http://www.cs.cornell.edu/courses/cs1110/2020sp

Lecture 19: Programming Practice (review list, for-loop, recursion)

CS 1110

Introduction to Computing Using Python

[E. Andersen, A. Bracy, D. Fan, D. Gries, L. Lee, S. Marschner, C. Van Loan, W. White]

Some live coding today

- Practice developing code
  - Will use Atom, command line, diagrams
  - Will experiment (just try things out!)
  - Watch me make and correct mistakes. It's cool!
- Review list, for-loop, recursion
- Demonstrate two sorting algorithms. Think of them as applications of list, loop, recursion—you don't need to know these algorithms. But know that recursion is awesome for sorting ☺
- Show why defining our own *custom classes* may be useful (next topic)

Find min value in a list

- … without using built-in min function
- We can come up with our own algorithm!
- Good opportunity to review list and for-loop

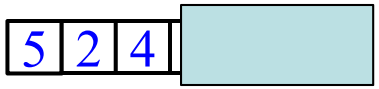
At each step check against min-so-far, NOT to previous value

Suppose you see only one value of the list at a time:

What's the min value so far? 5

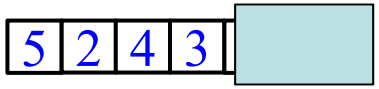
What's the min value so far? 2

Compare new value to min-so-far, update min-so-far



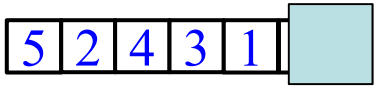
What's the min value so far? 2

Compare new value to min-so-far, keep min-so-far



What's the min value so far? 2

Compare new value to min-so-far, keep min-so-far



What's the min value so far? 1

Compare new value to min-so-far, update min-so-far

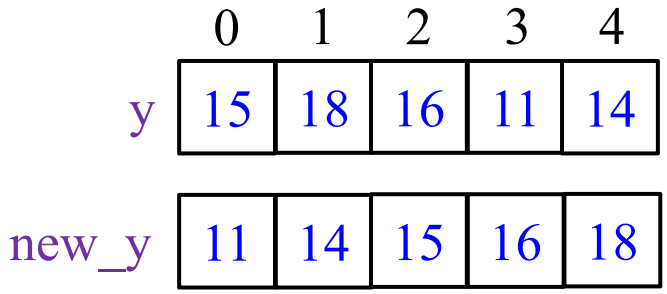
