Proof (completed)

Q. How many $h$'s cause $x$ and $y$ to collide?

A. There are $m$ choices for each of $a_1, a_2, \ldots, a_r$, but once these are chosen, exactly one choice for $a_0$ causes $x$ and $y$ to collide, namely

$$\left\{ \begin{array}{c} \sum_i \cdot (x_i - y_i) \\ a_0 = -a_i (x_0 - y_0) \mod m. \end{array} \right.$$ 

Thus, the number of $h$'s that cause $x$ and $y$ to collide is $m^r \cdot 1 = m^r = |H|/m$. 

October 5, 2005

Copyright © 2001-5 by Erik D. Demaine and Charles E. Leiserson

Order Statistics
- Randomized divide and conquer
- Analysis of expected time
- Worst-case linear-time order statistics
- Analysis

Prof. Erik Demaine
Order statistics

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

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

**Naive algorithm**: Sort and index $i$th element.
Worst-case running time $= \Theta(n \lg n) + \Theta(1)$
$= \Theta(n \lg n),$
using merge sort or heapsort (*not* quicksort).
Randomized divide-and-conquer algorithm

\textbf{Rand-Select}(A, p, q, i) \quad \triangleright \text{ } i\text{th smallest of } A[p..q]

\textbf{if} \ p = q \ \textbf{then return} \ A[p]

\textbf{r} \leftarrow \textbf{Rand-Partition}(A, p, q)

\textbf{k} \leftarrow r - p + 1 \quad \triangleright \ k = \text{rank}(A[r])

\textbf{if} \ i = k \ \textbf{then return} \ A[r]

\textbf{if} \ i < k

\quad \textbf{then return} \textbf{Rand-Select}(A, p, r - 1, i)

\textbf{else return} \textbf{Rand-Select}(A, r + 1, q, i - k)

\begin{center}
\begin{tikzpicture}
\draw[-latex] (0,0) -- (3,0) node[midway,right] {$k$};
\draw[dashed] (0,0.5) -- (3,0.5) node[midway,right] {$A[r]$};
\draw[dashed] (0,-0.5) -- (3,-0.5) node[midway,right] {$A[r]$};
\draw[fill=blue!30] (0,0) rectangle (1,0.5);
\draw[fill=red!30] (2,-0.5) rectangle (3,0.5);
\node at (1.5,0) {$\leq A[r]$};
\node at (2.5,0.5) {$A[r]$};
\node at (2.5,-0.5) {$\geq A[r]$};
\node at (-0.5,0) {$p$};
\node at (1.5,0) {$r$};
\node at (3.5,0) {$q$};
\end{tikzpicture}
\end{center}
Example

Select the $i = 7$th smallest:

$pivot$

Partition:

Select the $7 - 4 = 3$rd smallest recursively.
Intuition for analysis

(All our analyses today assume that all elements are distinct.)

Lucky:
\[ T(n) = T(9n/10) + \Theta(n) \]
\[ = \Theta(n) \]

Unlucky:
\[ T(n) = T(n - 1) + \Theta(n) \]
\[ = \Theta(n^2) \]

Worse than sorting!
Analysis of expected time

The analysis follows that of randomized quicksort, but it’s a little different.

Let $T(n) =$ the random variable for the running time of \textsc{Rand-Select} on an input of size $n$, assuming random numbers are independent.

For $k = 0, 1, \ldots, n-1$, define the \textit{indicator random variable}

$$X_k = \begin{cases} 
1 & \text{if PARTITION generates a } k : n-k-1 \text{ split,} \\
0 & \text{otherwise.}
\end{cases}$$
Analysis (continued)

