Sorting Algorithms Implemented
Announcements

• Lab 8 due Saturday at midnight
  - Ninja sessions tonight and Friday night
Today’s Plan

• Review quiz 4

• Review Monday’s lecture

• Implement a sorting algorithm

• Go over selection sort code

• Go over bubble sort code

• Analysis
Quiz 4 Review

• Q1: Value being mutated only appears once, possibly with multiple arrows pointing to it

• Q2: Need to use `for i in range(len(L))` loop

• Q3: `response = raw_input(question)`

• Q4: `append` method takes one argument, mutates, doesn’t return anything
Review

• An in-place sorting algorithm uses $O(1)$ extra space beyond the list being sorted. Such an algorithm can’t create a new list, it must mutate the list passed in.

• These algorithms proceed by orchestrating a series of swaps that result in the list being sorted.
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
Implement a sort in w10-sorting/sorts.py
Selection sort

For each valid position, $i$, in the list:

1. Find the index of the smallest value in the sublist from $i$ to the end of the list. Call this index $\text{min}$. 

2. Swap the values at $i$ and $\text{min}$. 
Selection sort

def selectionSort(L):

Purpose: sorts the given list in place
Parameters: L – a list of items that can be ordered
Returns: nothing, but mutates L

n = len(L)
for i in range(n):
    min = i
    for j in range(i+1, n):
        if L[j] < L[min]:
            min = j
    swap(L, i, min)
Bubble sort

Do the following $n$ times to sort a list, $L$, of length $n$:

- Examine each pair of consecutive values in $L$. If the left value is bigger than the right value, swap them.
def bubbleSort(L):
    n = len(L)
    for i in range(n):
        for j in range(n-1):
            if L[j] > L[j+1]:
                swap(L, j, j+1)
Bubble sort

Repeat the following until an entire pass of $L$ yields no swaps:

- Examine each pair of consecutive values in $L$. If the left value is bigger than the right value, swap them.
def bubbleSort(L):
    n = len(L)
    made_swap = True
    while made_swap:
        made_swap = False
        for j in range(n-1):
            if L[j] > L[j+1]:
                swap(L, j, j+1)
                made_swap = True
Analysis

• Selection sort: $O(n^2)$

• Bubble sort: $O(n^2)$

• We can make minor optimizations, but the big $O$ run time won’t change.
Visualizations

• https://visualgo.net-sorting
• https://www.cs.swarthmore.edu/~knerr/teaching/topics/sort.html
• https://www.youtube.com/watch?v=lyZQPjUT5B4#t=0m48s