What's the min value so far? 1

Compare new value to min-so-far, keep min-so-far

See coding demo on video

Simple idea for sorting

- Pick the smallest value
- Put it at index 0 of a new list



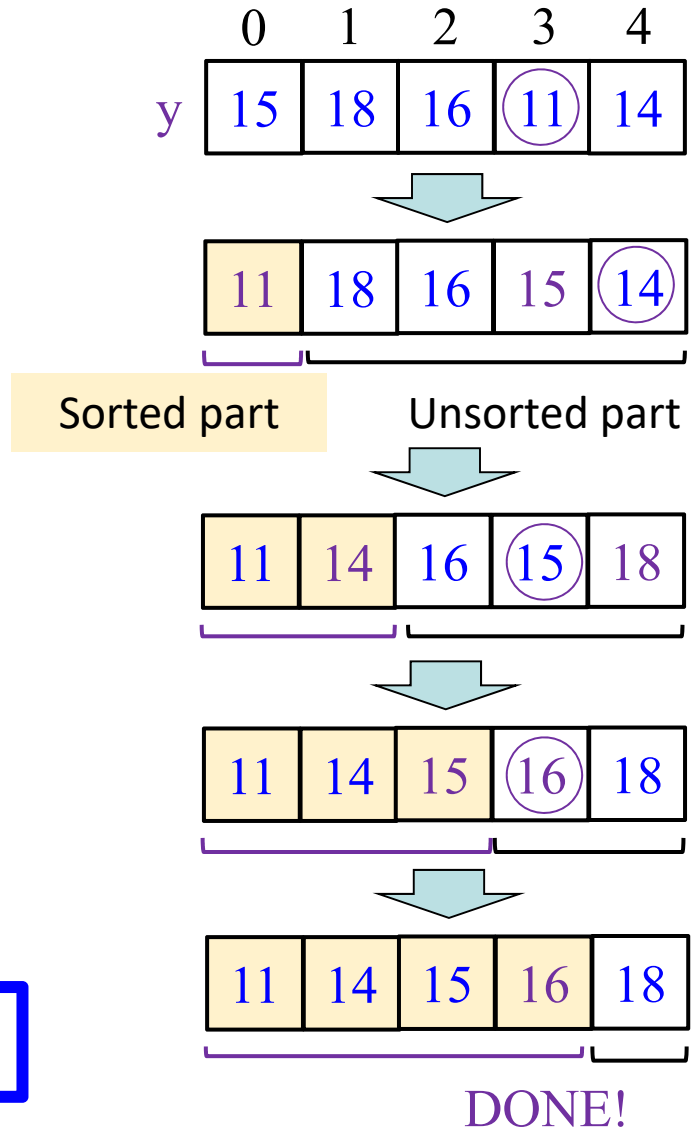
- Pick next smallest value

- Put it in the next position
- …

Can we make do without a whole other list?

- Pick the smallest value
- ~~Put it at index 0 of a new list~~
- **Swap** it with element at index 0. Use same list!
- ~~Pick next smallest value~~
- Pick smallest value starting at index 1—in unsorted part
- ~~Put it in the next position~~
- **Swap** it with element at index 1—start of unsorted part
- …

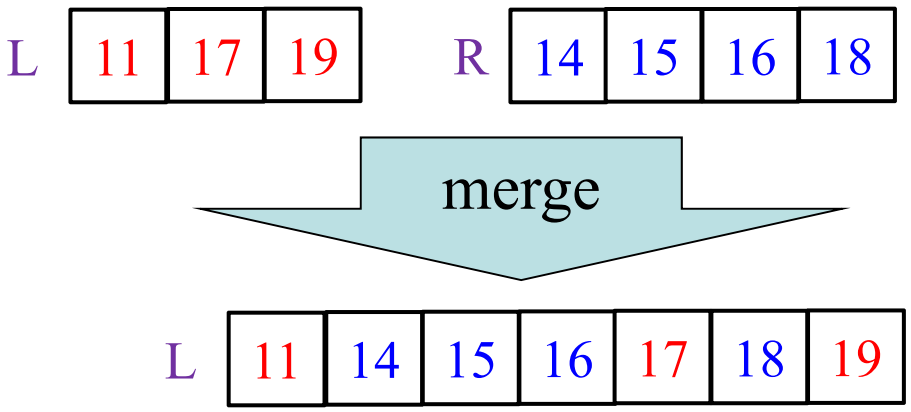
Selection Sort



See coding demo on video

Which algorithm does Python's sort use?

- Recursive algorithm that runs much faster than selection sort for the same size list (when the size is big)!
- A variant of an algorithm called "merge sort"
- Based on the idea that sorting is hard, but *"merging" two already sorted lists is easy*.



I give you function merge. (Straight forward but requires a kind of loop that you haven't seen yet.)

Let's think about the recursive aspect!

Merge sort: Motivation

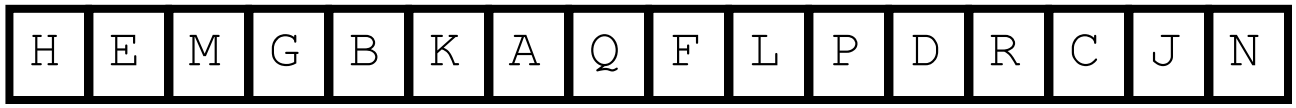
Since merging is easier than sorting, if I have two helpers, I'd…

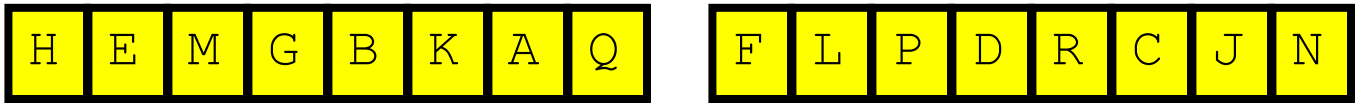
- Give each helper half the array to sort
- Then I get back their sorted subarrays and merge them.

What if those two helpers each had two sub-helpers?

