I & C SCI 46  Winter 2023
Lecture 21: Fundamentals of Sorting I
What is sorting?

- Input: sequence of \( n \) comparable values
- Reorder the input to be non-descending.
- Items we wish to sort are called “keys”
- Not here: retain associated information
Why discuss sorting?

- Standard library has sorting
- Why not use that and move on?

In this class, sorting is:
- a good intro for techniques
- a good intro to comparative algorithms
**SelectionSort**

**Idea:** Swap min into first spot, second-min to second, etc.
(This is hand-wavy on purpose)

<p>| | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>85</td>
<td>24</td>
<td>63</td>
<td>45</td>
<td>17</td>
<td>31</td>
<td>96</td>
<td>50</td>
</tr>
<tr>
<td>17</td>
<td>24</td>
<td>63</td>
<td>45</td>
<td>85</td>
<td>31</td>
<td>96</td>
<td>50</td>
</tr>
<tr>
<td>17</td>
<td>24</td>
<td>63</td>
<td>45</td>
<td>85</td>
<td>31</td>
<td>96</td>
<td>50</td>
</tr>
<tr>
<td>17</td>
<td>24</td>
<td>31</td>
<td>45</td>
<td>85</td>
<td>63</td>
<td>96</td>
<td>50</td>
</tr>
</tbody>
</table>
SelectionSort

\begin{align*}
\text{for } i & \leftarrow 1 \text{ to } n - 1 \text{ do} \\
\text{min} & \leftarrow i \\
\text{for } j & \leftarrow i + 1 \text{ to } n \text{ do} \\
\text{if } A[j] & < A[\text{min}] \text{ then} \\
\text{min} & \leftarrow j \\
\text{Swap } A[i] \text{ and } A[\text{min}] \\
\end{align*}

\text{\textbf{O}}(n^2)
What’s nice about SelectionSort?

▶ Easy to program
▶ Easy to explain
▶ Does it waste memory? \( O(1) \) plus input "in place"
▶ Does it only work for numbers?
▶ What other info do we need?
▶ Are there inputs that are sorted faster?
▶ Is there a lot of data movement? \( 2(n-1) \) a lot?
## BubbleSort

**Idea:** Think globally act locally

<table>
<thead>
<tr>
<th>shorter</th>
<th>85</th>
<th>24</th>
<th>63</th>
<th>45</th>
<th>17</th>
<th>31</th>
<th>96</th>
<th>50</th>
</tr>
</thead>
<tbody>
<tr>
<td>taller</td>
<td>24</td>
<td>63</td>
<td>45</td>
<td>17</td>
<td>31</td>
<td>85</td>
<td>50</td>
<td>96</td>
</tr>
<tr>
<td></td>
<td>24</td>
<td>45</td>
<td>17</td>
<td>31</td>
<td>63</td>
<td>50</td>
<td>85</td>
<td>96</td>
</tr>
<tr>
<td></td>
<td>24</td>
<td>17</td>
<td>31</td>
<td>45</td>
<td>50</td>
<td>63</td>
<td>85</td>
<td>96</td>
</tr>
</tbody>
</table>
BubbleSort

\[
\begin{align*}
\text{for } i & \leftarrow 1 \text{ to } n - 1 \text{ do} \\
\text{for } j & \leftarrow 1 \text{ to } n - i \text{ do} \\
\text{if } A[j + 1] < A[j] \text{ then} \\
& \quad \text{Swap } A[j] \text{ and } A[j + 1]
\end{align*}
\]

\[O(n^2)\]
## InsertionSort

### Idea:

<table>
<thead>
<tr>
<th>85</th>
<th>24</th>
<th>63</th>
<th>45</th>
<th>17</th>
<th>31</th>
<th>96</th>
<th>50</th>
</tr>
</thead>
<tbody>
<tr>
<td>24</td>
<td>85</td>
<td>63</td>
<td>45</td>
<td>17</td>
<td>31</td>
<td>96</td>
<td>50</td>
</tr>
<tr>
<td>24</td>
<td>63</td>
<td>85</td>
<td>45</td>
<td>17</td>
<td>31</td>
<td>96</td>
<td>50</td>
</tr>
<tr>
<td>24</td>
<td>45</td>
<td>63</td>
<td>85</td>
<td>17</td>
<td>31</td>
<td>96</td>
<td>50</td>
</tr>
</tbody>
</table>
InsertionSort

for $j \leftarrow 2$ to $n$ do
key $\leftarrow A[j]$
$i \leftarrow j - 1$
while $i > 0$ and $A[i] >$ key do
    $A[i + 1] \leftarrow A[i]$
    $i = i - 1$
$A[i + 1] \leftarrow$ key

What is the running time of InsertionSort?

$O(n + d)$ number of inverted pairs in input.
InsertionSort

for $j \leftarrow 2$ to $n$ do
key $\leftarrow A[j]$
$i \leftarrow j - 1$
while $i > 0$ and $A[i] > key$ do
    $A[i + 1] \leftarrow A[i]$
    $i = i - 1$
$A[i + 1] \leftarrow key$

Why is InsertionSort correct?

What is true every time we check the for loop? (including the time we find $j > n$ and stop)
About that running time ...

- Why are we so concerned with worst case?

- Why not examine average case?