Lecture 7
Medians and Order Statistics

View in slide-show mode
Medians and Order Statistics

\textit{ith order statistic}: \textit{i}th smallest element of a set of \textit{n} elements

\textit{minimum}: first order statistic

\textit{maximum}: \textit{n}th order statistic

\textit{median}: “halfway point” of the set

\[ i = \lfloor (n+1)/2 \rfloor \text{ or } \lceil (n+1)/2 \rceil \]
Selection Problem

- **Selection problem**: Select the $i^{th}$ smallest of $n$ elements

- **Naïve algorithm**: Sort the input array $A$; then return $A[i]$

  \[ T(n) = \Theta(n \log n) \]

  *using e.g. merge sort (but not quicksort)*

- Can we do any better?
Selection in Expected Linear Time

- Randomized algorithm using divide and conquer

- Similar to randomized quicksort
  - *Like quicksort*: Partitions input array recursively
  - *Unlike quicksort*: Makes a single recursive call

  *Reminder: Quicksort makes two recursive calls*

- Expected runtime: \( \Theta(n) \)

  *Reminder: Expected runtime of quicksort: \( \Theta(n \log n) \)*
Selection in Expected Linear Time: Example 1

Select the 2\textsuperscript{nd} smallest element:

\begin{array}{cccccccc}
6 & 10 & 13 & 5 & 8 & 3 & 2 & 11 \\
\end{array} \quad i = 2

Partition the input array:

\begin{array}{cccccccc}
2 & 3 & 5 & 13 & 8 & 10 & 6 & 11 \\
\end{array}

make a recursive call to select the 2\textsuperscript{nd} smallest element in left subarray
Selection in Expected Linear Time: Example 2

Select the 7th smallest element:

\[i = 7\]

Partition the input array:

\[
\begin{array}{cccccccc}
2 & 3 & 5 & 13 & 8 & 10 & 6 & 11 \\
\end{array}
\]

make a recursive call to select the 4th smallest element in right subarray
Selection in Expected Linear Time

\[
\text{R-SELECT}(A, p, r, i) \\
\quad \text{if } p = r \text{ then} \\
\quad \quad \text{return } A[p] \\
\quad q \leftarrow \text{R-PARTITION}(A, p, r) \\
\quad k \leftarrow q - p + 1 \\
\quad \text{if } i \leq k \text{ then} \\
\quad \quad \text{return } \text{R-SELECT}(A, p, q, i) \\
\quad \text{else} \\
\quad \quad \text{return } \text{R-SELECT}(A, q+1, r, i-k)
\]

\[
\begin{array}{ccc}
\leq x \text{ (k smallest elements)} & \geq x \\
\hline
p & q & r
\end{array}
\]

\[
x = \text{pivot}
\]
Selection in Expected Linear Time

- All elements in $L \leq$ all elements in $R$
- $L$ contains $|L| = q - p + 1 = k$ smallest elements of $A[p...r]$
  
  if $i \leq |L| = k$ then
  
  search $L$ recursively for its $i$-th smallest element

  else

  search $R$ recursively for its $(i-k)$-th smallest element
Runtime Analysis

- **Worst case:**

  Imbalanced partitioning at every level
  and the recursive call always to the larger partition

```
1 2 3 4 5 6 7 8   i=8
```

recursive call

```
2 3 4 5 6 7 8   i=7
```

recursive call
Runtime Analysis

- **Worst case:**
  \[ T(n) = T(n-1) + \Theta(n) \]
  \[ \Rightarrow T(n) = \Theta(n^2) \]
  Worse than the naïve method (based on sorting)

- **Best case:** Balanced partitioning at every recursive level
  \[ T(n) = T(n/2) + \Theta(n) \]
  \[ \Rightarrow T(n) = \Theta(n) \]

