## Evaluation

- **Exam/Competition**: 50%
- **Solving Problems**: 40%
- **Attendance**: 5%
- **Class Participation**: 5%

9/1/11  COP 4516
Sorting

• Input is a list of $n$ items that can be compared.
• Output is an ordered list of those $n$ items.
• Fundamental problem that has received a lot of attention over the years.
• Used in many applications.
• Scores of different algorithms exist.
• Task: To compare algorithms
  - On what bases?
    • Time
    • Space
    • Other
Sorting Algorithms

• Number of Comparisons
• Number of Data Movements
• Additional Space Requirements
Sorting Algorithms

- SelectionSort
- InsertionSort
- BubbleSort
- ShakerSort
- MergeSort
- HeapSort
- QuickSort
- Bucket & Radix Sort
- Counting Sort
**SelectionSort**

1. $N \leftarrow \text{length}[A]$
2. **for** $p \leftarrow 1$ **to** $N$
3. **do** Compute $j$, the index of the smallest item in $A[p..N]$
4. **swap** $A[p]$ and $A[j]$
SelectionSort

\textbf{SELECTIONSORT}(array \textit{A})

1 \hspace{0.5cm} N \leftarrow \text{length}[\textit{A}]
2 \textbf{for} \hspace{0.3cm} p \leftarrow 1 \textbf{ to } N
3 \hspace{0.5cm} \textbf{do} \triangleright \text{Compute } j
4 \hspace{0.5cm} \hspace{0.5cm} j \leftarrow p
5 \hspace{0.5cm} \textbf{for} \hspace{0.3cm} m \leftarrow p + 1 \textbf{ to } N
6 \hspace{0.5cm} \hspace{0.5cm} \textbf{do \ if} (A[m] < A[j])
7 \hspace{0.5cm} \hspace{0.5cm} \hspace{0.5cm} \triangleright \text{Swap } A[p] \text{ and } A[j]
8 \hspace{0.5cm} \hspace{0.5cm} \hspace{0.5cm} \text{temp} \leftarrow A[p]
9 \hspace{0.5cm} \hspace{0.5cm} \hspace{0.5cm} \hspace{0.5cm} A[p] \leftarrow A[j]
10 \hspace{0.5cm} \hspace{0.5cm} \hspace{0.5cm} \hspace{0.5cm} A[j] \leftarrow \text{temp}
SelectionSort

\texttt{SelectionSort}(array \ A)

1. \( N \leftarrow \text{length}[A] \)
2. \textbf{for} \( p \leftarrow 1 \) \textbf{to} \( N \)
   \> Compute \( j \)
3. \( j \leftarrow p \)
4. \textbf{for} \( m \leftarrow p + 1 \) \textbf{to} \( N \)
   \> \textbf{do} if \( (A[m] < A[j]) \)
5. \> \> then \( j \leftarrow m \)
\> Swap \( A[p] \) and \( A[j] \)
7. \texttt{temp} \leftarrow A[p]
9. \( A[j] \leftarrow \texttt{temp} \)

O(n²) time
O(1) space
## Solving Recurrence Relations

Page 62, [CLR]

<table>
<thead>
<tr>
<th>Recurrence; Cond</th>
<th>Solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>$T(n) = T(n-1) + O(1)$</td>
<td>$T(n) = O(n)$</td>
</tr>
<tr>
<td>$T(n) = T(n-1) + O(n)$</td>
<td>$T(n) = O(n^2)$</td>
</tr>
<tr>
<td>$T(n) = T(n-c) + O(1)$</td>
<td>$T(n) = O(n)$</td>
</tr>
<tr>
<td>$T(n) = T(n-c) + O(n)$</td>
<td>$T(n) = O(n^2)$</td>
</tr>
<tr>
<td>$T(n) = 2T(n/2) + O(n)$</td>
<td>$T(n) = O(n \log n)$</td>
</tr>
<tr>
<td>$T(n) = aT(n/b) + O(n)$; $a = b$</td>
<td>$T(n) = O(n \log n)$</td>
</tr>
<tr>
<td>$T(n) = aT(n/b) + O(n)$; $a &lt; b$</td>
<td>$T(n) = O(n)$</td>
</tr>
<tr>
<td>$T(n) = aT(n/b) + f(n)$; $f(n) = O(n^{\log_b a - \epsilon})$</td>
<td>$T(n) = \Theta(n^{\log_b a})$</td>
</tr>
<tr>
<td>$T(n) = aT(n/b) + f(n)$; $f(n) = \Theta(f(n))$</td>
<td>$T(n) = O(n)$</td>
</tr>
<tr>
<td>$a f(n/b) \leq c f(n)$</td>
<td>$T(n) = \Omega(n^{\log_b a \log n})$</td>
</tr>
</tbody>
</table>

9/1/11
**Insertion-Sort**($A$)

1. for $j \leftarrow 2$ to $\text{length}[A]$
2. do $\text{key} \leftarrow A[j]$
3. $\triangleright$ Insert $A[j]$ into the sorted sequence $A[1 \ldots j - 1]$
4. $i \leftarrow j - 1$
5. while $i > 0$ and $A[i] > \text{key}$
6. do $A[i + 1] \leftarrow A[i]$
7. $i \leftarrow i - 1$
8. $A[i + 1] \leftarrow \text{key}$

Loop invariants and the correctness of insertion sort
**Insertion-Sort(A)**

1. for $j \leftarrow 2$ to length[$A$]
2. \hspace{1em} do $key \leftarrow A[j]$
3. \hspace{2em} $\triangleright$ Insert $A[j]$ into the sorted sequence $A[1 \ldots j - 1]$.
4. $i \leftarrow j - 1$
5. while $i > 0$ and $A[i] > key$
6. \hspace{1em} do $A[i + 1] \leftarrow A[i]$
7. \hspace{2em} $i \leftarrow i - 1$
8. $A[i + 1] \leftarrow key$

