
Augmenting Data Structures

Chapter 14

3/7/11 CS380 Algorithm Design and Analysis 1

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

3/7/11 CS380 Algorithm Design and Analysis 2

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)

3/7/11 CS380 Algorithm Design and Analysis 3

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

3/7/11

CS380 Algorithm Design and Analysis

4

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



3/7/11

CS380 Algorithm Design and Analysis

5

Order Statistic Tree

- Example

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

3/7/11

CS380 Algorithm Design and Analysis

6

OS-SELECT(*x*, *i*)

```
OS-SELECT(x, i)  
r = x.left.size + 1  
if i == r  
    return x  
elseif i < r  
    return OS-SELECT(x.left, i)  
else return OS-SELECT(x.right, i - r)
```

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)

```
OS-RANK( $T, x$ )  
 $r = x.left.size + 1$   
 $y = x$   
while  $y \neq T.root$   
    if  $y == y.p.right$   
         $r = r + y.p.left.size + 1$   
     $y = y.p$   
return  $r$ 
```

3/7/11

CS380 Algorithm Design and Analysis

10

Example

- What is the result of OS-RANK($T, 38$)

- What is the running time of OS-RANK?

3/7/11

CS380 Algorithm Design and Analysis

11

Maintaining Subtree Sizes

- Can the sizes be efficiently maintained?

3/7/11

CS380 Algorithm Design and Analysis

12

Your Turn

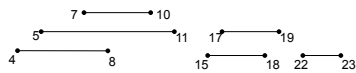
- OS-SELECT(root[T], 5) on the following tree
 - Note that you will need to calculate the sizes
- INSERT("K") into the tree

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
 - o Red-black tree keyed on the low endpoint
2. Determine additional information to be stored in the data structure
 - o 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

3/7/11

CS380 Algorithm Design and Analysis

16

Example

3/7/11

CS380 Algorithm Design and Analysis

17

Modifying Operations

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

3/7/11

CS380 Algorithm Design and Analysis

18

New Operations

- Develop new dynamic set operations that use the information

```
INTERVAL-SEARCH(T, i)  
x = T.root  
while x ≠ T.nil and i does not overlap x.int  
  if x.left ≠ T.nil and x.left.max ≥ i.low  
    x = x.left  
  else x = x.right  
return x
```

Example

- INTERVAL-SEARCH(*T*, [14, 16])

Another Example

- INTERVAL-SEARCH(*T*, [12, 14])