- **Avg case:** Expected runtime – need analysis
Reminder: Various Outcomes of H-PARTITION

\[ P(\text{rank}(x) = i) = \frac{1}{n} \quad \text{for } 1 \leq i \leq n \]

\[ \begin{align*} 
\text{if } \text{rank}(x) = 1 & \text{ then } |L| = 1 \\
\text{if } \text{rank}(x) > 1 & \text{ then } |L| = \text{rank}(x) - 1 
\end{align*} \]

\[ P(|L| = 1) = P(\text{rank}(x) = 1) + P(\text{rank}(x) = 2) \]

\[ P(|L| = i) = P(\text{rank}(x) = i+1) \quad \text{for } 1 < i < n \]

\[ P(|L| = 1) = 2/n \]

\[ P(|L| = i) = 1/n \quad \text{for } 1 < i < n \]
To compute the **upper bound** for the avg case, assume that the $i^{th}$ element always falls into the **larger partition**.

We will analyze the case where the recursive call is always made to the larger partition.

➔ this will give us an upper bound for the avg case.
## Various Outcomes of H-PARTITION

<table>
<thead>
<tr>
<th>rank(x)</th>
<th>prob.</th>
<th>( T(n) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1/n</td>
<td>( \leq T(\max(1, n-1)) + \Theta(n) )</td>
</tr>
<tr>
<td>2</td>
<td>1/n</td>
<td>( \leq T(\max(1, n-1)) + \Theta(n) )</td>
</tr>
<tr>
<td>3</td>
<td>1/n</td>
<td>( \leq T(\max(2, n-2)) + \Theta(n) )</td>
</tr>
<tr>
<td>i+1</td>
<td>1/n</td>
<td>( \leq T(\max(i, n-i)) + \Theta(n) )</td>
</tr>
<tr>
<td>n</td>
<td>1/n</td>
<td>( \leq T(\max(n-1, 1)) + \Theta(n) )</td>
</tr>
</tbody>
</table>

![Diagram showing various outcomes of H-PARTITION](image_url)
Average-Case Analysis of Randomized Select

Recall: \( P(|L|=i) = \begin{cases} 
\frac{2}{n} & \text{for } i = 1 \\
\frac{1}{n} & \text{for } i = 2, 3, \ldots, n-1 
\end{cases} \)

Upper bound: Assume \( i \)-th element always falls into the larger part

\[
T(n) \leq \frac{1}{n} T(\max(1, n-1)) + \frac{1}{n} \sum_{q=1}^{n-1} T(\max(q, n-q)) + O(n)
\]

Note: \( \frac{1}{n} T(\max(1, n-1)) = \frac{1}{n} T(n-1) = \frac{1}{n} \quad O(n^2) = O(n) \)

\[
\therefore \quad T(n) \leq \frac{1}{n} \sum_{q=1}^{n-1} T(\max(q, n-q)) + O(n)
\]
Average-Case Analysis of Randomized Select

\[ T(n) \leq \frac{1}{n} \sum_{q=1}^{n-1} T(\max(q, n-q)) + O(n) \]

\[ \max(q, n-q) = \begin{cases} 
q & \text{if } q \geq \lfloor n/2 \rfloor \\
q & \text{if } q < \lfloor n/2 \rfloor 
\end{cases} \]

- \( n \) is odd: \( T(k) \) appears twice for \( k = \lfloor n/2 \rfloor + 1, \lfloor n/2 \rfloor + 2, \ldots, n-1 \)
- \( n \) is even: \( T(\lceil n/2 \rceil) \) appears once \( T(k) \) appears twice for \( k = \lceil n/2 \rceil + 1, \lfloor n/2 \rfloor + 2, \ldots, n-1 \)

Hence, in both cases:

\[ \sum_{q=1}^{n-1} T(\max(q, n-q)) + O(n) \leq 2 \sum_{q=\lfloor n/2 \rfloor}^{n-1} T(q) + O(n) \]

\[ T(n) \leq \frac{2}{n} \sum_{q=\lfloor n/2 \rfloor}^{n-1} T(q) + O(n) \]
Average-Case Analysis of Randomized Select

\[ T(n) \leq \frac{2}{n} \sum_{q=\lfloor n/2 \rfloor}^{n-1} T(q) + O(n) \]