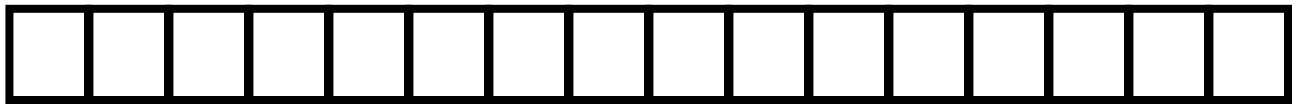
And the sub-helpers each had two sub-sub-helpers? And…

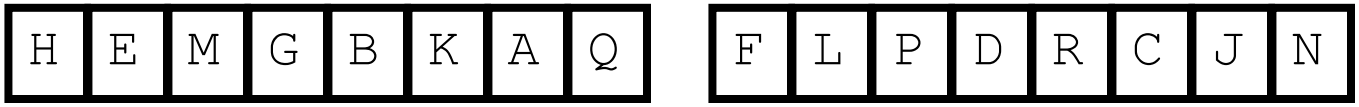
Subdivide the sorting task

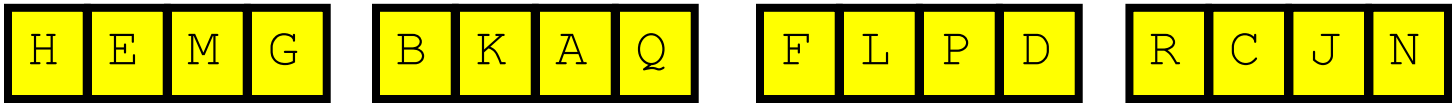




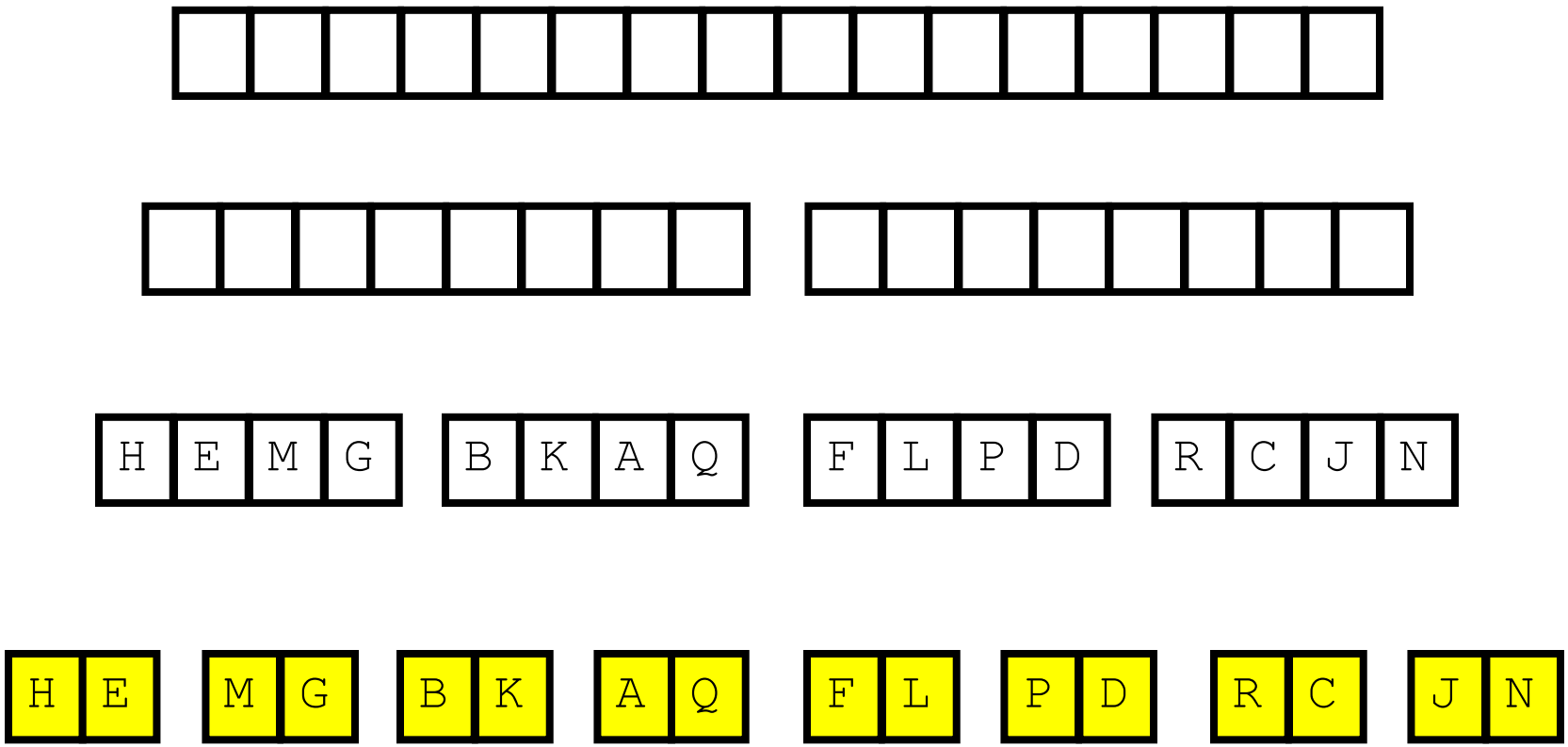
