i\}$, set a_i to v; return $f(p) = v$
  - If $a_i \notin p$, return $f(p)$
  - If $a_i \in p$,
    - set $f(p)$ equal to $M\big(U(p \to left, i, v), U(p \to right,\ i, v)\big)$
    - Return $f(p)$
- Time Complexity:
  - Path from root to leaf
  - $\log(N)$



Finding the Partition: An Algorithm

- Terminology:
  - Given a query range Q, and a range $p \in P$, we say p 'expands Q' if $\exists x, y \in p$ such that $x \in Q$ and $y \notin Q$.
- Algorithm $F(p \in P, Q) = f(p \cap Q)$
- Initial call to $F(p = root, Q)$
  - If p is completely contained in Q (that is, $p \cap Q = p$), return $f(p)$
  - If $p \cap \mathrm{Q} = \emptyset$, return neutral element.
  - If p expands Q, return $M(F(p \rightarrow left\ child, Q),\ F(p \rightarrow right\ child, Q))$.

Queries: Time Complexity

- Claim: At most two nodes at the same depth expand Q.
- Proof by contradiction
  - Assume three nodes at the same depth expand $Q = [i, k]$.
  - All contiguous ranges, so to expand, a range contains either $i \text{ and } i - 1$ or $k \text{ and } k + 1$.
  - Pigeonhole principle → two ranges must contain same elements, a contradiction.
- At most 2·(height of Tree) expanded nodes
  - $O(\log(N))$ Query time

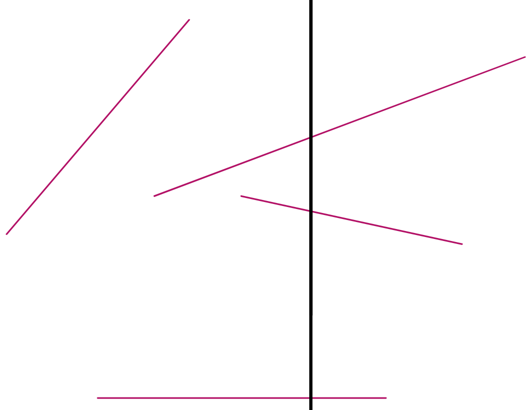
Questions?

… If not, we'll move on to the original segment tree problem

Segment Intersection

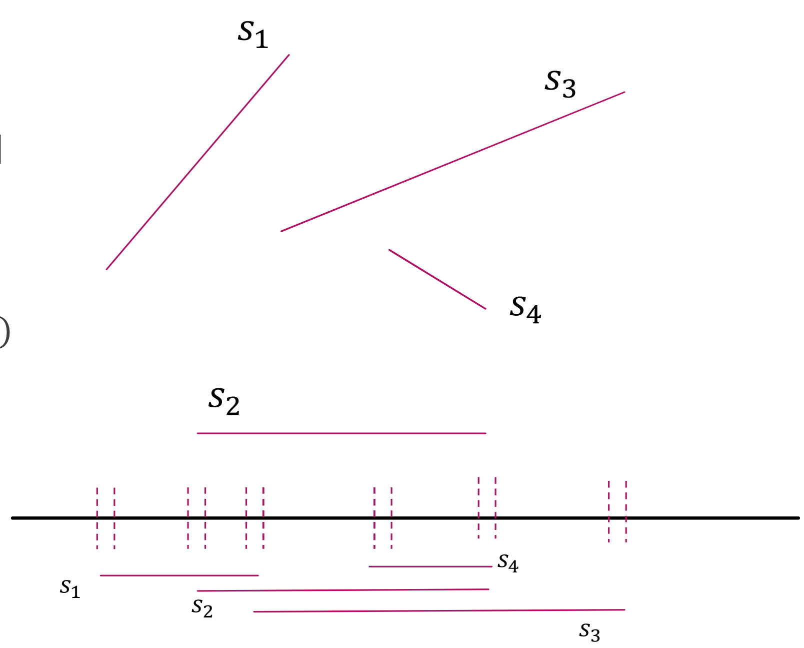
The Problem

- Given:
  - A static set of V nonintersecting non-horizontal line segments.
- Queries:
  - Given a vertical line segment, s, find which segments in V intersect s.
- Easier Subproblem:
  - Stabbing Query: Given vertical line, find which segments in V it intersects.

