Slide 1

Administrivia

- Reminder: Quiz 3 Wednesday. Likely topic arrays.
- Homework 5 was to be due Wednesday. I am adding another problem and moving the due date to next Monday. (I'll send mail when the second problem is ready to work on.)

Slide 2

Sorting and Searching — Executive-Level Summary

- Both are nice examples of problems that can be solved in various different ways (different "algorithms").
- Useful to know details of at least one, but in practice likely that you would use a library function.

Slide 3

"Order of Magnitude" of Functions — Executive-Level Summary

- Idea here is to categorize algorithms by how execution time (or other measures, e.g., amount of memory required) scales with problem size, for large problems.
- Can help rule out algorithms that would not be practical/feasible for large problems.

  A famous(?) example — "traveling salesperson problem", for which all known algorithms require considering, for N cities, all possible permutations, making them $O(N!)$. Not reasonable! (Worth noting that there apparently *are* practical approximations. Still!)

Slide 4

Text Data — Single Characters

- We have not talked much about it, but C does have a data type for text data — `char`. Big enough to represent ASCII (7-bit encoding) and other about-equal-size encodings (e.g., EBCDIC). Newer standards also provide support for "wide characters".
- Can use `scanf` and `printf`, but simpler and more efficient to use `getchar()` and `putchar()`. Worth noting that the input functions read *all* input characters, including whitespace and end-of-line.
- Many standard-library functions for working with `char`, e.g., `isspace`.

Text Data — Strings

- Most more-recent languages have nice ways of working with “strings” of text data that hide details and provide nice functionality.
- C, in contrast, provides a bare-bones version, in which text strings are represented as arrays of `char`, with an end-of-string character (`'\0'`) that allows an array of fixed size to store strings of different sizes.

  Simple but subject to all the perils of arrays!
- C standard library includes functions for working with (its version of) strings, but most must be used with care.

Text Strings — Output

- Can use `printf` with `%s`.
- Can also use `puts` (which adds a newline).

Slide 7

Text Strings — Input

- Surprisingly (or not, given C's minimalist implementation of arrays), no nice way to do this!
- Can use `scanf`, but no nice/general way to be sure you don't overflow array, and getting something that includes whitespace may be tricky.
- Can get a whole line with `fgets`, but must provide a fixed-size array (so, what size?) and deal with newlines.

  `gets` looks nice but observe what its `man` page says(!).
- Consider processing data character by character, or using command-line arguments (next time?).

Slide 8

Minute Essay

- None really — sign in.