Announcements

- Weekly Reading: Chap 8 & 9 (CLRS)
- Homework#3 due today
- In-Class Midterm Review on Thursday, October 27, 2005
- In-Class Midterm on Thursday, November 3, 2005

Order Statistic

- *i*th order statistic of a set of n elements is the ith smallest element
- *Minimum*: the first order statistic
- *Maximum*: the nth order statistic
- *Selection problem* can be specified as:
  - Input: A set A of n distinct numbers and a number i, with $1 \leq i \leq n$
  - Ouput: the element $x \in A$ that is larger than exactly i-1 other elements of A

Minimum (A)

$min \leftarrow A[1]$

for $i \leftarrow 2$ to $length[A]$

    do if $min > A[i]$

        then $min \leftarrow A[i]$

return min

Algorithm Analysis

- $T(n) = \Theta(n)$ for Minimum(A) or Maximum(A)
- Line 4 is executed $\Theta(lg\ n)$

  For any $1 \le i \le n$, the probability of line 4 is executed is the probability that A[i] is the minimum among all A[j] for $1 \le j \le i$, which is $1/i$. So, the expectation of s

$$
\begin{aligned}
E[s] &= E[s_1 + s_2 + \ldots + s_n] \\
&= 1/1 + \ldots. + 1/n \\
&= \ln n + O(1) = \Theta(lg\ n)
\end{aligned}
$$

- Only $3\lceil n/2 \rceil$ comparisons are necessary to find both the minimum and the maximum.

Randomized-Select

1. Partition the input array around a randomly chosen element x using *Randomized-Partition*. Let k be the number of elements on the low side and n-k on the high side.

1. Use *Randomized-Select* recursively to find the ith smallest element on the low side if $i \leq k$, or the (i-k)th smallest element on the high side if $i > k$

Randomized-Select (A, p, r, i)

```
if p = r
    then return A[p]
q ← Randomized-Partition(A, p, r)
k ← q - p + 1
if i ≤ k
    then Randomized-Select(A, p, q, i)
    else Randomized-Select(A, q+1, r, i-k)
```

The worst-case running time can be $\Theta(n^2)$, but the average performance is $O(n)$.

Randomized-Partition Example

8	1	5	3	4

- Goal: Find 3rd smallest element

Randomized-Partition Example

8	1	5	<u>3</u>	4

Randomized-Partition Example

8	1	5	3	4

1	3	8	5	4

Randomized-Partition Example

8	1	5	3	4

1	3	8	5	4

8	5	4

Randomized-Partition Example

8	1	5	3	4

1	3	8	5	4

8	5	4

Randomized-Partition Example

8	1	5	3	4

1	3	8	5	4

4	5	8

Randomized-Partition Example

8	1	5	3	4

1	3	8	5	4

<u>4</u>	5	8

4

Average-Case Analysis

- $T(n) \le 1/n(\ T(\max(1, n\text{-}1)) + \sum_{k=1 \text{ to } n-1} T(\max(k, n\text{-}k))\ ) + O(n)$
  $\le 1/n(\ T(n\text{-}1) + 2 \sum_{k=\lceil n/2 \rceil \text{ to } n-1} T(k)\ ) + O(n)$
  $= 2/n \sum_{k=\lceil n/2 \rceil \text{ to } n-1} T(k) + O(n)$
- **Substitution Method: Guess** $T(n) \le c\, n$

$T(n) \le 2/n \sum_{k=\lceil n/2 \rceil \text{ to } n-1} ck + O(n)$

$\le 2c/n\ (\sum_{k=1 \text{ to } n-1} k - \sum_{k=1 \text{ to } \lceil n/2 \rceil -1} k\ ) + O(n)$

$= 2c/n\ (\ (n\text{-}1)n/2 - 1/2(\lceil n/2 \rceil -1\ )\lceil n/2 \rceil\ ) + O(n)$

$\le c(n - 1) - \ (c/n)(n/2 -1)(n/2) + O(n)$

$\le c(3n/4 - 1/2) + O(n)$

$\le cn$ $\Leftarrow$ if we pick c large enough so that $c(n/4 + 1/2)$ dominates $O(n)$

Selection in Worst-Case Linear Time

- It finds the desired element(s) by recursively partitioning the input array

- Basic idea: to generate a good split when array is partitioned using a modified deterministic partition

Selection

1. Divide the n elements of input array into $\lfloor n/5 \rfloor$ groups of 5 elements each and at most one group made up of the remaining (n mod 5) elements.
2. Find the median of each group by insertion sort & take its middle element (smaller of 2 if even number input).
3. Use *Select* recursively to find the median x of the $\lceil n/5 \rceil$ medians found in step 2.
4. Partition the input array around the median-of-medians x using a modified *Partition*. Let k be the number of elements on the low side and n-k on the high side.
5. Use *Select* recursively to find the ith smallest element on the low side if $i \leq k$, or the (i-k)th smallest element on the high side if $i > k$

