Ordered Collection ADT and Search Trees

Conceptual view \( \langle a, a, f, g, r, t, v, w, y \rangle \)

Methods

- Iteration is sorted

Most data structures for ordered collection

- ADT

Support these in logarithmic time

add, locate/search, remove

\begin{align*}
\text{min} & \quad \text{in ex } a \\
\text{max} & \quad \text{in ex } y \\
\text{Successor } & \quad \text{in ex } \text{successor}(k) \text{ is } v \\
\text{predecessor } & \quad \text{in ex } \text{predecessor}(k) \text{ is } g
\end{align*}
Abstract Search Tree

Each node can hold any # of elements
Let $S$ be the # elements (size) of a node
Have $s+1$ children (possibly empty)
In order-traversal - Sorted order linear time

\[
\text{Visit}(X) \begin{align*}
\text{visit}(x, \text{child}(0)) & \\
\text{output element 0} & \\
\text{visit}(x, \text{child}(1)) & \\
\text{output element 1} & \\
\text{visit}(x, \text{child}(s)) & \\
\end{align*}
\]

\textbf{loop}
Representation Property

**INORDER**

Binary Search Tree:

- $T(x, \text{Left}) \leq x.data \leq T(x, \text{right})$
- all items in left subtree

\[
T(x, \text{child}(0)) \leq x.data(0) \leq T(x, \text{child}(1)) \leq x.data(1) \\
\leq \ldots \leq x.data(s-1) \leq T(x, \text{child}(s))
\]
Review binary search trees

search j
search g

insertion is a search that ends at "null" and place it.
Finding predecessor

\[ g \leq v \leq x \leq s \]

Find pred. of \( x \)

Case 1: \( T \) exists
- \( v \) is pred

Case 2: \( T \) empty
- if \( g \) exists then \( g \) is pred.

only \( v \) could be pred (rest are smaller)
Case 3.

T empty \( x \) doesn't exist

No predecessor

\( x \) is first

in an in order traversal

Successor is symmetric
Delete

ey easy case - delete a leaf
remove it

medium case - delete a node with one child

null could be replaced by right child
Hardest case: delete a node with 2 children. Could use `succ` to find pred. Pred has 0 or 1 children. Remove pred from left subtree.