To obtain an upper bound, assume that the $i$th element always falls in the larger side of the partition:

$$T(n) = \begin{cases} 
T(\max\{0, n-1\}) + \Theta(n) & \text{if } 0:n-1 \text{ split,} \\
T(\max\{1, n-2\}) + \Theta(n) & \text{if } 1:n-2 \text{ split,} \\
\vdots \\
T(\max\{n-1, 0\}) + \Theta(n) & \text{if } n-1:0 \text{ split,}
\end{cases}$$

$$= \sum_{k=0}^{n-1} X_k \left( T(\max\{k, n-k-1\}) + \Theta(n) \right).$$
Calculating expectation

\[
E[T(n)] = E \left[ \sum_{k=0}^{n-1} X_k (T(\max\{k, n-k-1\}) + \Theta(n)) \right]
\]

Take expectations of both sides.
Calculating expectation

\[ E[T(n)] = E\left[ \sum_{k=0}^{n-1} X_k (T(\max\{k, n-k-1\}) + \Theta(n)) \right] \]

\[ = \sum_{k=0}^{n-1} E[X_k (T(\max\{k, n-k-1\}) + \Theta(n))] \]

Linearity of expectation.
Calculating expectation

\[
E[T(n)] = E \left[ \sum_{k=0}^{n-1} X_k \left( T(\max \{k, n - k - 1\}) + \Theta(n) \right) \right]
\]

\[
= \sum_{k=0}^{n-1} E[X_k \left( T(\max \{k, n - k - 1\}) + \Theta(n) \right)]
\]

\[
= \sum_{k=0}^{n-1} E[X_k] \cdot E[T(\max \{k, n - k - 1\}) + \Theta(n)]
\]

Independence of \(X_k\) from other random choices.
Calculating expectation

\[
E[T(n)] = E\left[ \sum_{k=0}^{n-1} X_k \left( T(\max\{k, n-k-1\}) + \Theta(n) \right) \right]
\]

\[
= \sum_{k=0}^{n-1} E[X_k \left( T(\max\{k, n-k-1\}) + \Theta(n) \right)]
\]

\[
= \sum_{k=0}^{n-1} E[X_k] \cdot E[T(\max\{k, n-k-1\}) + \Theta(n)]
\]

\[
= \frac{1}{n} \sum_{k=0}^{n-1} E[T(\max\{k, n-k-1\})] + \frac{1}{n} \sum_{k=0}^{n-1} \Theta(n)
\]

Linearity of expectation; \( E[X_k] = 1/n \).
Calculating expectation

\[
E[T(n)] = E \left[ \sum_{k=0}^{n-1} X_k (T(\max\{k, n-k-1\}) + \Theta(n)) \right]
\]

\[
= \sum_{k=0}^{n-1} E[X_k] \cdot E[T(\max\{k, n-k-1\}) + \Theta(n)]
\]

\[
= \frac{1}{n} \sum_{k=0}^{n-1} E[T(\max\{k, n-k-1\})] + \frac{1}{n} \sum_{k=0}^{n-1} \Theta(n)
\]

\[
\leq \frac{2}{n} \sum_{k=\lceil n/2 \rceil}^{n-1} E[T(k)] + \Theta(n)
\]

Upper terms appear twice.
Hairy recurrence

(But not quite as hairy as the quicksort one.)

\[ E[T(n)] = 2 \sum_{k=\lfloor n/2 \rfloor}^{n-1} E[T(k)] + \Theta(n) \]

Prove: \( E[T(n)] \leq cn \) for constant \( c > 0 \).

- The constant \( c \) can be chosen large enough so that \( E[T(n)] \leq cn \) for the base cases.

Use fact: \( \sum_{k=\lfloor n/2 \rfloor}^{n-1} k \leq \frac{3}{8} n^2 \) (exercise).
Substitution method

\[ E[T(n)] \leq 2 \sum_{n=\lfloor n/2 \rfloor}^{n-1} c_k + \Theta(n) \]

Substitute inductive hypothesis.
Substitution method

\[ E[T(n)] \leq \frac{2}{n} \sum_{k=\lfloor n/2 \rfloor}^{n-1} ck + \Theta(n) \]

\[ \leq \frac{2c}{n} \left( \frac{3}{8} n^2 \right) + \Theta(n) \]