Pictorial Analysis of Select

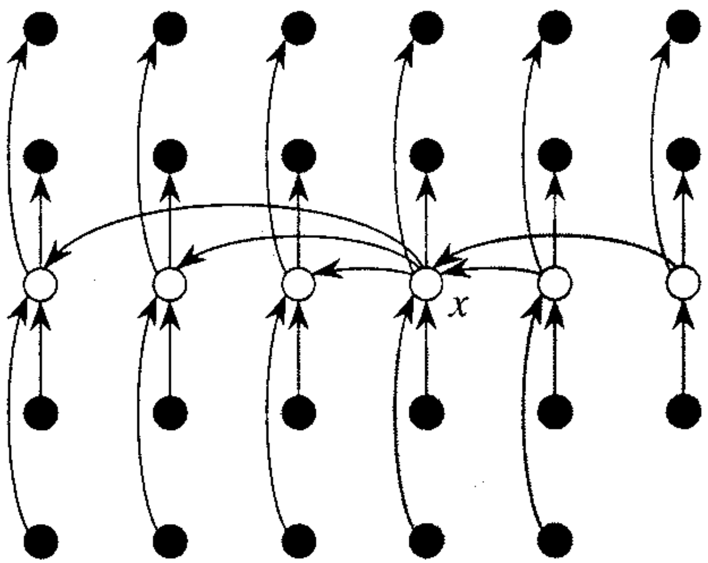

Figure 10.1 Analysis of the algorithm SELECT. The n elements are represented by small circles, and each group occupies a column. The medians of the groups are whitened, and the median-of-medians x is labeled. Arrows are drawn from larger elements to smaller, from which it can be seen that 3 out of every group of 5 elements to the right of x are greater than x, and 3 out of every group of 5 elements to the left of x are less than x. The elements greater than x are shown on a shaded background.

Algorithm Analysis (I)

- **At least half of the medians found in step 2 are greater or equal to the median-of-medians x. Thus, at least half of the $\lceil n/5 \rceil$ groups contribute 3 elements that are greater than x, except the one that has < 5 and the one group containing x. The number of elements > x is at least**

$$3\left(\left\lceil (1/2)\lceil n/5 \rceil \right\rceil - 2\right) \geq 3n/10 - 6$$

Similarly the number of elements < x is at least $3n/10 - 6$. In the worst case, SELECT is called recursively on at most $7n/10 + 6$.

Solving Recurrence

- **Step 1, 2 and 4 take $O(n)$ time. Step 3 takes time $T(\lceil n/5 \rceil)$ and step 5 takes time at most $T(7n/10 + 6)$.**

$$T(n) \leq \Theta(1), \text{ if } n \leq 80$$

$$T(n) \leq T(\lceil n/5 \rceil) + T(7n/10 + 6) + O(n), \text{ if } n > 80$$

- **Substitution Method: Guess $T(n) \leq cn$**

$$\begin{aligned} T(n) &\leq c\lceil n/5 \rceil + c\,(7n/10 + 6) + O(n) \\ &\leq cn/5 + c + 7cn/10 + 6c + O(n) \\ &\leq 9\,c\,n/10 + 7\,c + O(n) = c\,n - (c(n/10-7) - O(n)) \\ &\leq c\,n \Leftarrow \text{if we choose } c \text{ large enough such} \\ &\qquad \text{that } c(n/10 - 7) \text{ is larger than } O(n),\ n>80 \end{aligned}$$

Algorithm Analysis (II)

- **Assumption: ignoring the partial group**
- **At least half of the 5-element medians found in step 2 are less or equal to the median-of-medians x. Thus, at least half of the $\lfloor n/5 \rfloor$ groups contribute 3 elements that are greater than x. The number of elements $\leq x$ is at least $3\lfloor n/10 \rfloor$**
- **For $n \geq 50$, $3\lfloor n/10 \rfloor \geq n/4$ => the running time on $n < 50$ is $O(1)$**
- **Similarly at least $n/4$ elements $\geq x$**

Solving Recurrence

- Step 1, 2 and 4 take $O(n)$ time. Step 3 takes time $T(\lfloor n/5 \rfloor)$ and step 5 takes time at most $T(3n/4)$.

$$T(n) \le \Theta(1), \text{ if } n \le 50$$

$$T(n) \le T(\lfloor n/5 \rfloor) + T(3n/4) + O(n), \text{ if } n > 50$$

- Substitution Method: Guess $T(n) \le cn$

$$\begin{aligned} T(n) &\le c\, n/5 + 3cn/4 + O(n) \\ &\le 19\, c\, n/20 + O(n) \\ &= c\, n - (c\, n/20 - O(n)) \\ &\le c\, n \Leftarrow \text{if we choose } c \text{ large enough such} \\ &\qquad \text{that } c\, n/20 \text{ is larger than } O(n),\ n > 50 \end{aligned}$$