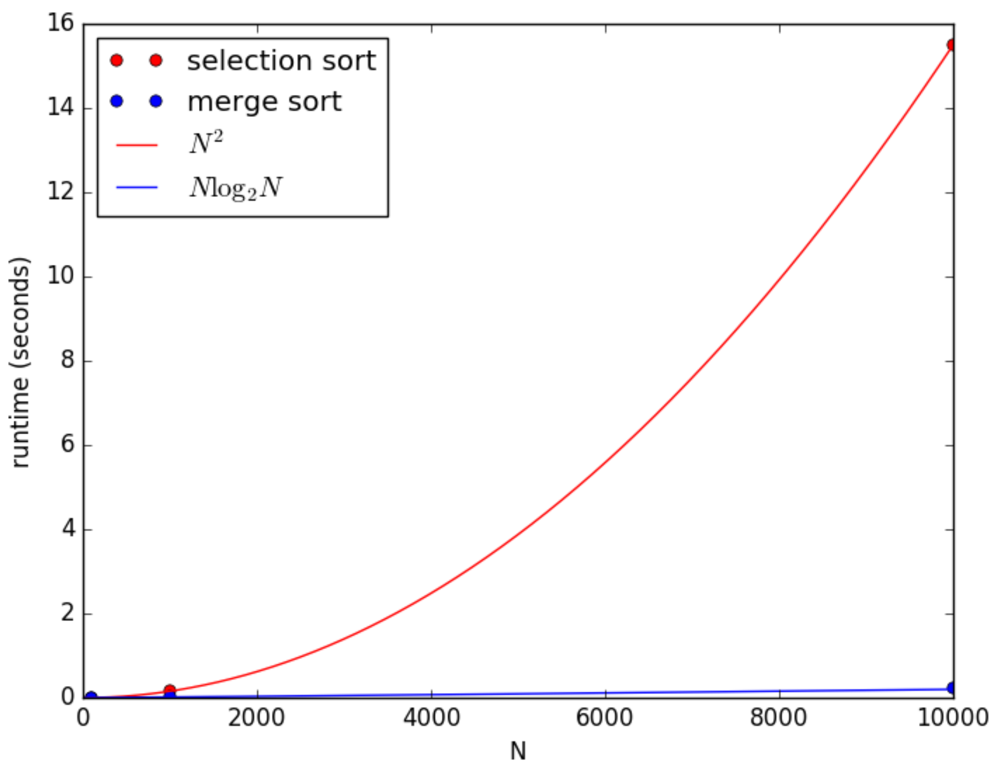
Applications in Scientific Computing
Assignment 3: Sorting, searching, and FFTs

530.390.13

Due: Tuesday 12 January 2016

Submit all code by committing it to the directory `assignments/assignment3` in your `530.390.13` GitHub repository. For a reminder of how to use Git, refer to the repository file `notes/using-git`.

1. Compare the run times for the selection sort and merge sort algorithms using the same randomly generated array for each algorithm. Plot the run times for arrays of various size N. Do these data match the performance expectations of $O(N^2)$ for selection sort and $O(N \log N)$ for merge sort? Compare the ratio of the run times for each algorithm for $N = \{100, 1000, 10000\}$.

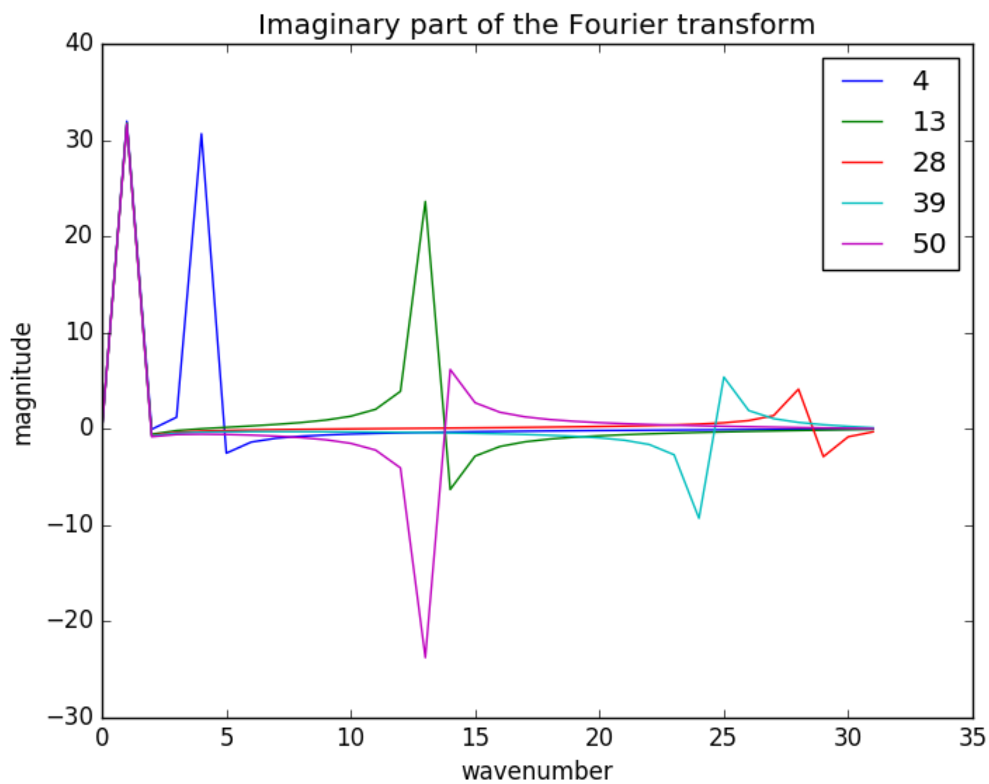


   - See the code in `sorting.py`.
   - As plotted, the data match the expected scaling.
   - Ratios (selection sort / merge sort): {0.96, 8.3, 70}

2. Write a recursive algorithm for calculating the n^{th} Fibonacci number, F_n. Recall that $F_0 = 0$ and $F_1 = 1$. What is F_{24}?

- See the code in `fib.py`.
- $F_{24} = 46\,368$

3. Consider the function $f_b = \sin(x) + \sin(bx)$ for $b = \{4, 13, 28, 39, 50\}$. Discretize each of these functions on a grid with $N = 64$ and apply the Fourier transform to each and plot the results. What changes in each of the respective spectra? Note carefully the behavior when $b > N/2$. This effect, called *aliasing*, means that waves in a signal with wavenumber higher than $b > N/2$ (related to the *Nyquist frequency*) are under-resolved and appear in the signal as *lower* wavenumbers.



- See the code in `aliasing.py` and `fft.py`.
- The peak at b first moves to the right, and then back to the left after $b > N/2$.