Use fact.
Substitution method

\[
E[T(n)] \leq 2 \sum_{k=\lfloor n/2 \rfloor}^{n-1} ck + \Theta(n)
\]

\[
\leq \frac{2c}{n} \left( \frac{3}{8} n^2 \right) + \Theta(n)
\]

\[
= cn - \left( \frac{cn}{4} - \Theta(n) \right)
\]

Express as \textit{desired} – \textit{residual}.
Substitution method

\[ E[T(n)] \leq \frac{2}{n} \sum_{k=\lfloor n/2 \rfloor}^{n-1} ck + \Theta(n) \]

\[ \leq \frac{2c}{n} \left( \frac{3}{8} n^2 \right) + \Theta(n) \]

\[ = cn - \left( \frac{cn}{4} - \Theta(n) \right) \]

\[ \leq cn, \]

if \( c \) is chosen large enough so that \( cn/4 \) dominates the \( \Theta(n) \).
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, and Tarjan [1973].

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

\text{Select}(i, n)

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

2. Recursively \text{Select} the median \( x \) of the \( \lfloor n/5 \rfloor \) group medians to be the pivot.

3. Partition around the pivot \( x \). Let \( k = \text{rank}(x) \).

4. \textbf{if} \( i = k \) \textbf{then return} \( x \)
   \textbf{elseif} \( i < k \)
   \hspace{1em} \textbf{then} recursively \text{Select} the \( i \)th smallest element in the lower part
   \textbf{else} recursively \text{Select} the \( (i-k) \)th smallest element in the upper part

\begin{itemize}
  \item Same as \texttt{Rand-Select}
\end{itemize}
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.

lesser

greater
Choosing the pivot

1. Divide the $n$ elements into groups of 5. Find the median of each 5-element group by rote.
2. Recursively \texttt{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 $\left\lfloor \frac{n}{5} \right\rfloor / 2 = \left\lfloor \frac{n}{10} \right\rfloor$ group medians.
**Analysis** (Assume all elements are distinct.)

At least half the group medians are \( \leq x \), which is at least \( \lceil \lceil n/5 \rceil /2 \rceil = \lceil n/10 \rceil \) group medians.

- Therefore, at least \( 3 \lceil n/10 \rceil \) elements are \( \leq x \).
**Analysis** (Assume all elements are distinct.)

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

- Therefore, at least $3 \left\lfloor \frac{n}{10} \right\rfloor$ elements are $\leq x$.
- Similarly, at least $3 \left\lceil \frac{n}{10} \right\rceil$ elements are $\geq x$. 
Minor simplification

- For \( n \geq 50 \), we have \( 3 \left\lfloor n/10 \right\rfloor \geq n/4 \).
- Therefore, for \( n \geq 50 \) the recursive call to \texttt{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 \( T(n) = \Theta(1) \).
Developing the recurrence

\[
T(n) \quad \text{SELECT}(i, n)
\]

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 \( \lfloor n/5 \rfloor \) group medians to be the pivot.

3. Partition around the pivot \( x \). Let \( k = \text{rank}(x) \).

4. \textbf{if } i = k \textbf{ then return } x \\
   \textbf{elseif } i < k \\
   \textbf{then recursively SELECT the } i \text{th smallest element in the lower part} \\
   \textbf{else recursively SELECT the } (i-k)\text{th smallest element in the upper part}
Solving the recurrence

\[ T(n) = T\left(\frac{1}{5}n\right) + T\left(\frac{3}{4}n\right) + \Theta(n) \]

**Substitution:**

\[ T(n) \leq \frac{1}{5}cn + \frac{3}{4}cn + \Theta(n) \]
\[ = \frac{19}{20}cn + \Theta(n) \]
\[ = cn - \left(\frac{1}{20}cn - \Theta(n)\right) \]
\[ \leq cn \]

if \( c \) is chosen large enough to handle both the \( \Theta(n) \) and the initial conditions.
Conclusions

• Since the work at each level of recursion is a constant fraction \((19/20)\) 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?