By substitution guess \( T(n) = O(n) \)

Inductive hypothesis: \( T(k) \leq ck, \ \forall \ k < n \)

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

\[ = \frac{2c}{n} \left( \sum_{k=1}^{n-1} k - \sum_{k=1}^{[n/2]-1} k \right) + O(n) \]

\[ = \frac{2c}{n} \left( \frac{1}{2} n (n-1) - \frac{1}{2} \left[ \frac{n}{2} \right] \left( \frac{n}{2} - 1 \right) \right) + O(n) \]
Average-Case Analysis of Randomized Select

\[ T(n) \leq \frac{2c}{n} \left( \frac{1}{2} n(n-1) - \frac{1}{2} \left\lfloor \frac{n}{2} \right\rfloor \left( \frac{n}{2} - 1 \right) \right) + O(n) \]

\[ \leq c(n-1) - \frac{c}{4} n + \frac{c}{2} + O(n) \]

\[ = cn - \frac{c}{4} n - \frac{c}{2} + O(n) \]

\[ = cn - \left( \frac{c}{4} n + \frac{c}{2} \right) - O(n) \]

\[ \leq cn \]

*since we can choose c large enough so that \( \frac{cn}{4} + \frac{c}{2} \) dominates \( O(n) \)*
Summary of Randomized Order-Statistic Selection

• Works fast: linear expected time
• Excellent algorithm in practise
• 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[1973]

Idea: Generate a good pivot recursively..
Selection in Worst Case Linear Time

\[
\text{SELECT}(S, n, i) \triangleright \text{return } i\text{-th element in set } S \text{ with } n \text{ elements}
\]

\[
\text{if } n \leq 5 \text{ then}
\]

\[
\text{SORT } S \text{ and return the } i\text{-th element}
\]

\[
\text{DIVIDE } S \text{ into } \left\lfloor \frac{n}{5} \right\rfloor \text{ groups}
\]

\[
\triangleright \text{first } \left\lfloor \frac{n}{5} \right\rfloor \text{ groups are of size 5, last group is of size } n \mod 5
\]

\[
\text{FIND median set } M=\{m_1, \ldots, m_{\left\lfloor n/5 \right\rfloor}\} \triangleright m_j: \text{median of } j\text{-th group}
\]

\[
x \leftarrow \text{SELECT}(M, \left\lfloor \frac{n}{5} \right\rfloor, \left\lfloor (\left\lfloor n/5 \right\rfloor+1)/2 \right\rfloor)
\]

\[
\text{PARTITION set } S \text{ around the pivot } x \text{ into } L \text{ and } R
\]

\[
\text{if } i \leq |L| \text{ then}
\]

\[
\text{return } \text{SELECT}(L, |L|, i)
\]

\[
\text{else}
\]

\[
\text{return } \text{SELECT}(R, n-|L|, i-|L|)
\]
Selection in Worst Case Linear Time - Example

**Input**: Array $S$ and index $i$

**Output**: The $i^{th}$ smallest value

\[
S = \{25, 9, 16, 8, 11, 27, 39, 42, 15, 6, 32, 14, 36, 20, 33, 22, 31, 4, 17, 3, 30, 41, 2, 13, 19, 7, 21, 10, 34, 1, 37, 23, 40, 5, 29, 18, 24, 12, 38, 28, 26, 35, 43\}
\]
### Selection in Worst Case Linear Time - Example

**Step 1**: Divide the input array into **groups of size 5**

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>25</td>
<td>27</td>
<td>32</td>
<td>22</td>
<td>30</td>
<td>7</td>
</tr>
<tr>
<td>9</td>
<td>39</td>
<td>14</td>
<td>31</td>
<td>41</td>
<td>21</td>
</tr>
<tr>
<td>16</td>
<td>42</td>
<td>36</td>
<td>4</td>
<td>2</td>
<td>10</td>
</tr>
<tr>
<td>8</td>
<td>15</td>
<td>20</td>
<td>17</td>
<td>13</td>
<td>34</td>
</tr>
<tr>
<td>11</td>
<td>6</td>
<td>33</td>
<td>3</td>
<td>19</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>26</td>
<td>18</td>
<td>35</td>
<td>24</td>
<td>40</td>
<td>12</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>28</td>
<td>38</td>
<td>19</td>
<td>13</td>
<td>34</td>
<td>5</td>
</tr>
</tbody>
</table>
Selection in Worst Case Linear Time - Example

