Operations on **Dynamic** RB Trees

- **K-Selection**

  - **Select** an item with a specified rank

"Efficient" solution not possible without preprocessing

Preprocessing – store additional information at nodes

- **Inverse of K-Selection**

  - Find **rank** of an item in the tree

- **What information should be stored?**

  - Rank

  - ??

OS-Rank

OS-RANK(x,y)

// Different from text (recursive version)

// Find the rank of x in the subtree rooted at y

1   r = size[left[y]] + 1

2   if x = y then return r

3 else if (key[x] < key[y]) then

4   return OS-RANK(x,left[y])

5 else return r + OS-RANK(x,right[y])

Time Complexity O(log n)

OS-Select

OS-SELECT(x,i) //page 304

// Select the node with rank i

// in the subtree rooted at x

1. r = size[left[x]]+1

2. if i = r then

3.   return x

4. elseif i < r then   Time Complexity O(log n)

5.   return OS-SELECT (left[x], i)

6. else   return OS-SELECT (right[x], i-r)

RB-Tree Augmentation

- Augment x with **Size(x)**, where
  - Size(x) = size of subtree rooted at x
  - Size(NIL) = 0

Augmented Data Structures

- Why is it needed?
  - Because basic data structures not enough for all operations
  - storing extra information helps execute special operations more efficiently.
- Can any data structure be augmented?
  - Yes. Any data structure can be augmented.
- Can a data structure be augmented with any additional information?
  - Theoretically, yes.
- How to choose which additional information to store.
  - Only if we can maintain the additional information efficiently under all operations. That means, with additional information, we need to perform old and new operations efficiently maintain the additional information efficiently.

How to augment data structures

1. choose an underlying data structure

2. determine additional information to be maintained in the underlying data structure,

3. develop new operations,

4. verify that the additional information can be maintained for the modifying operations on the underlying data structure.

Augmenting RB-Trees

Theorem 14.1, page 309

Let f be a field that augments a red-black tree T with n nodes, and f(x) can be computed using only the information in nodes x, left[x], and right[x], including f[left[x]] and f[right[x]].

Then, we can <u>maintain</u> f(x) during insertion and deletion without asymptotically affecting the O(log n) performance of these operations.

For example,

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

rank[x] = ?

Examples of augmenting information for RB-Trees

- Parent

- Height

- Any associative function on all previous values or all succeeding values.

- Next

- Previous