1 Setup

Copy the files from /courses/cs1114/section/search/ into a working directory in your home directory. If you have your quicksort implementation handy, copy quicksort.m and partition_array.m into the working directory as well. If not, you can borrow my quicksort implementation, which can be found in /courses/cs1114/section/search/qsort/.

Finally, open Matlab, navigate to your working directory, and load up the list we'll be searching:

>> load list.mat

2 Linear Search

1. Implement a $O(n)$ algorithm to find the index (position) of a value (key) in the list. Return 0 if the key is not in the list. Write your search algorithm in linear_search.m. The function header is:

function [index] = linear_search(A, key)
% Perform a linear search on A for key. Return the index of key in A
% or 0 if key does not appear in A.

3 Binary Search

2. I’m thinking of a number between 0 and 1,000,000. Each time you guess a number, I’ll tell you if you’re correct, too high, or too low. What is your guessing strategy?

3. Implement binary search in binary_search.m. The function header is:

function [index] = binary_search(A, key, low, high)
% Perform a binary search on A for key. Return the index of key in A
% or 0 if key does not appear in A. If A contains multiple copies of
% key, we can return the index of any one of them.
%
% A must be a pre-sorted list.

4. What is the asymptotic runtime of binary search?

5. Run the search_test, which will test and time your search methods (it also calls quicksort to sort the list). This function takes a list and a number of searches to perform; for example:
You should find that binary search is faster, but not if you consider the time it takes to sort the list. However, if you need to do many searches, sorting the list may become worthwhile. How many searches does it take to make sort+binary search faster on average than linear search?

6. If you have time: is your solution iterative or recursive? If it’s iterative write a recursive solution, and vice versa.

4 Searching Linked Lists

7. What’s the most efficient algorithm for searching an arbitrary linked list?

8. What if the list is sorted?

9. Can you think of something a little different that still uses links and nodes (so that it has sublinear insertion and deletion) but is efficiently searchable? Think about how you could design a data structure on which you could perform something like a binary search.