Subdivide again



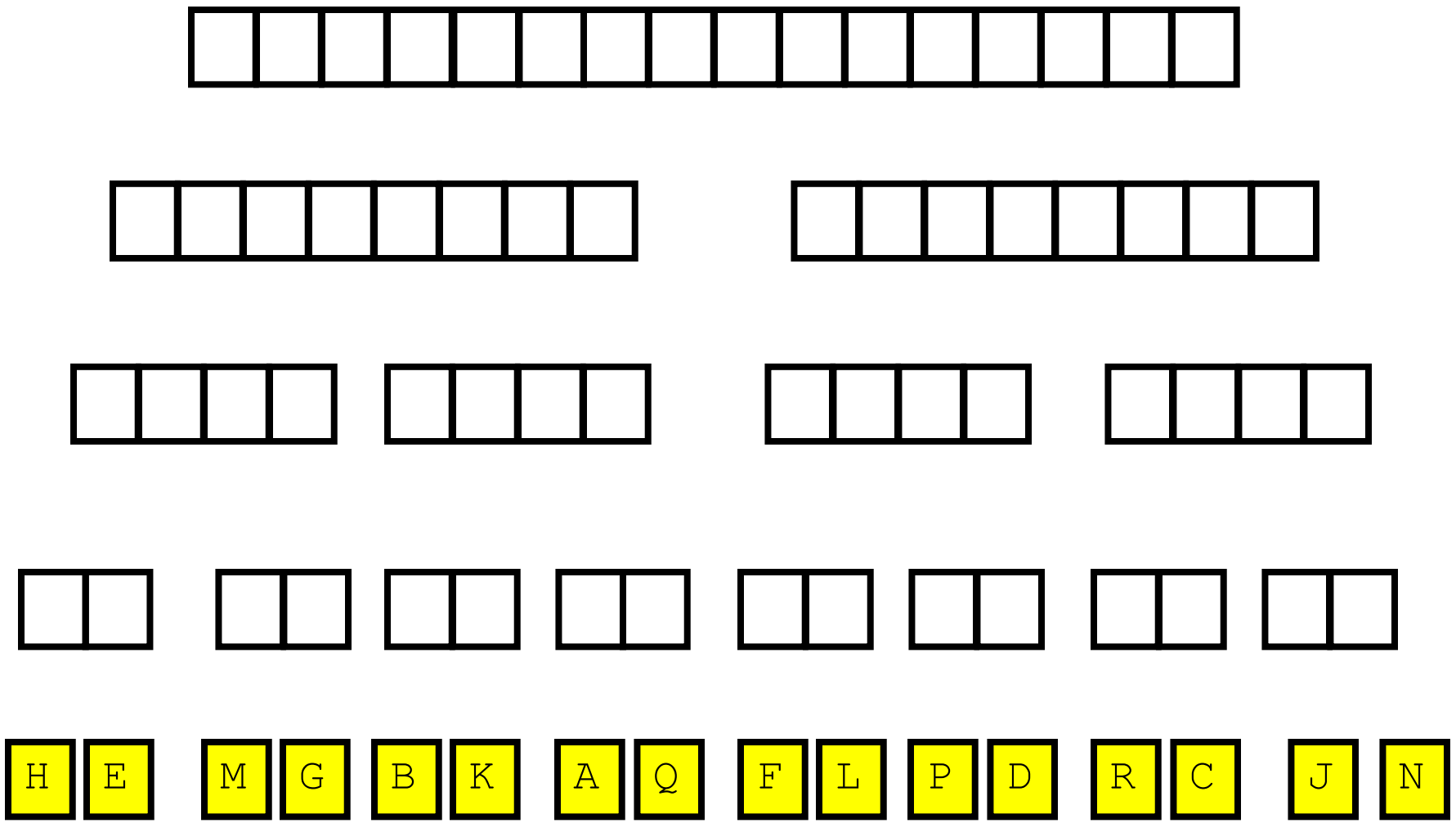




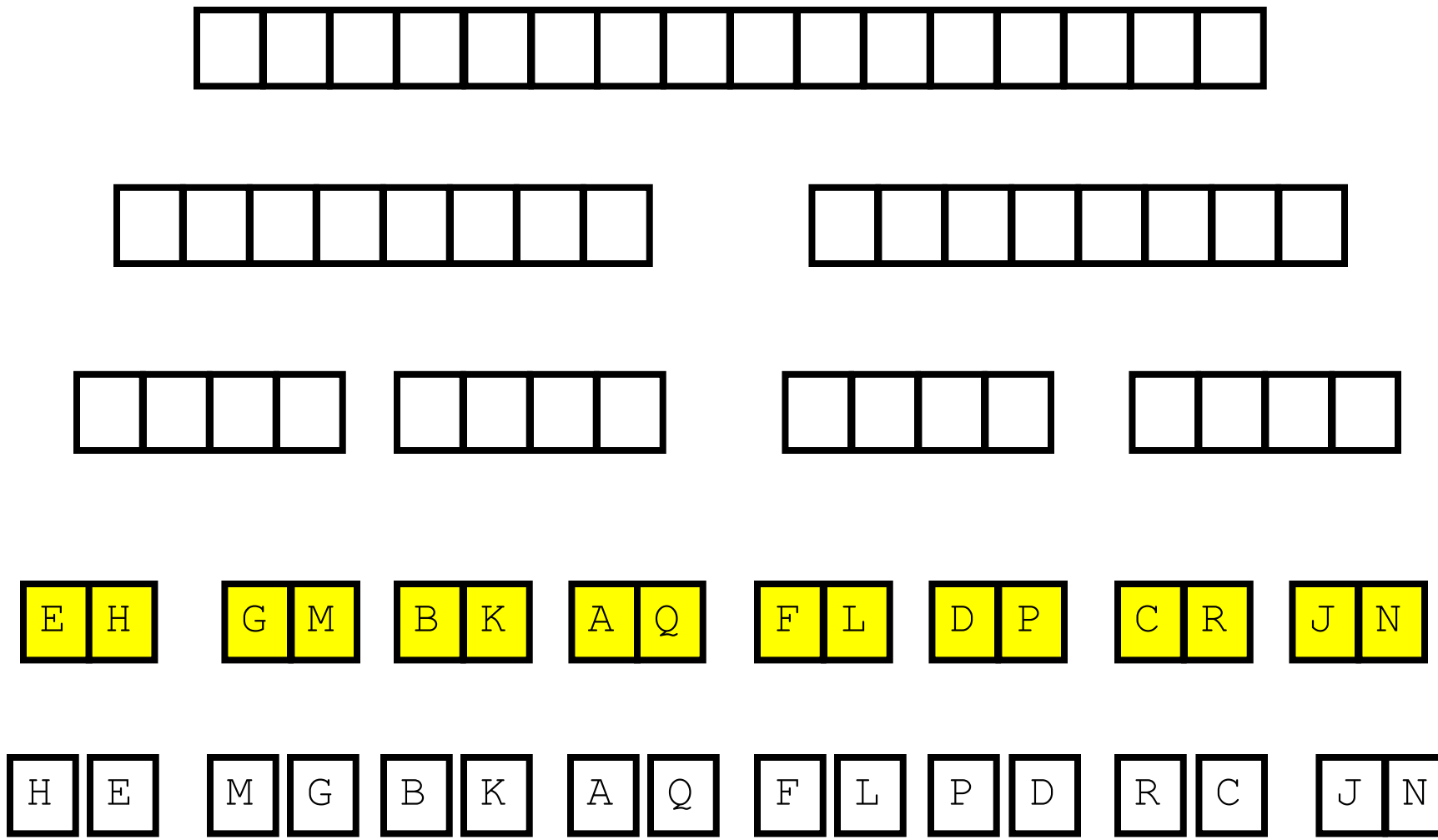
And again



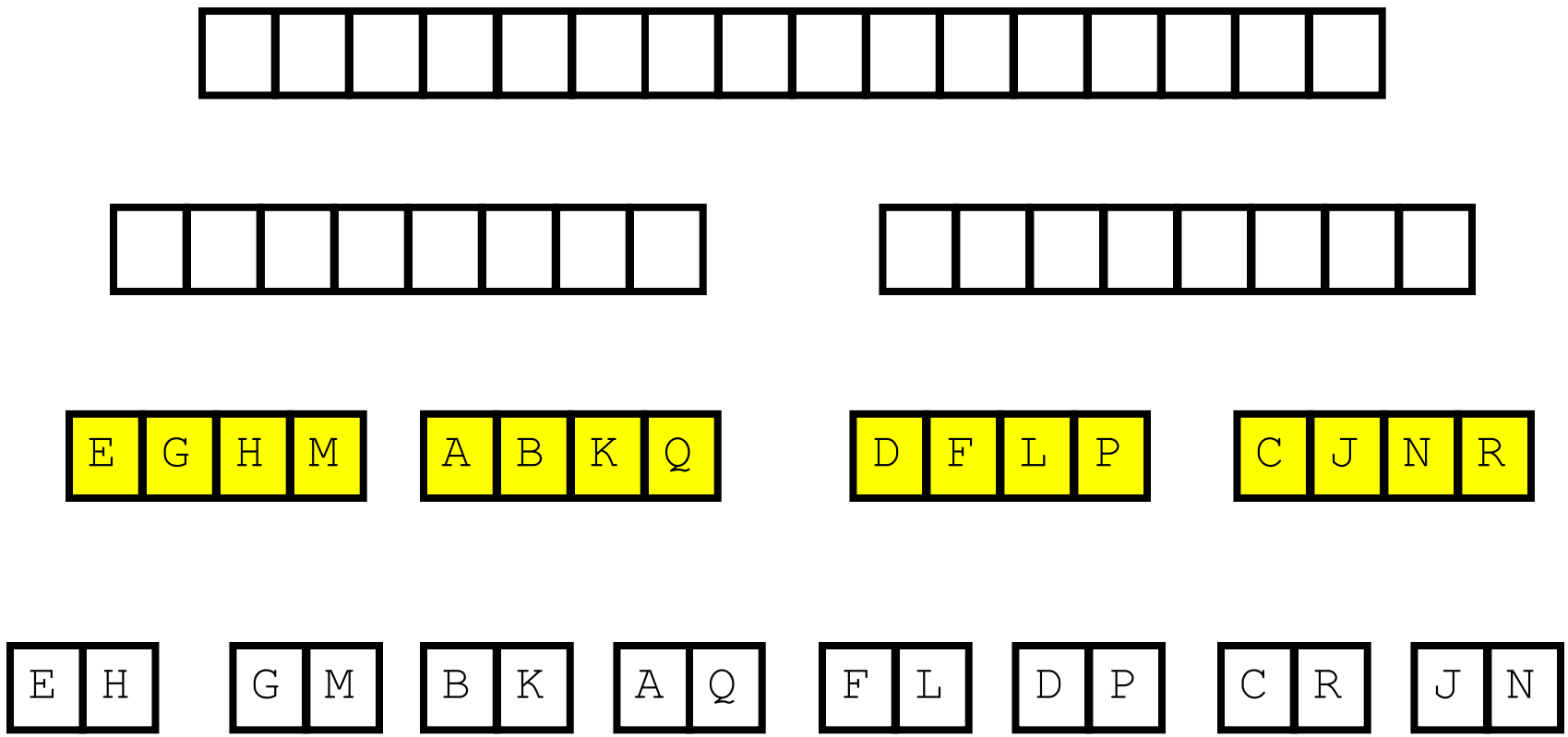
And one last time



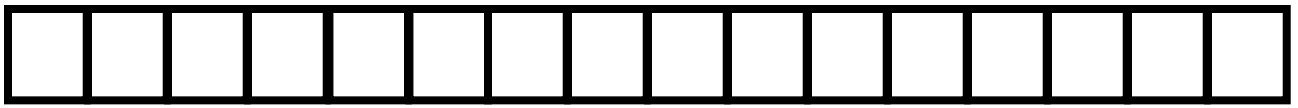
