- Discuss difficulty with test
  - Curve is possible
- Java was difficult
- Method operation was confusing
- Request for more practice questions

Time vs space trade-offs:
- Sometimes you can see a speed increase by increasing

Data cleanup problem:
- Remove φ’s from dataset
  n = 8
  (5 4 9 4 1 4 2 1 3) Input
  (2 1 4 9 1 2 1 3) Output

Algorithm 1: Sort (ends up grouping the features together and you can remove them all at once at the end)
Algorithm 2: Scan left to right and for each φ move the rest of the array one position to the left
  - preserve the original order of the data

21 6 4 9 0 6 2 1 3
  1 1
  ← Shift 1 to left, remove φ

21 4 6 9 0 6 2 1 3
  1 1
  ← Shift & remove φ

21 4 9 0 6 2 1 3
  1 1
  ← Shift & remove φ

21 4 9 0 2 1 3
  1 1
  ← Shift & remove φ

21 4 9 2 1 3
  1 1
  Done
Algorithm 1

- worst case $O(n^2)$

  This is the case when all entries are $\neq n$
  - $n$: comparison operations
  - $n-1$: insert operations (potentially)

Algorithm 2

  In reality this is really $O(n^2)$
  also $O(n^2)$ with selection sort

Algorithm 2.

2 1 0 4 5 0 0 2 13 — INPUT ARRAY

BASIC IDEA — use MORE SPACE. IE: a second array

1 1 1 1 1 1 1 1 1 — OUTPUT ARRAY

Copy all non-zero entries from the input array to the output array

Scan — $O(n)$
Copy — $O(n)$ in the worst case if no $0$ entries

TOTAL: $O(n) + O(n) = O(2n) = O(n)$

Copy

2 1 0 4 5 0 0 2 13 — INPUT ARRAY

↑ Scan ↑ ↑ ↓ Copy ↑ ↑

2 1 0 4 5 0 0 2 13 — OUTPUT ARRAY

↑ in this instance there are 8 scan operations
and 5 copy operations

- this change increases the speed by an order of magnitude
  over Algorithm 1 and Algorithm 1
  Algorithm 1 is quadratic $O(n^2)$
  Algorithm 2 is linear $O(n)$

- this change doubles the space complexity from
  $O(n)$ in Algorithm 1 to $O(2n)$ in Algorithm 2
  however $O(n) \neq O(2n)$ have the same magnitude
  so Algorithm 2 is $O(n)$ space complexity
Algorithm 3

Can we combine Algorithm 1 + Algorithm 2: Yes.

Set two markers A & B

1. scan left to right from A to B to find φ
2. when we find φ swap φ with value at location B
3. decrement B

Repeat 1-3 until all non-zero are on the left & all zeros are on the right

10. A & B mark the same place in the array

This is NOT a stable algorithm.

(this means that it is not order preserving)

This does at most n comparisons — O(n)
This does at most n-1 swaps — O(n - 1)

Total is O(n) + O(n - 1) = O(n) time complexity.

This does all operations in place on the array.
So no extra space is needed for O(n) space complexity.

— Best case is all φ entries.
— Worst case is first half is φ second half is non-zero

Read and of Chapter 3.

Be on the lookout for Assignment 4.