1 Introduction

In many cases, we can use existing data structures in unexpected ways by augmenting the data they store.

Basic steps:

1. Choose an underlying data structure.
2. Determine additional information to maintain in that data structure.
3. Modify the operations of that data structure to maintain that information.
4. Develop new operations using that information.

This approach is much more common than “starting from scratch” with a new data structure.

Many examples are based on balanced search trees.

2 Dynamic order statistics

Suppose we want a data structure that supports these operations:

- \text{INSERT}(k)
- \text{SEARCH}(k)
- \text{DELETE}(k)
- \text{SELECT}(i) — find the \(i\)th smallest element
- \text{RANK}(v) — how many elements are smaller than the one at node \(v\)?

This differs from the standard selection problem, because the set is dynamic — elements may be added or deleted.

We can form a data structure that supports these operations by augmenting your favorite rotation-based balanced binary search tree data structure.
3 Order statistic trees
In addition to the usual attributes (key, left, right, parent), add a new attribute:

- Store the number of nodes in the subtree rooted at \( v \) as \( v.\text{size} \).

4 Maintaining the size attribute: INSERT
For each INSERT: Increment the size of each node along the way.

5 Maintaining the size attribute: ROTATE
Each rotation changes the size for only two nodes: The two nodes incident to the edge being rotated.

6 SELECT in Order Statistic Trees
The SELECT operation in an order statistic tree looks much like QUICKSELECT:

\[
\text{OST-SELECT}(v, i) \\
\quad r = v.\text{left}.\text{size} + 1 \\
\quad \text{if } i = r \text{ then} \\
\quad \quad \text{return } v \\
\quad \text{else if } i < r \text{ then} \\
\quad \quad \text{return } \text{OST-SELECT}(v.\text{left}, i) \\
\quad \text{else} \\
\quad \quad \text{return } \text{OST-SELECT}(v.\text{right}, i - r) \\
\text{end if}
\]

This takes time proportional to the height of the tree, which is \( \Theta(\lg n) \) for a balanced binary search tree.
7 Computing rank in order statistic trees
How can we use this data structure to compute the rank of a given node in the tree?

(CLRS 342)