Chapter 3, Case Studies:

- Linear-time sorting algorithms (Today and Wednesday)
- Disjoint sets and array forests (Friday)
- Priority queues (Next week Monday)
- N-sets and bit vectors (Next week Wednesday)

Today (and Wednesday):

- Recent quiz problem
- Iterators in adapter data structures
- Intro to “case studies”
- Limitations of comparison-based sorting
- Counting sort
- Radix sort
ArrayList

get()

set()

add()

LinkedList
Can’t you tell a good tree from a poor tree?

I have just been thinking, and I have come to a very important decision. These are the wrong sort of bees.

**Good sorts**
- Merge
- Quick (expected case)
- Shell (unassigned project)
- Heap (Section 3.3)

**Bad sorts**
- Selection
- Insertion
- Bubble
The ultimate measure for algorithms is their running time, but counting the number of comparisons between elements in the sequence is a reasonable proxy for running time when comparing sorting algorithms. At least we can say that if the expected case of a sorting algorithm makes $\Theta(n \lg n)$ comparisons, then its running time must be $\Omega(n \lg n)$—that is, it can’t be better than $n \lg n$.

It turns out that $\Theta(n \lg n)$ is in fact the best we can do for the expected case of sorting algorithms that make decisions by comparing elements in the sequence. Put formally,

**Theorem 1.** If $T$ is an algorithm that sorts a sequence by comparing pairs of elements in the sequence, then the expected running time of $T$ is $\Omega(n \lg n)$. 
0. Alice 0
1. Bob 2
2. Carol 4
3. Dave 4
4. Eve 2
5. Fred 0
6. Georgia 0
7. Henry 1
8. Ida 4
9. Jack 2
10. Karen 4
11. Larry 0
12. Moira 2
13. Nate 3
14. Olivia 1
15. Pete 1
16. Queenie 1
17. Ralph 4
18. Sara 2
19. Trent 4
20. Ursulla 2
21. Vick 3
22. Wendy 1
23. Xavier 2
24. Yvette 0
25. Zeke 3
Coming up:

Do “basic data structures” practice problems (suggested by today, Mon, Jan 31)
Do “implementing ADTs” project (suggested by Wed, Feb 2)

Due **Wed, Feb 2**: (class time)
Read Section 3.1
Do Exercises 2.(21–23)
Take sorting quiz

Due **Fri, Feb 4**: (end of day)
Read Section 3.2
Do Exercises 2.(12 & 15) and 3.8.
Take disjoint sets quiz
Invariant (Loop of radix_sort)

(a) \(i\) is the number of iterations completed.
(b) \(r_{\text{pow}} = r^i\).
(c) \(\forall k \in [0, n-1), \text{sequence}[k] \mod r^i \leq \text{sequence}[k+1] \mod r^i\)
<p>| 0110 | 1011 | 1100 | 0111 | 0001 | 1110 | 1001 | 1101 |</p>
<table>
<thead>
<tr>
<th>0110</th>
<th>1011</th>
<th>1100</th>
<th>0111</th>
<th>0001</th>
<th>1110</th>
<th>1001</th>
<th>1101</th>
</tr>
</thead>
<tbody>
<tr>
<td>1100</td>
<td>0001</td>
<td>1001</td>
<td>1101</td>
<td>0110</td>
<td>1110</td>
<td>1011</td>
<td>0111</td>
</tr>
<tr>
<td>0110</td>
<td>1100</td>
<td>1110</td>
<td>1011</td>
<td>0111</td>
<td>0001</td>
<td>1001</td>
<td>1101</td>
</tr>
<tr>
<td>1100</td>
<td>0001</td>
<td>1001</td>
<td>1101</td>
<td>0110</td>
<td>1110</td>
<td>1011</td>
<td>0111</td>
</tr>
<tr>
<td>0110</td>
<td>1011</td>
<td>1100</td>
<td>0111</td>
<td>0001</td>
<td>1110</td>
<td>1001</td>
<td>1101</td>
</tr>
<tr>
<td>------</td>
<td>------</td>
<td>------</td>
<td>------</td>
<td>------</td>
<td>------</td>
<td>------</td>
<td>------</td>
</tr>
<tr>
<td>0110</td>
<td>1100</td>
<td>1110</td>
<td>1011</td>
<td>0111</td>
<td>0001</td>
<td>1001</td>
<td>1101</td>
</tr>
<tr>
<td>1100</td>
<td>0001</td>
<td>1001</td>
<td>1101</td>
<td>0110</td>
<td>1110</td>
<td>1011</td>
<td>0111</td>
</tr>
<tr>
<td>0001</td>
<td>1001</td>
<td>1011</td>
<td>1100</td>
<td>1101</td>
<td>0110</td>
<td>1110</td>
<td>0111</td>
</tr>
<tr>
<td>0001</td>
<td>0110</td>
<td>0111</td>
<td>1001</td>
<td>1011</td>
<td>1100</td>
<td>1101</td>
<td>1110</td>
</tr>
</tbody>
</table>