Lecture 7
Medians and Order Statistics

View in slide-show mode
Medians and Order Statistics

*i*th order statistic: *i*th smallest element of a set of *n* elements

**minimum**: first order statistic

**maximum**: *n*th order statistic

**median**: “halfway point” of the set

\[ i = \left\lfloor \frac{n+1}{2} \right\rfloor \text{ or } \left\lceil \frac{n+1}{2} \right\rceil \]
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{align*}
6 & \quad 10 & \quad 13 & \quad 5 & \quad 8 & \quad 3 & \quad 2 & \quad 11 & \quad i = 2
\end{align*}

Partition the input array:

\begin{align*}
2 & \quad 3 & \quad 5 & \quad 13 & \quad 8 & \quad 10 & \quad 6 & \quad 11
\end{align*}

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

Select the 7\textsuperscript{th} smallest element:

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

i = 7

Partition the input array:

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

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

\[ \text{R-SELECT}(A, p, r, i) \]

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

\[ x = \text{pivot} \]

\[ p \leq x \text{ (k smallest elements)} \quad \quad \geq x \]

\[ q \quad r \]
Selection in Expected Linear Time

Let \( p \leq x \leq q \) and \( x \geq r \).

- 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

\( x = pivot \)
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

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

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

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

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

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

\[ \Pr(|L| = i) = \frac{1}{n} \quad \text{for } 1 < i < n \]
Average Case Analysis of Randomized Select

- 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]
Average-Case Analysis of Randomized Select

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

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

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

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

\[ \max(q, n-q) = \begin{cases} 
q & \text{if } q \geq \left\lceil n/2 \right\rceil \\
n-q & \text{if } q < \left\lceil n/2 \right\rceil 
\end{cases} \]

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

Hence, in both cases:

\[ \sum_{q=1}^{n-1} T(\max(q, n-q)) + O(n) \leq 2 \sum_{q=\left\lceil n/2 \right\rceil}^{n-1} T(q) + O(n) \]

\[ T(n) \leq \frac{2}{n} \sum_{q=\left\lceil n/2 \right\rceil}^{n-1} T(q) + O(n) \]
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: \( T(k) \leq ck, \ \forall \ k < n \)

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

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

\[ = \frac{2c}{n} \left( \frac{1}{2} n (n-1) - \frac{1}{2} \left\lfloor \frac{n}{2} \right\rfloor \left( \frac{n}{2} - 1 \right) \right) + O(n) \]
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 \((cn/4+c/2)\) dominates \(O(n)\)
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 & Tarjan[1973]

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

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

if \( n \leq 5 \) then

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

DIVIDE \( S \) into \( \lceil n/5 \rceil \) groups

\[ \triangleright \text{first } \lceil n/5 \rceil \text{ groups are of size 5, last group is of size } n \mod 5 \]

FIND median set \( M = \{ m_1, \ldots, m_{\lceil n/5 \rceil} \} \quad \triangleright \text{median of } j\text{-th group} \)

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

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

if \( i \leq |L| \) then

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

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

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

$x \leftarrow \text{SELECT} \left( M, |M|, \left\lfloor \frac{|M|+1}{2} \right\rfloor \right)$

where $|M| = \left\lfloor n/5 \right\rfloor$

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

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

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

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*.

About half of the medians greater than $x$ about $n/10$

2 out of 5 in each group greater than the median in the group, which is greater than $x$ about $2n/10$

About 3n/10 elts greater than $x$
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 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 \( \sim 3n/10 \) and \( \sim 7n/10 \) in the worst case.

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

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

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

if \( n \leq 5 \) then

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

DIVIDE \( S \) into \( \lceil n/5 \rceil \) groups

\[
\triangleright \text{first } \lceil n/5 \rceil \text{ groups are of size 5, last group is of size } n \mod 5
\]

FIND median set \( M = \{m_1, \ldots, m_{\lceil n/5 \rceil}\} \quad \triangleright m_j: \text{median of } j\text{-th group}

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

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

if \( i \leq |L| \) then

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

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
Choosing the Pivot

At least half of the medians \( \geq x \)

Thus \( m = \lceil n/5 \rceil / 2 \) 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

$$n - \left( \frac{3n}{10} - 6 \right) = \frac{7n}{10} + 6$$
Selection in Worst Case Linear Time

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

\[ \begin{aligned}
\text{if } n \leq 5 \text{ then} \\
\quad \text{SORT } S \text{ and return the } i\text{-th element}
\end{aligned} \]

\[ \Theta(n) \]

\[ \begin{aligned}
\text{DIVIDE } S \text{ into } \lceil n/5 \rceil \text{groups} \\
\quad \triangledown \text{first } \lceil n/5 \rceil \text{ groups are of size 5, last group is of size } n \text{ mod 5}
\end{aligned} \]

\[ \Theta(n) \]

\[ \begin{aligned}
\text{FIND median set } M=\{m_1, \ldots, m_{\lceil n/5 \rceil}\} \quad \triangledown m_j: \text{median of } j\text{-th group}
\end{aligned} \]

\[ \Theta(n) \]

\[ \begin{aligned}
x \leftarrow \text{SELECT}(M, \lceil n/5 \rceil, (\lfloor \lceil n/5 \rceil +1)/2 \rfloor) \\
\text{PARTITION set } S \text{ around the pivot } x \text{ into } L \text{ and } R
\end{aligned} \]

\[ \begin{aligned}
\text{if } i \leq |L| \text{ then} \\
\quad \text{return } \text{SELECT}(L, |L|, i)
\end{aligned} \]

\[ \begin{aligned}
\text{else} \\
\quad \text{return } \text{SELECT}(R, n-|L|, i-|L|)
\end{aligned} \]

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

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