COMP 446 / 546 ALGORITHM DESIGN AND ANALYSIS

LECTURE 7 RANDOMIZED SELECTION

ALPTEKİN KÜPÇÜ

Based on slides of David Luebke, Michael Goodrich, and Roberto Tamassia

THE SELECTION PROBLEM

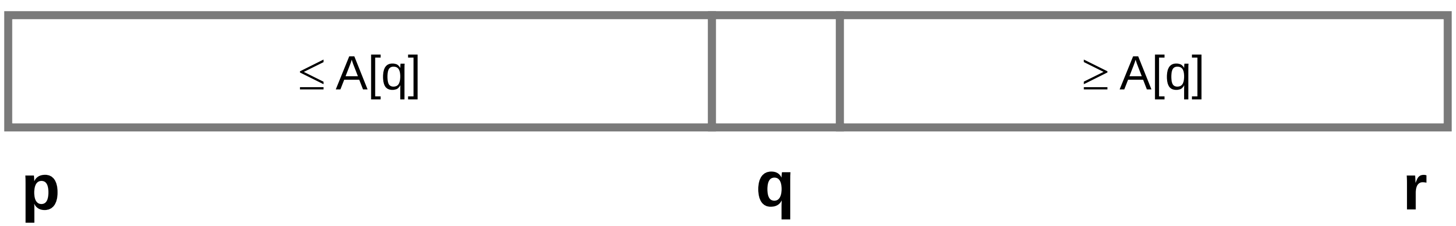
- **Given an integer k and n elements $x_1, x_2, \ldots, x_n$, taken from a total order, find the k^{th} smallest element in this set.**
  - **x** values are not necessarily sorted
- **Solution 1: Sort x values in O(n log n) time return the k^{th} element.**

k=3 7 4 9 <u>6</u> 2 → 2 4 <u>6</u> 7 9

- ***Can we solve the selection problem faster?***

FASTER SELECTION

- **Solution 2: A practical randomized algorithm with O(n) expected running time**
  - Divide and Conquer style
  - Use **R-PARTITION()** from Randomized Quicksort to divide
  - Key idea: only need to conquer one sub-array
  - This savings shows up in running time: O(n) instead of O(n log n)



if k < q, conquer left

otherwise, conquer right

RANDOMIZED SELECTION

```
RANDOMIZEDSELECT (A, p, r, i)
      if (p == r) then
            return A[p]
      q = R-PARTITION(A, p, r)
      t = q - p + 1;
      if (i == t) then
            return A[q] // pivot is what we want
      if (i < t) then
            return RANDOMIZEDSELECT(A, p, q-1, i)
      else
            return RANDOMIZEDSELECT(A, q+1, r, i-t)
```

Initial call: `RANDOMIZEDSELECT(A, 1, n, k)`

RANDOMIZED-SELECT VISUALIZATION

k=5, S=(7 4 9 3 2 6 5 1 8)
S'=(1 2 3 9 4 6 5 7 8)

k=2, S=(7 4 9 6 5 8)
S'=(7 4 6 5 8 9)

k=2, S=(7 4 6 5)
S'=(4 7 6 5)

k=1, S=(7 6 5)
S'=(5 6 7)

5

RANDOMIZED-SELECT ANALYSIS

- **Same R-PARTITION() as Randomized Quicksort**
  - Thus, in expectation, each sub-array is roughly $n/2$ size
  - $T(n) = T(n/2) + cn \le 2cn = O(n)$ expected
    - Expected one sub-problem of half the size and cn time for partitioning
- **Alternative Analysis:**
  - Use the Paranoid Quicksort partitioning idea
  - Expected number of partitionings before good partitioning is 2
  - Expected largest sub-problem size is ¾ n
  - $T(n) \le T(3n/4) + \#iterations*cn$
  - $T(n) \le T(3n/4) + 2*cn$
  - $T(n) = O(n)$ expected
- **Worst-case: $T(n) = T(n-1) + cn = O(n^2)$**

DETERMINISTIC SELECTION

- **Solution 3: O(n) worst-case (of theoretical interest, not practical)**
- **Idea: Recursively use the selection algorithm to find a good pivot for R-PARTITION()**
  - Divide S into n/5 sets of 5 each
  - Find a median in each set
  - Recursively find the median of the "baby" medians.