Today

- Order statistics (e.g., finding median)
- Two $O(n)$ time algorithms:
  - Randomized: similar to Quicksort
  - Deterministic: quite tricky
- Both are examples of divide and conquer
Order statistics

Select the $i$th smallest of $n$ elements (the element with rank $i$).

- $i = 1$: minimum;
- $i = n$: maximum;
- $i = \lceil (n+1)/2 \rceil$ or $\lfloor (n+1)/2 \rfloor$: median.

How fast can we solve the problem?

- Min/max: $O(n)$
- General $i$: $O(n \log n)$ by sorting
- We will see how to do it in $O(n)$ time
Randomized Algorithm for Finding the $i^{th}$ element

- Divide and Conquer Approach
- Main idea: PARTITION

If $i < k$, recurse on the left
If $i > k$, recurse on the right
Otherwise, output $x$
Randomized Divide-and-Conquer

**RAND-SELECT**($A, p, r, i$)

if $p = r$ then return $A[p]$

$q \leftarrow \text{RAND-PARTITION}(A, p, r)$

$k \leftarrow q - p + 1$

if $i = k$ then return $A[q]$

if $i < k$

then return $\text{RAND-SELECT}(A, p, q - 1, i)$

else return $\text{RAND-SELECT}(A, q + 1, r, i - k)$
Example

Select the $i = 7$th smallest:

```
6 10 13 5 8 3 2 11
```

**pivot**

Partition:

```
2 5 3 6 8 13 10 11
```

$k = 4$

Select the $7 - 4 = 3$rd smallest recursively.
Analysis

• What is the worst-case running time?

Unlucky:
\[ T(n) = T(n - 1) + \Theta(n) \quad \text{arithmetic series} \]
\[ = \Theta(n^2) \]

• Recall that a lucky partition splits into arrays with size ratio at most 9:1

• What if all partitions are lucky?

Lucky:
\[ T(n) = T(9n/10) + \Theta(n) \quad n^{\log_{10}9} = n^0 = 1 \]
\[ = \Theta(n) \quad \text{CASE 3} \]
Expected Running Time

• The probability that a random pivot induces lucky partition is at least $\frac{8}{10}$ (Lecture 4)
• Let $t_i$ be the number of partitions performed between the $(i-1)$-th and the $i$-th lucky partition
• The total time is at most…
  $$T = t_1 n + t_2 \left(\frac{9}{10}\right) n + t_3 \left(\frac{9}{10}\right)^2 n + \ldots$$
• The total *expected* time is at most:
  $$E[T] = E[t_1] n + E[t_2] \left(\frac{9}{10}\right) n + E[t_3] \left(\frac{9}{10}\right)^2 n + \ldots$$
  $$= \frac{10}{8} \times \left[ n + \left(\frac{9}{10}\right)n + \ldots \right]$$
  $$= O(n)$$
Digression: 9 to 1

• Do we need to define the lucky partition as 9:1 balanced?

• No. Suffices to say that both sides have size \( \geq \alpha n \), for \( 0 < \alpha < \frac{1}{2} \)

• Probability of getting a lucky partition is \( 1 - 2\alpha \)
How Does it Work In Practice?

- Need 7 volunteers (a.k.a. elements)
- Will choose the median according to height
Partitioning subroutine

\textbf{PARTITION}(A, p, r) \triangleq A[p \ldots r]
\begin{align*}
x &\leftarrow A[p] \quad \triangleq \text{pivot} = A[p] \\
i &\leftarrow p \\
\text{for } j \leftarrow p + 1 \text{ to } r \\
&\quad \text{do if } A[j] \leq x \\
&\quad \quad \text{then } i \leftarrow i + 1 \\
&\quad \quad \text{exchange } A[i] \leftrightarrow A[j] \\
&\quad \text{exchange } A[p] \leftrightarrow A[i] \\
\text{return } i
\end{align*}

\textbf{Invariant:}
\begin{array}{cccc}
x & \leq x & \geq x & ? \\
p & i & j & r
\end{array}
Summary of randomized order-statistic selection

• Works fast: linear expected time.
• Excellent algorithm in practice.
• But, the worst case is very bad: $\Theta(n^2)$.