**Step 2**: Compute the median of each group

Let $M$ be the set of the medians computed:

$$M = \{11, 27, 32, 17, 19, 10, 29, 24, 35\}$$
Selection in Worst Case Linear Time - Example

Step 3: Compute the median of the median group $M$

Let $x \leftarrow \text{SELECT} (M, |M|, \left\lceil \frac{|M|+1}{2} \right\rceil )$ where $|M| = \left\lceil n/5 \right\rceil$

```
9  8  15 14  4  2  7  5  18
6  3  20  3 13  1 23 12 26
11 27 32 17 19 10 29 24 35
16 42 33 31 30 34 40 28 43
25 39 36 22 41 21 37 38
```

$\Rightarrow$ median $= 24$

The runtime of the recursive call: $T(|M|) = T\left\lceil n/5 \right\rceil$
Selection in Worst Case Linear Time - Example

Step 4: Partition the input array $S$ around the median-of-medians $x$

$S = \{25\ 9\ 16\ 8\ 11\ 27\ 39\ 42\ 15\ 6\ 32\ 14\ 36\ 20\ 33\ 22\ 31\ 4\ 17\ 3\ 30\ 41\ 2\ 13\ 19\ 7\ 21\ 10\ 34\ 1\ 37\ 23\ 40\ 5\ 29\ 18\ 24\ 12\ 38\ 28\ 26\ 35\ 43\}$

Partition $S$ around $x = 24$

Claim: Partitioning around $x$ is guaranteed to be well-balanced.
Selection in Worst Case Linear Time - Example

*Claim*: Partitioning around $x=24$ is guaranteed to be *well-balanced*.

2 out of 5 in each group greater than the median in the group, which is greater than $x$

About half of the medians greater than $x$

About 3n/10 elts greater than $x$

About 2n/10 elts greater than the median in the group, which is greater than $x$

2 out of 5 in each group greater than $x$

About n/10
Selection in Worst Case Linear Time - Example

Claim: Partitioning around $x=24$ is guaranteed to be well-balanced.

About half of the medians less than $x$

About $2n/10$

About $3n/10$ elts less than $x$

2 out of 5 in each group less than the median in the group, which is less than $x$
Selection in Worst Case Linear Time - Example

\[
S = \{25 \ 9 \ 16 \ 8 \ 11 \ 27 \ 39 \ 42 \ 15 \ 6 \ 32 \ 14 \ 36 \ 20 \ 33 \ 22 \ 31 \ 4 \ 17 \ 3 \ 30 \ 41 \\
\quad \quad 2 \ 13 \ 19 \ 7 \ 21 \ 10 \ 34 \ 1 \ 37 \ 23 \ 40 \ 5 \ 29 \ 18 \ 24 \ 12 \ 38 \ 28 \ 26 \ 35 \ 43\}
\]

Partitioning \( S \) around \( x = 24 \) will lead to partitions of sizes \(~3n/10\) and \(~7n/10\) in the worst case.

