Red-Black Trees & Augmenting Data Structures

Chapters 13 and 14

Elementary Data Structures

- Why are red-black trees considered “better” than binary search trees?
- What are the properties of a red-black tree?
- What operations are performed on red-black trees?

Rotations

- Why are rotations necessary in red-black trees?
- How are rotations performed?
- What is the running time of rotation?

Insertion

- How is a node inserted into a red-black tree?
- What is the running time?

Deletion

- How is deletion performed in a red-black tree?
- What is the running time?

Augmenting Data Structures

- Sometimes a “textbook” data structure is sufficient to solve a problem exactly as it is
- However, there will be times when augmenting an existing data structure by adding more data will be required
- Rarely will you invent a brand new data structure

Dynamic Order Statistic

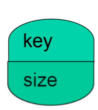
- OS-SELECT(i, S):
- OS-RANK(x, S):
- Example
  - S: {6, 3, 74,23, 84, 8, 19, 21}
  - What's the result of OS-SELECT(4, S)
  - What's the result of OS-RANK(23, S)

Order Statistics

- We have previously seen that any order statistic can be determined in O(n) from an unordered set
- How?
- Today we'll speed this up to O(lg n) time

Idea

- Augment a red-black tree
- The red-black tree will represent the set
- The size of every subtree will be stored in the node
- Notation for nodes



Order Statistic Tree

- Example

- size[x] = size[left[x]] + size[right[x]] + 1

OS-SELECT(x, i)

Example

- What's the result of OS-SELECT(root[T], 17)

Running Time

- What's the running time of OS-SELECT?

OS-Rank(T, x)

Example

- What is the result of OS-RANK(T, 38)
- What is the running time of OS-RANK?

Maintaining Subtree Sizes

- Can the sizes be efficiently maintained?

Your Turn

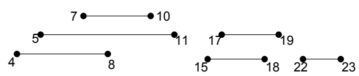
- OS-SELECT(root[T], 5) on the following tree
  - Note that you will need to calculate the sizes
- INSERT("K") into the tree

Methodology for Augmentation

1. Choose an underlying data structure
2. Determine additional information to be stored in the data structure
3. Verify that this information can be maintained for modifying operations
4. Develop new dynamic set operations that use the information

Interval Trees

- Goal: Maintain a dynamic set of intervals (closed), such as time intervals



- Query: for a given interval i, find an interval in the set that overlaps i

Following the Methodology

1. Choose an underlying data structure
   - Red-black tree keyed on the low endpoint
2. Determine additional information to be stored in the data structure
   - Store in each node x the largest value m[x] in the subtree rooted at x, as well as the interval int[x] corresponding to the key

Example

Modifying Operations

3. Verify that this information can be maintained for modifying operations
   - Insert: fix m's on the way down
   - Rotation and fixup: O(1)

New Operations

- Develop new dynamic set operations that use the information
- INTERVAL-SEARCH(i)

Example

- INTERVAL-SEARCH([14, 16])

Another Example

- INTERVAL-SEARCH([12, 14])