<table>
<thead>
<tr>
<th>cost</th>
<th>times</th>
</tr>
</thead>
<tbody>
<tr>
<td>$c_1$</td>
<td>$n$</td>
</tr>
<tr>
<td>$c_2$</td>
<td>$n - 1$</td>
</tr>
<tr>
<td>$c_3$</td>
<td>$n - 1$</td>
</tr>
<tr>
<td>$c_4$</td>
<td>$\sum_{j=2}^{n} t_j$</td>
</tr>
<tr>
<td>$c_5$</td>
<td>$\sum_{j=2}^{n} (t_j - 1)$</td>
</tr>
<tr>
<td>$c_6$</td>
<td>$\sum_{j=2}^{n} (t_j - 1)$</td>
</tr>
<tr>
<td>$c_7$</td>
<td>$n - 1$</td>
</tr>
</tbody>
</table>

$O(n^2)$ time

$O(1)$ space
Figure 2.4 The operation of merge sort on the array $A = (5, 2, 4, 7, 1, 3, 2, 6)$. The lengths of the sorted sequences being merged increase as the algorithm progresses from bottom to top.
Figure 2.3 The operation of lines 10-17 in the call \textsc{merge}(A[9..16]), when the subarray A[9..16] contains the sequence (2, 4, 5, 7, 1, 2, 3, 6). After copying and inserting sentinels, the array \textit{L} contains (2, 4, 5, 7, \infty), and the array \textit{R} contains (1, 2, 3, 6, \infty). Lightly shaded positions in \textit{A} contain their final values, and lightly shaded positions in \textit{L} and \textit{R} contain values that have yet to be copied back into \textit{A}. Taken together, the lightly shaded positions always comprise the values originally in A[9..16], along with the two sentinels. Heavily shaded positions in \textit{A} contain values that will be copied over, and heavily shaded positions in \textit{L} and \textit{R} contain values that have already been copied back into \textit{A}. (a)-(h) The arrays \textit{A}, \textit{L}, and \textit{R}, and their respective indices \textit{k}, \textit{i}, and \textit{j} prior to each iteration of the loop of lines 12-17. (i) The arrays and indices at termination. At this point, the subarray in A[9..16] is sorted, and the two sentinels in \textit{L} and \textit{R} are the only two elements in these arrays that have not been copied into \textit{A}. 
Assumption: Array A is sorted from positions p to q and also from positions q+1 to r.

```plaintext
MERGE(A, p, q, r)
1   n₁ ← q − p + 1
2   n₂ ← r − q
3   create arrays L[1..n₁ + 1] and R[1..n₂ + 1]
4   for i ← 1 to n₁
5       do L[i] ← A[p + i − 1]
6   for j ← 1 to n₂
7       do R[j] ← A[q + j]
8   L[n₁ + 1] ← ∞
9   R[n₂ + 1] ← ∞
10  i ← 1
11  j ← 1
12  for k ← p to r
13      do if L[i] ≤ R[j]
14         then A[k] ← L[i]
15         i ← i + 1
16      else A[k] ← R[j]
17         j ← j + 1
```
MERGE-SORT($A$, $p$, $r$)

1. if $p < r$
2. then $q \leftarrow \lfloor (p + r)/2 \rfloor$
3. MERGE-SORT($A$, $p$, $q$)
4. MERGE-SORT($A$, $q + 1$, $r$)
5. MERGE($A$, $p$, $q$, $r$)
Figure 2.5 The construction of a recursion tree for the recurrence $T(n) = 2T(n/2) + cn$. Part (a) shows $T(n)$, which is progressively expanded in (b)–(d) to form the recursion tree. The fully expanded tree in part (d) has $\lg n + 1$ levels (i.e., it has height $\lg n$, as indicated), and each level contributes a total cost of $cn$. The total cost is $cn\lg n + cn$, which is $\Theta(n\lg n)$. 
Sorting Algorithms

• SelectionSort
• InsertionSort
• BubbleSort
• ShakerSort
• MergeSort
• HeapSort
• QuickSort
• Bucket & Radix Sort
• Counting Sort
Animations

- http://cg.scs.carleton.ca/~morin/misc/sortalg/
- http://home.westman.wave.ca/~rhenry/sort/
  - time complexities on best, worst and average case
  - runs on almost sorted, reverse, random, and unique inputs; shows code with invariants
  - comparisons, movements & stepwise animations with user data
- http://maven.smith.edu/~thiebaut/java/sort/demo.html
  - comparisons & data movements and step by step execution
Comparing $O(n^2)$ Sorting Algorithms

- InsertionSort and SelectionSort (and ShakerSort) are roughly twice as fast as BubbleSort for small files.
- InsertionSort is the best for very small files.
- $O(n^2)$ sorting algorithms are NOT useful for large random files.
- If comparisons are very expensive, then among the $O(n^2)$ sorting algorithms, insertion sort is best.
- If data movements are very expensive, then among the $O(n^2)$ sorting algorithms, ?? is best.
Selection

- Given a set of $n$ items and a number $k$, select the $k^{th}$ smallest item from the set.
  - $k = 1$
  - $k = n$
  - $k = n/2$
  - Arbitrary $k$

- General Solution:
  - Sort, then select
Problems to think about!

• What is the least number of comparisons you need to sort a list of 3 elements? 4 elements? 5 elements?

• How to arrange a tennis tournament in order to find the tournament champion with the least number of matches? How many tennis matches are needed?

• How to randomize the order of a list?
Search

- Given a set of $n$ items, search for item $x$
  - Unordered list
  - Ordered list
    - Array list
    - Linked List
    - ??