# Multi-Dimensional Queries and Segtree Walks

Spencer Compton

CPSI 2018: Squires Lecture 9

## 1 Introduction

USACO and other contests like to include advanced data structures, this frequently includes queries where multiple dimensions are involved or the nature of segment trees are utilized with "walks".

## 2 Segment Tree Walks

Consider the following problem where you supposed the following queries:

- $Insert(L, R, X)$: Add $X$ occurrences of all numbers in range $[L, R]$ to your multiset.
- $get(K)$: Print out the $Kth$ smallest element in your multiset.

One might consider approaching this by using a lazy-propagation segment tree that supports queries in the form $Add(L, R, X)$ and $Sum(L, R)$, using a binary search to get this answer in $O(\log^2 N)$. However, if we "walk" down the segment tree with one query, we can do this in $O(\log N)$

## 3 Sweeping to Remove a Dimension

Consider we are given an array and want to count the number of values with indices in range $[L, R]$ that are $<= K$. We can view this with a geometric interpretation, with array elements graphed as $(i, A[i])$. This transforms our problem into counting the number of points in a rectangle. We can use a sweep and a segment tree to accomplish this. We sweep from left to right as we process queries, thus removing the x-dimension. Now, our segment tree only needs to hold information about the points' y-dimension. This works for answering the queries offline in $O((N + Q) \times \log{(N + Q)})$

## 4 Merge Sort Tree

What if we are given the same queries, but want to process the queries online (i.e. we can't sort the queries beforehand)? We can use a merge sort tree. A merge sort tree essentially is just the segments that you sort when you do a merge sort, but we keep all the intermediate segments. If we want to query the number of elements in range $[L, R]$ that are $<= K$, then we can view our merge sort tree similarly to a segment tree. There are $O(\log N)$ segments completely covered that we care about for this query in range $[L, R]$. Since each segment is already sorted, we can binary search to find the number of elements in it that are $<= K$. This gives us a runtime of $O(N \log N)$ preprocessing, $O(\log^2 N$ queries, and $O(N \log N)$ memory.

## 5 Segment Tree + Treap = Segtreap?

What if we want to answer these queries, but we also want to update or change values in the array between queries? Consider the same idea as a Merge Sort Tree, except instead of maintaining an array inside each segment, maintain a treap (or use Policy-Based Data Structures, if this works and you are familiar with them). This allows us to remove and insert an element from a segment in $O(\log N)$. Treaps also allow us to find the number of elements $<= K$ for a certain segment in $O(\log N)$. Since for each query we need to consider $O(\log N)$ segments and in each segment we do $O(\log N)$ work with our treap. We have a runtime of $O(N \log^2 N)$ preprocessing , $O(\log^2 N$ queries, and $O(N \log N)$ memory.

## 6 Tips

- If you can get rid of a dimension (e.g. with sweeps) this will make your life a LOT easier than trying to use mutli-dimensional data structures.
- Combining Data Structures (such as Segment Tree + Treap = Segtreap) can be helpful.
- We don't just have to store sum, min, or lazy values in our segment tree intervals. We can store a variety of things, such as: arrays, treaps, other values, etc.
- Coordinate Compression can often be used to make your life easier.
- On-Demand Segment Trees are also possible, but try use coordinate compression instead if it works in the context of the problem.

## 7 Practice Problems

USACO Platinum 2016 January: Mowing the Field

USACO Platinum 2017 February: Why Did the Cow Cross the Road III

## 8 Topics to Explore

- Policy-Based Data Structures (if you use C++, learning this is potentially VERY helpful)
- Persistent Segment Trees (can be helpful as well, but if you use C++ the former is much higher priority)

## 9 Resources

Policy-Based Data Structures

https://codeforces.com/blog/entry/11080

Anudeep Persistent Segment Tree Tutorial

https://blog.anudeep2011.com/persistent-segment-trees-explained-with-spoj-problems/

Another Persistent Segment Tree Tutorial

https://codeforces.com/blog/entry/49777