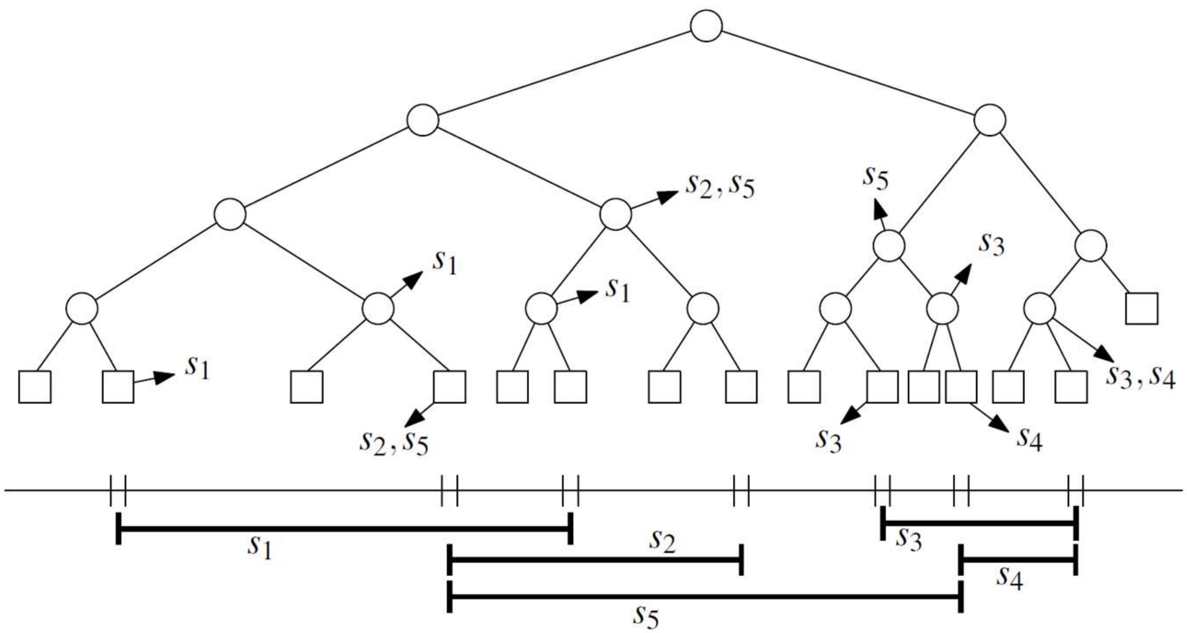


Elementary Intervals

- Sort x-coordinates of all segments.
  - $p_1, p_2, \ldots, p_m$
- Consider the partitioning of the real line induced by the points p_i.
  - Call these elementary intervals:
  - $(-\infty, p_1), [p_1, p_1], (p_1, p_2), \ldots, (p_{m-1}, p_m), [p_m, p_m], (p_m, \infty)$
- For each elementary interval, we can store which segments lie within it.
  - Might be $O(n^2)$ storage

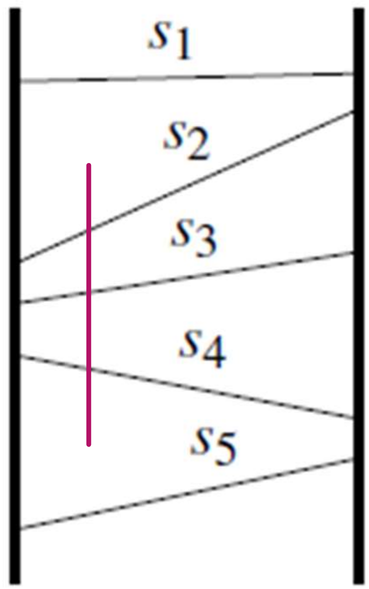


Segment Tree Solution (Subproblem)



Back to the Original Problem

- May have many candidate intersections.
- For a segment s to be listed in a node, s must completely span the range of the node.
- Zooming in on a single node.
  - Binary Search for lowest segment which is higher than the bottom of the query.



The End