15-451/651 Algorithm Design & Analysis, Spring 2025

Recitation #5

Objectives

- Understand the SegTree data structure and use it to speed up algorithms
- Use custom associative operators in SegTrees

Recitation Problems

1. **(SegTrees for RangeAdd)** Suppose we want to design a data structure to support the following API over an array of n integers, all initially zero.

   - **RangeAdd**(i, j, x): Add x to every integer in the array from positions i (inclusive) to j (exclusive).
   - **Get**(i): Return the value at index i.

   Implement this data structure by *reducing* it to the API of a SegTree.

2. **(Abby's Favorite Problem)** Suppose we start with some array of integers A. Given a sequence of query intervals in the form $[l_1, r_1), [l_2, r_2), ..., [l_m, r_m)$, return the maximum element and how many times it appears in $A[l_i], ..., A[r_i - 1]$ in $O(\log n)$ time for each query $[l_i, r_i)$. You can use $O(n)$ time for preprocessing.

3. **(Crossing intervals)** Suppose we have a list of n intervals $I_i = [a_i, b_i]$ where $0 \le a_i, b_i < 2n$, such that the endpoints of all of the intervals are distinct (no two intervals ever share an endpoint at either end). A pair of intervals I_i and I_j are *crossing* if they overlap but one does not strictly contain the other. We want to devise an algorithm to count the number of pairs of crossing intervals.

   (a) Give a simple $O(n^2)$ algorithm for the problem

   (b) Come up with a more efficient $O(n \log n)$ algorithm by making use of a SegTree