**Step 5**: Make a recursive call to one of the partitions

\[
\text{if } i \leq |L| \text{ then} \\
\quad \quad \text{return } \text{SELECT}(L, |L|, i) \\
\text{else} \\
\quad \quad \text{return } \text{SELECT}(R, n–|L|, i–|L|)
\]
Selection in Worst Case Linear Time

\[ \text{SELECT}(S, n, i) \triangleright \text{return } i\text{-th element in set } S \text{ with } n \text{ elements} \]

\[
\text{if } n \leq 5 \text{ then}
\]

\[
\text{SORT } S \text{ and return the } i\text{-th element}
\]

DIVIDE \( S \) into \( \left\lfloor n/5 \right\rfloor \) groups

\[
\triangleright \text{first } \left\lfloor n/5 \right\rfloor \text{ groups are of size } 5, \text{ last group is of size } n \mod 5
\]

FIND median set \( M=\{m_1, \ldots, m_{\left\lfloor n/5 \right\rfloor}\} \triangleright m_j: \text{median of } j\text{-th group}

\[
x \leftarrow \text{SELECT}(M, \left\lfloor n/5 \right\rfloor, \left\lfloor (\left\lfloor n/5 \right\rfloor+1)/2 \right\rfloor)
\]

PARTITION set \( S \) around the pivot \( x \) into \( L \) and \( R \)

\[
\text{if } i \leq |L| \text{ then}
\]

\[
\text{return } \text{SELECT}(L, |L|, i)
\]

\[
\text{else}
\]

\[
\text{return } \text{SELECT}(R, n–|L|, i–|L|)
\]
Choosing the Pivot

1. Divide $S$ into groups of size 5
Choosing the Pivot

1. Divide $S$ into groups of size 5
2. Find the median of each group
Choosing the Pivot

1. Divide $S$ into groups of size 5
2. Find the median of each group
3. Recursively select the median $x$ of the medians

\[ x \geq x \]
Choosing the Pivot

At least half of the medians $\geq x$

Thus $m = \lfloor \sqrt{n/5} \rfloor / 2$ groups contribute 3 elements to $R$ except possibly the last group and the group that contains $x$

$|R| \geq 3 \left( m - 2 \right) \geq \frac{3n}{10} - 6$
Similarly
\[ |L| \geq \frac{3n}{10} - 6 \]
Therefore, SELECT is recursively called on at most
\[ n - \left( \frac{3n}{10} - 6 \right) = \frac{7n}{10} + 6 \text{ elements} \]
Selection in Worst Case Linear Time

\[ \text{SELECT}(S, n, i) \]  \( \triangleright \) return \( i \)-th element in set \( S \) with \( n \) elements

if \( n \leq 5 \) then

- \( \text{SORT} \ S \text{ and return the } i \text{-th element} \)

\( \Theta(n) \)

- \( \text{DIVIDE} \ S \text{ into } [n/5] \text{ groups} \)
  - \( \triangleright \text{ first } [n/5] \text{ groups are of size 5, last group is of size } n \mod 5 \)

\( \Theta(n) \)

- \( \text{FIND} \text{ median set } M = \{ m, \ldots, m_{[n/5]} \} \) \( \triangleright m_j: \text{ median of } j \text{-th group} \)

\( T([n/5]) \)

- \( x \leftarrow \text{SELECT}(M, [n/5], [(n/5)+1)/2]) \)

\( \Theta(n) \)

- \( \text{PARTITION} \text{ set } S \text{ around the pivot } x \text{ into } L \text{ and } R \)

\( T(\frac{7n}{10}+6) \)

- if \( i \leq |L| \) then
  - return \( \text{SELECT}(L, |L|, i) \)

  else

  return \( \text{SELECT}(R, n-|L|, i-|L|) \)
Selection in Worst Case Linear Time

Thus recurrence becomes

\[ T(n) \leq T\left(\left\lfloor \frac{n}{5} \right\rfloor\right) + T\left(\frac{7n}{10} + 6\right) + \Theta(n) \]

Guess \( T(n) = O(n) \) and prove by induction

Inductive step: \( T(n) \leq c\left\lfloor n/5 \right\rfloor + c \left(\frac{7n}{10}+6\right) + \Theta(n) \)

\[ \leq cn/5 + c + 7cn/10 + 6c + \Theta(n) \]

\[ = 9cn/10 + 7c + \Theta(n) \]

\[ = cn - [c(n/10 − 7) − \Theta(n)] \leq cn \text{ for large } c \]

Work at each level of recursion is a constant factor (9/10) smaller.