Chapter 3, Case Studies:

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

Today (and Friday):

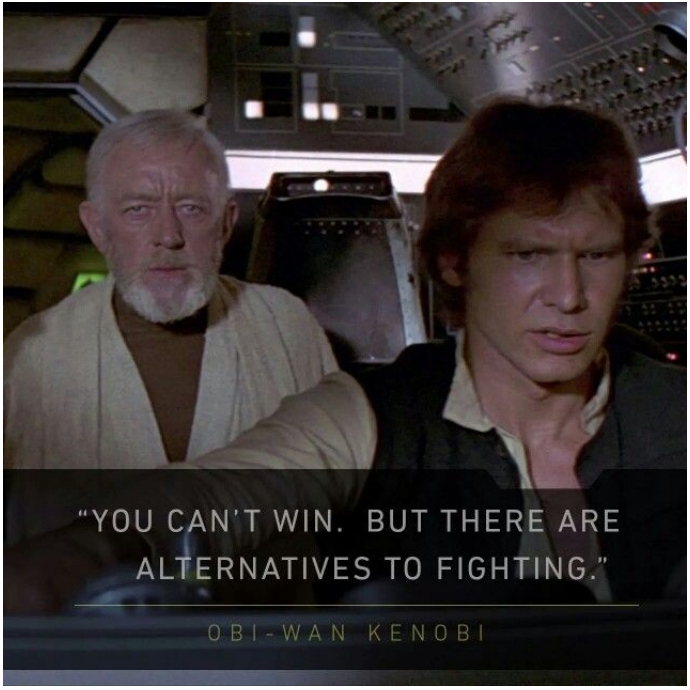
- Iterators in adapter data structures
- Recent quiz problem
- Intro to "case studies"
- Limitations of comparison-based sorting
- Counting sort
- (Lab: Bucket sort)
- Radix sort

	ArrayList	LinkedList
`get()`		
`set()`		
`add()`		

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	**Bad sorts**
Merge	Selection
Quick (expected case)	Insertion
Shell (unassigned project)	Bubble
Heap (Section 3.3)	



You can't comparison-sort in linear time. But there are alternatives to comparisons.

Meme from https://www.pinterest.com/pin/561542647262613858/

1 0 1 1 4 0 2 1 3 0 1 1 3 2 2 1 2 1 4 0 4 2 3 1 1 2 1 1 2 1 3 2 4 0 4

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 **Implementing ADTs Project** *(due Fri, Sept 19)*
Do **Linear Sorting Project** *(due next Tues, Sept 10)*

Due **Fri, Sept 19***:*
Read Section 3.1
Do Exercises 2.(22–24)
Take sorting quiz

Due **Mon, Sept 22***:*
Read Section 3.2
Do Exercises 2.(12 & 16) and 3.8.
Take disjoint sets quiz

Invariant (Loop of `radix_sort`)

(a) i *is the number of iterations completed.*

(b) $\texttt{r\_pow} = \texttt{r}^{\texttt{i}}$.

(c) $\forall\ k \in [0, n-1), \texttt{sequence}[k] \mod \texttt{r}^{\texttt{i}} \leq \texttt{sequence}[k+1] \mod \texttt{r}^{\texttt{i}}$

0110	1011	1100	0111	0001	1110	1001	1101

0110	1011	1100	0111	0001	1110	1001	1101

0110	1100	1110	1011	0111	0001	1001	1101

0110	1011	1100	0111	0001	1110	1001	1101

0110	1100	1110	1011	0111	0001	1001	1101

1100	0001	1001	1101	0110	1110	1011	0111

0110	1011	1100	0111	0001	1110	1001	1101

0110	1100	1110	1011	0111	0001	1001	1101

1100	0001	1001	1101	0110	1110	1011	0111

0001	1001	1011	1100	1101	0110	1110	0111

0110	1011	1100	0111	0001	1110	1001	1101

0110	1100	1110	1011	0111	0001	1001	1101

1100	0001	1001	1101	0110	1110	1011	0111

0001	1001	1011	1100	1101	0110	1110	0111

0001	0110	0111	1001	1011	1100	1101	1110