Shell and Merge Sorts
Lecture 34
Sections 13.1, 13.4

Robb T. Koether
Hampden-Sydney College
Mon, Apr 13, 2009
Outline

1. Sorting
2. Comparison of Run Times
3. The Shell Sort
4. The Merge Sort
5. Assignment
To sort a list is to arrange the members into either increasing order or decreasing order.

A total order relation on a set is a relation that has the following two properties.

- **Transitivity**: If \( a < b \) and \( b < c \), then \( a < c \).
- **Trichotomy**: Either \( a = b \), or \( a < b \), or \( a > b \).

The order is determined by an order operator: \(<, >, \leq, \) or \(\geq\).

This operator must define a total order on the class.
Most elementary sorting algorithms are inefficient for long lists.

Examples
- Bubble Sort.
- Selection Sort.
- Insertion Sort.

These algorithms have run times of order $O(n^2)$. 
Efficient Sorting Algorithms

- The efficient sorting algorithms are more complicated.
- Examples
  - Shell Sort
  - Merge Sort
  - Quick Sort
- These algorithms have run times of order $O(n \log n)$. 
Comparison of Algorithms

How much fast is $O(n \log n)$ than $O(n^2)$? Let’s compare.

Let $A$ be an algorithm of order $O(n^2)$.

Let $B$ be an algorithm of order $O(n \log n)$.

Suppose both algorithms require 1 μsec to process a list of size $n = 100$.

How long will they take to process lists of sizes $10^3$, $10^4$, $10^5$, $10^6$, $10^7$, $10^8$, and $10^9$?
Comparison of Algorithms

- **Algorithm $A$** has run time
  \[
  \frac{n^2}{100^2} = 0.0001n^2.
  \]

- **Algorithm $B$** has run time
  \[
  \frac{n \log n}{100 \log 100} = 0.005n \log n.
  \]

- **Evaluate these functions when** $n = 10^2, 10^3, 10^4, 10^5, 10^6, 10^7, 10^8, \text{ and } 10^9$. 

Comparison of Algorithms

<table>
<thead>
<tr>
<th>n</th>
<th>Algorithm A</th>
<th>Algorithm B</th>
</tr>
</thead>
<tbody>
<tr>
<td>$10^2$</td>
<td>1 $\mu$s</td>
<td>1 $\mu$s</td>
</tr>
<tr>
<td>$10^3$</td>
<td>100 $\mu$s</td>
<td>15 $\mu$s</td>
</tr>
<tr>
<td>$10^4$</td>
<td>10 ms</td>
<td>200 $\mu$s</td>
</tr>
<tr>
<td>$10^5$</td>
<td>1 s</td>
<td>2.5 ms</td>
</tr>
<tr>
<td>$10^6$</td>
<td>100 s</td>
<td>30 ms</td>
</tr>
<tr>
<td>$10^7$</td>
<td>2.8 h</td>
<td>350 ms</td>
</tr>
<tr>
<td>$10^8$</td>
<td>11.6 d</td>
<td>4 s</td>
</tr>
<tr>
<td>$10^9$</td>
<td>3.2 y</td>
<td>45 s</td>
</tr>
</tbody>
</table>
The Shell Sort is a variation of the Insertion Sort.

- The run time is $O(n \log n)$ with probability near 100%.
- The worst case has run time $O(n^2)$ with probability near 0%.
The Shell Sort Algorithm

- Start with a “gap” equal to half the size of the list.
- Perform Insertion Sorts on the interwoven sublists ($k = \text{gap}$).

\[
\begin{align*}
\{a_0, a_k, a_{2k}, \ldots\} \\
\{a_1, a_{k+1}, a_{2k+1}, \ldots\} \\
\{a_2, a_{k+2}, a_{2k+2}, \ldots\} \\
&\quad \vdots \\
\{a_{k-1}, a_{2k-1}, a_{3k-1}, \ldots\}.
\end{align*}
\]
Example (The Shell Sort)

- Let the list be
  \[\{30, 60, 80, 20, 90, 50, 10, 70, 40\}\].
- Begin with a list of size \(n = 9\).
- Initialize gap = 4.
### Example: The Shell Sort

#### Example (The Shell Sort)

- **Pass #1, first sublist.**

<table>
<thead>
<tr>
<th>30</th>
<th>60</th>
<th>80</th>
<th>20</th>
<th>90</th>
<th>50</th>
<th>10</th>
<th>70</th>
<th>40</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>4</td>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>30</th>
<th>60</th>
<th>80</th>
<th>20</th>
<th>40</th>
<th>50</th>
<th>10</th>
<th>70</th>
<th>90</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>4</td>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Example: The Shell Sort

**Example (The Shell Sort)**

- Pass #1, 2nd sublist.

```
30 60 80 20 40 50 10 70 90
```

```
30 50 80 20 40 60 10 70 90
```

```
1 5
1 5
```
Example: The Shell Sort

Example (The Shell Sort)

- Pass #1, 3rd sublist.

