Select with duplicates

Let S be a set of (not necessarily distinct) integers.

Let $x_1, x_2, \ldots, x_n$ be the elements of S ordered so that $x_i < x_j$.

Then x_k is the k^{th} smallest element of S.

(If confusion descends … recall the basic issue.)

Select(S,k) When S has duplicates

Lower bound:

Finding the min requires at least n-1 comparisons (no assumptions of distinctness)

Any algorithm that finds the k^{th} smallest element is $\Omega(n)$

Select(S,k) When S has duplicates

Choose x from S

Let $S_1 = \{y \in S \mid y < x\}$

Let $S_2 = \{y \in S \mid y > x\}$

Let $S_3 = \{y \in S \mid y = x\}$

If $||S_1|| \geq k$ then return Select(S_1,k)

Else if $||S_1|| + ||S_3|| \geq k$ then return x

Else return Select(S_2, $k - ||S_1|| - ||S_3||$)

Selection Rule

Choose $x \in S$

Random: Choose $x \in_R S$

Deterministic: Divide S into groups of 5

Find the median of each group

Let x be the median of medians

Is there any confusion on how these algorithms work when duplicates are allowed?

Analysis Regardless of selection rule

- Recurrence: $T(n) \leq T(\max(||S_1||,||S_3||) + cn$
- In worst case $||S_2||=1$.

Rank

- Let S be a set of integers
- For $x \in S$, $\text{rank}(x) = ||\{ y \in S \mid y \leq x\}||$

x is the k^{th} smallest element of S

⇕

$x = \min\{y \in S \mid \text{rank}(y) \geq k\}$

Rank

- Let S be a set of distinct integers
- For $x \in S$, rank(x) = $\|\{ y \in S \mid y \leq x \}\|$

x is the k^{th} smallest element of S

$\Updownarrow$

rank(x) = k

Ranking

- Let S be a set of distinct integers
- Questions:
  - Which element in S has rank k?
  - For x in S, what is rank(x)?
  - Is x in S and, if so, what is its rank?

Search Problems

- Input: Set of integers S and integer x
- Question: Is x an element of S?

MORE GENERALLY

- Input: Set of "keyed" records S and integer x
- Question: Is there a record in S with key=x?

DYNAMIC Search Problems

- Input: Set of "keyed" records S
- Operations:
  - Add record to S
  - Delete record from S
  - Find record in S with key=x (if one exists)

Dictionary Data Structure

Data structure that supports add, delete, find for set of keyed records.

- Binary search tree
  - Balanced binary search tree
- General search tree
- Hash Table

Binary Search Tree for S

- T is a binary tree
- 1-1 relationship between the nodes in T and the records in S
- BST Property: For any node X in T
  - For any node Y in the left subtree of X, Y.key $\leq$ X.key
  - For any node Y in the right subtree of X, Y.key $\geq$ X.key

Keeping a good balance …

- Search trees: O(n) time per operation
- "Balanced trees" insure O(log n) time per operation.
- Different approaches to balance:
  - Red/black trees
  - 2-3 trees
  - AVL trees

Rank

- How can we add rank-finding to the dictionary data structure?