Now merge

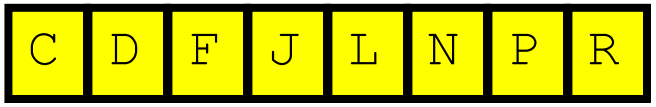


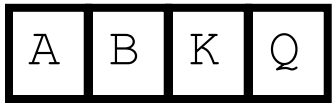
And merge again

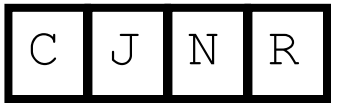


And again

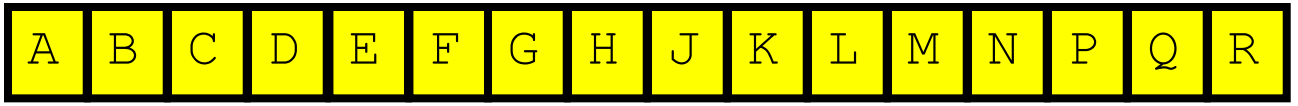


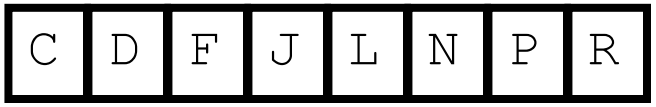




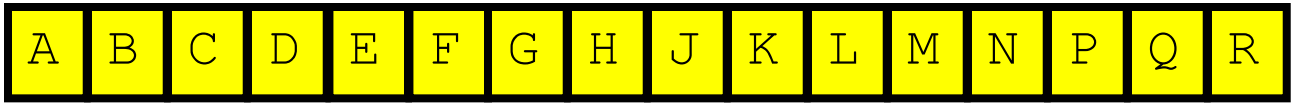


And one last time





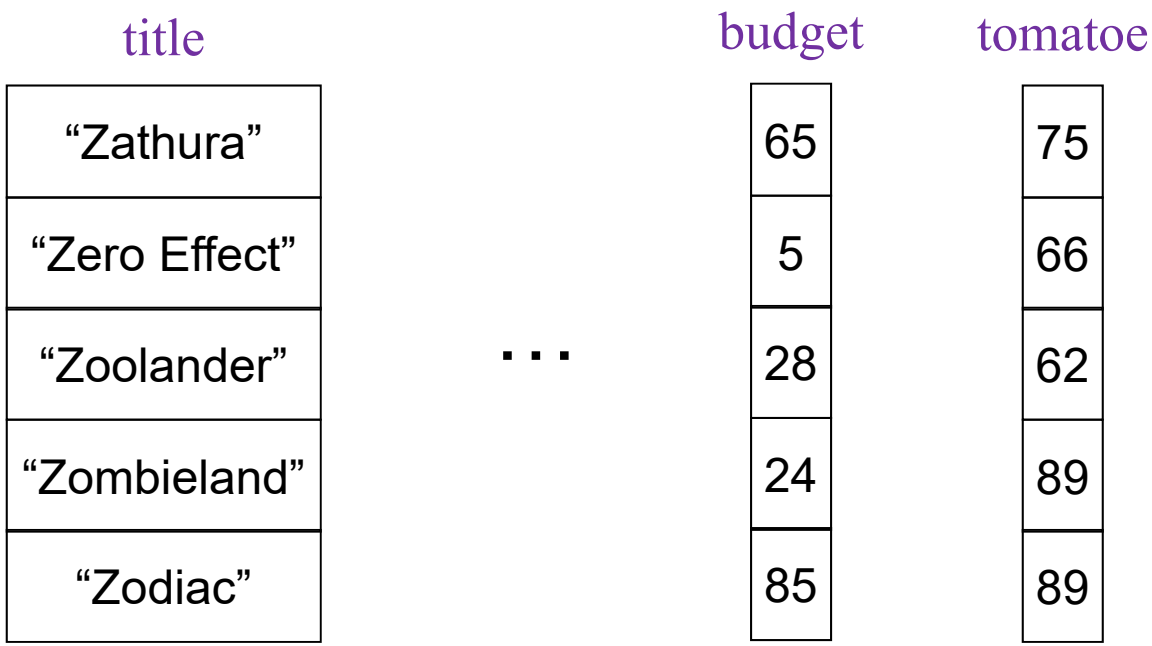
Done!



See coding demo on video

Remember that our movie data set has many columns…

- Shouldn't just sort one list (e.g., list of budget)
- Need to maintain correlation with the other columns



Can define a custom class for our data

Can sort **mList** according to specific attribute, e.g., budget, tomatoe, …. Generally make data easier to work with!