<table>
<thead>
<tr>
<th>30</th>
<th>60</th>
<th>80</th>
<th>20</th>
<th>40</th>
<th>50</th>
<th>10</th>
<th>70</th>
<th>90</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>30</th>
<th>50</th>
<th>10</th>
<th>20</th>
<th>40</th>
<th>60</th>
<th>80</th>
<th>70</th>
<th>90</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Example (The Shell Sort)

- **Pass #1, 4th sublist.**

<p>| | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>30</td>
<td>60</td>
<td>80</td>
<td>20</td>
<td>40</td>
<td>50</td>
<td>10</td>
<td>70</td>
<td>90</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td>7</td>
<td></td>
</tr>
</tbody>
</table>

```
30 60 80 20 40 50 10 70 90
30 50 10 20 40 60 80 70 90
3 7
3 7
```
Example: The Shell Sort

Example (The Shell Sort)

- Divide gap by 2 and repeat the above procedure.
- Quit when gap = 0.
Example: The Shell Sort

Example (The Shell Sort)

- Halve the gap: gap = 4 / 2 = 2.
- Pass #2, 1st sublist

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>2</th>
<th>4</th>
<th>6</th>
<th>8</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>30</td>
<td>60</td>
<td>10</td>
<td>20</td>
<td>40</td>
<td>50</td>
<td>80</td>
</tr>
<tr>
<td>70</td>
<td>90</td>
<td>10</td>
<td>60</td>
<td>30</td>
<td>20</td>
<td>40</td>
</tr>
<tr>
<td>40</td>
<td>50</td>
<td>80</td>
<td>70</td>
<td>90</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Halve the gap: gap = 4 / 2 = 2.

Pass #2, 1st sublist
Example: The Shell Sort

Example (The Shell Sort)

- Halve the gap: \( \text{gap} = 4 / 2 = 2 \).
- Pass #2, 2nd sublist

<table>
<thead>
<tr>
<th></th>
<th>30</th>
<th>60</th>
<th>80</th>
<th>20</th>
<th>40</th>
<th>50</th>
<th>10</th>
<th>70</th>
<th>90</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>7</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>10</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>30</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>20</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>50</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>40</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>60</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>80</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>70</td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Assignment
Example: The Shell Sort

Example (The Shell Sort)

- Halve the gap: gap = 2 / 2 = 1.

<p>| | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>50</td>
<td>30</td>
<td>20</td>
<td>40</td>
<td>60</td>
<td>80</td>
<td>70</td>
<td>90</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>5</td>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<p>| | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>20</td>
<td>30</td>
<td>40</td>
<td>50</td>
<td>60</td>
<td>70</td>
<td>80</td>
<td>90</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>5</td>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
The Merge Sort

- Merging two sorted lists of length $n$ has run time $O(n)$.
- The run time of the Merge Sort is $O(n \log n)$. 
The Merge Sort Algorithm

- Begin by considering the list to be a collection of sublists each of length 1.

```
50 30 70 20 80 40 10 60
```

- Merge adjacent sublists in pairs.
- Continue to merge adjacent sublists until there remains only one sublist.
Example: The Merge Sort

The Merge Sort
- Begin with a list of size $n = 8$. 

50 30 70 20 80 40 10 60
Example: The Merge Sort

The Merge Sort

- Pass #1: Sublist size = 1.

```
30 50 70 20 80 40 10 60
```
Example: The Merge Sort

The Merge Sort

- Pass #2: Sublist size = 2.

Example

30 50 20 70 80 40 10 60
Example: The Merge Sort

The Merge Sort

- Pass #2: Sublist size = 2.

Example:

- 30 50 20 70 40 80 10 60
Example: The Merge Sort

The Merge Sort

- Pass #2: Sublist size = 2.

<p>| | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>30</td>
<td>50</td>
<td>20</td>
<td>70</td>
<td>40</td>
<td>80</td>
<td>10</td>
</tr>
</tbody>
</table>
Example: The Merge Sort

The Merge Sort

- Pass #3: Sublist size = 4.

```
20  30  50  70  40  80  10  60
```
Example: The Merge Sort

The Merge Sort

- Pass #3: Sublist size = 4.
Example: The Merge Sort

Example

The Merge Sort
- Pass #4: Sublist size = 8.

10 20 30 40 50 60 70 80
Assignment

Shell and Merge Sorts

Robb T. Koether

Sorting
Comparison of Run Times
The Shell Sort
The Merge Sort
Assignment

Homework

- Read Section 13.1, pages 721 - 732.
- Read Section 13.4, pages 762 - 769.