Today

- Insertion Sort (cont)
- Bubble Sort

Assignments

- HW6 due today
- HW7 out (due Tues)
(Linear) Insertion Sort

The basic idea (note: beginning of list is sorted, rest is unsorted)

- Select first item (in unsorted region)
- Insert item into correct position in sorted region (shifting larger items over)
- Stop when unsorted region is empty

Pass 1:

\[
\begin{array}{cccc}
29 & 10 & 14 & 13 \\
\Rightarrow & 10 & 29 & 14 & 13 \\
\end{array}
\]

Pass 2:

\[
\begin{array}{cccc}
10 & 29 & 14 & 13 \\
\Rightarrow & 10 & 14 & 29 & 13 \\
\end{array}
\]

Pass 3:

\[
\begin{array}{cccc}
10 & 14 & 29 & 13 \\
\Rightarrow & 10 & 13 & 14 & 29 \\
\end{array}
\]

The algorithm (psuedocode):

\[
\text{LinearInsertionSort}(T \text{ array}[], \text{ int } n) \\
2. \text{ for } i = 1 \text{ to } n-1 \text{ do } \quad \# \text{n-1 passes} \\
3. \quad \text{val := array}[i] \quad \# \text{val to insert} \\
4. \quad j := i \quad \# \text{val to insert} \\
5. \quad \text{while } j > 0 \text{ and array}[j-1] > \text{val do} \quad \# \text{until insert spot found} \\
6. \quad \quad \text{array}[j] = \text{array}[j-1] \quad \# \text{shift forward} \\
7. \quad \quad j := j - 1 \quad \# \text{place val}
\]
Q: What are the best and worst cases?

- **Best case**: List already sorted (no shifts, just n-1 passes)
- **Worst case**: List in reverse order (shift max number of times each pass)

Q: What is the worst-case time?

- Do \((n - 1)\) passes
- Each pass requires \(p\) comparisons (and moves)
- Number of array comparisons:
  \[
  \sum_{p=1}^{n-1} p = \frac{n(n - 1)}{2}
  \]
  - Thus, insertion sort is \(O(n^2)\)

Q: What is the best-case time?

- Each pass only do one comparison (and two moves)
- Number of array comparisons:
  \[
  \sum_{i=1}^{n-1} 1 = n - 1
  \]
  - Thus, in the best case insertion sort is \(O(n)\)
A variant: Binary Insertion Sort

Instead of finding insertion index iteratively, use binary search!

- Thus, we do $O(\log n)$ comparisons instead of $O(n)$ w/ linear insertion sort

However, insertion sort is still worst-case $O(n^2)$ ... why?

- we still have to do $O(n)$ shifts (moves) in the worst case
- another example of why it is good to count moves and insertions separately

<table>
<thead>
<tr>
<th>Sorting Alg</th>
<th>Best Case</th>
<th>Avg Case</th>
<th>Worst Case</th>
</tr>
</thead>
<tbody>
<tr>
<td>Selection Sort</td>
<td>$O(n^2)$</td>
<td>$O(n^2)$</td>
<td>$O(n^2)$</td>
</tr>
<tr>
<td>Insertion Sort</td>
<td>$O(n)$</td>
<td>$O(n^2)$</td>
<td>$O(n^2)$</td>
</tr>
<tr>
<td>Bubble Sort</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Merge Sort</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Quick Sort</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tree Sort</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Heap Sort</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Bubble Sort

The basic idea (note: unsorted region on left, sorted on right)

- Compare adjacent items
- Swap if out of order
- Repeat with items 0 to \( n - 2 \)
- Stop when only one item left

Pass 1:

\[
\begin{array}{cccc}
29 & 10 & 14 & 13 \\
\end{array} \Rightarrow
\begin{array}{cccc}
10 & 29 & 14 & 13 \\
\end{array} \Rightarrow
\begin{array}{cccc}
10 & 14 & 29 & 13 \\
\end{array} \Rightarrow
\begin{array}{cccc}
10 & 14 & 13 & 29 \\
\end{array}
\]

Pass 2:

\[
\begin{array}{cccc}
10 & 14 & 13 & 29 \\
\end{array} \Rightarrow
\begin{array}{cccc}
10 & 14 & 13 & 29 \\
\end{array} \Rightarrow
\begin{array}{cccc}
10 & 13 & 14 & 29 \\
\end{array}
\]

Pass 3:

\[
\begin{array}{cccc}
10 & 13 & 14 & 29 \\
\end{array} \Rightarrow
\begin{array}{cccc}
10 & 13 & 14 & 29 \\
\end{array}
\]

The algorithm (psuedocode): With short-circuit check

```python
BubbleSort(T array[], int n)
1. sorted := false  # done early
2. for i = 1 to n-1 and not sorted do  # n-1 passes
3.     sorted := true  # assume sorted
4.     for j = 1 to n-i do  # n-i iterations
5.         if array[j-1] > array[j] then  # 1 comparison
6.             swap(array[j-1], array[j])  # 3 moves
7.             sorted := false  # swap needed
```
Q: What are the best and worst cases for bubble sort?

- Note that bubble sort has a “short circuit”
- **Best case:** Already sorted (requires only one pass of comparisons)
- **Worst case:** List in reverse sorted order (swap all pairs in every pass)

\[
\sum_{i=1}^{n-1}(n - i) = \frac{n(n - 1)}{2} \quad \text{which is } O(n^2)
\]

<table>
<thead>
<tr>
<th>Sorting Alg</th>
<th>Best Case</th>
<th>Avg Case</th>
<th>Worst Case</th>
</tr>
</thead>
<tbody>
<tr>
<td>Selection Sort</td>
<td>(O(n^2))</td>
<td>(O(n^2))</td>
<td>(O(n^2))</td>
</tr>
<tr>
<td>Insertion Sort</td>
<td>(O(n))</td>
<td>(O(n^2))</td>
<td>(O(n^2))</td>
</tr>
<tr>
<td>Bubble Sort</td>
<td>(O(n))</td>
<td>(O(n^2))</td>
<td>(O(n^2))</td>
</tr>
<tr>
<td>Merge Sort</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Quick Sort</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tree Sort</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Heap Sort</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>