Q. Is there an algorithm that runs in linear time in the worst case?
A. Yes, due to [Blum-Floyd-Pratt-Rivest-Tarjan’73].

Idea: Generate a good pivot recursively.
Worst-case linear-time order statistics

**SELECT**(i, n)

1. Divide the n elements into groups of 5. Find the median of each 5-element group by hand.
2. Recursively **SELECT** the median x of the \([n/5]\) group medians to be the pivot.
3. Partition around the pivot x. Let \(k = \text{rank}(x)\).
4. **if** \(i = k\) **then** return \(x\)
   **elseif** \(i < k\)
   **then** recursively **SELECT** the \(i\)th smallest element in the lower part
   **else** recursively **SELECT** the \((i-k)\)th smallest element in the upper part

Same as **RAND-SELECT**
Choosing the pivot
Choosing the pivot

1. Divide the $n$ elements into groups of 5.
Choosing the pivot

1. Divide the $n$ elements into groups of 5. Find the median of each 5-element group by rote.

© Piotr Indyk and Charles Leiserson

Introduction to Algorithms

September 22, 2004   L5.16
Choosing the pivot

1. Divide the $n$ elements into groups of 5. Find the median of each 5-element group by rote.
2. Recursively SELECT the median $x$ of the $\left\lfloor n/5 \right\rfloor$ group medians to be the pivot.
Analysis

At least half the group medians are $\leq x$, which is at least $\left\lfloor \frac{n/5}{2} \right\rfloor = \left\lfloor \frac{n}{10} \right\rfloor$ group medians.
At least half the group medians are $\leq x$, which is at least $\lceil n/5 \rceil / 2 = \lceil n/10 \rceil$ group medians.

- Therefore, at least $3 \lceil n/10 \rceil$ elements are $\leq x$. 
At least half the group medians are $\leq x$, which is at least $\lfloor \lfloor n/5 \rfloor / 2 \rfloor = \lfloor n/10 \rfloor$ group medians.

- Therefore, at least $3 \lfloor n/10 \rfloor$ elements are $\leq x$.
- Similarly, at least $3 \lfloor n/10 \rfloor$ elements are $\geq x$. 
Developing the recurrence

\[ T(n) \]

\[ \Theta(n) \]

\[ T(n/5) \]

\[ \Theta(n) \]

\[ T(7n/10) \]

**SELECT** \((i, n)\)

1. Divide the \(n\) elements into groups of 5. Find the median of each 5-element group by rote.
2. Recursively **SELECT** the median \(x\) of the \(\lceil n/5 \rceil\) group medians to be the pivot.
3. Partition around the pivot \(x\). Let \(k = \text{rank}(x)\).
4. \(\text{if } i = k \text{ then return } x\)
   \(\text{elseif } i < k\)
   \(\text{then recursively **SELECT** the } i\text{th smallest element in the lower part}\)
   \(\text{else recursively **SELECT** the } (i-k)\text{th smallest element in the upper part}\)
Solving the recurrence

\[ T(n) = T\left(\frac{1}{5} n\right) + T\left(\frac{7}{10} n\right) + \Theta(n) \]

Substitution:

\[ T(n) \leq \frac{1}{5} cn + \frac{7}{10} cn + \Theta(n) \]

\[ = \frac{18}{20} cn + \Theta(n) \]

\[ = cn - \left(\frac{2}{20} cn - \Theta(n)\right) \]

\[ \leq cn \]

if \( c \) is chosen large enough to handle the \( \Theta(n) \).
Minor simplification

• For $n \geq 50$, we have $3\lceil n/10 \rceil \geq n/4$.

• Therefore, for $n \geq 50$ the recursive call to SELECT in Step 4 is executed recursively on $\leq 3n/4$ elements.

• Thus, the recurrence for running time can assume that Step 4 takes time $T(3n/4)$ in the worst case.

• For $n < 50$, we know that the worst-case time is $T(n) = \Theta(1)$. 
Conclusions

• Since the work at each level of recursion is a constant fraction $\frac{18}{20}$ smaller, the work per level is a geometric series dominated by the linear work at the root.

• In practice, this algorithm runs slowly, because the constant in front of $n$ is large.

• The randomized algorithm is far more practical.

**Exercise:** Why not divide into groups of 3?