Sorting
Announcements

• Quiz 4 will be handed back Wednesday

• Lab 8 is posted; due Saturday at midnight
Today’s plan

• Continue with run time analysis practice
• Describe task of sorting
• Exercise: design a sorting algorithm
• Exercise: Implement a sorting algorithm
• https://www.cs.swarthmore.edu/~mauskop/cs21/s17/practice/9F.html
Sorting

• Take a list of elements of the same type, which can be ordered.

• Rearrange the elements of the list so that all the original elements are there, but now in non-decreasing (or sorted) order.

• For now, we want to do this in-place, that is, without needing to use an extra list.
Sorting applications

• Prepare a list for binary search
• Present information to a user in sorted order
• Gather stats like mode, median
• Do an equality check for two sets
• And many more…
Exercise: design sorting algorithm
In-place sorting

• Do the sort without using more than $O(1)$ extra memory.

• Relies on the idea of swapping the values at two indices in a list.
Buggy swap

def buggySwap(L, i, j):
    L[i] = L[j]
    L[j] = L[i]
def swap(L, i, j):
    # Purpose: swaps the values at i and j in list L
    # Parameters: L - a list
    #             i, j - valid indices for L
    # Returns: nothing, but mutates L
    temp = L[i]
    L[i] = L[j]
    L[j] = temp
Aside: shuffling

• Once we know how to swap we can re-implement the shuffle function

```python
def shuffle(L):
    """
    Purpose: randomly shuffles the contents of list L
    Parameters: L - a list
    Returns: nothing, but mutates L
    """

    n = len(L)
    for i in range(0, n-1):
        j = randrange(i, n)
        swap(L, i, j)
```
Exercise: implement sorting algorithm