# Section 07: Sorting, Memory, and a Dash of Graphs

## 1. Sorting: Mystery

Consider the following sorting algorithm in pseudocode. Note that, in this case, the upper bound of each for loop is *inclusive*, so they run up to and including $i = A$.length $- 1$ and $j = i - 1$.

```
function MysterySort(A)
    for i = 1 to A.length − 1 do
        for j = 0 to i − 1 do
            if A[j] ≥ A[i] then
                x = A[i]
                shift every item from j to i − 1 right by one
                A[j] = x
                break
```

(a) Is MysterySort most similar to insertion sort, merge sort, quick sort, or selection sort?

(b) Is MysterySort a stable sorting algorithm? Why or why not?

(c) What is the best-case runtime (as a tight big-$\mathcal{O}$ bound) for MysterySort? Why is this the best case?

**Hint:** What happens when MysterySort is given an array that is already sorted?

## 2. Sorting: Design Decisions

For each of the following scenarios, say which sorting algorithm you think you would use and why. There may be more than one right answer.

(a) Suppose we have an array where we expect the majority of elements to be sorted "almost in order". What would be a good sorting algorithm to use?

(b) You are writing code to run on the next Mars rover to sort the data gathered each night. (Think about sorting with limited memory and computational power.)

(c) You're writing the backend for the website `SortMyNumbers.com`, which sorts numbers given by users.

(d) Your artist friend says for a piece she wants to make a computer sort every possible ordering of the numbers $1, 2, \ldots, 15$. Your friend says something special will happen after the last ordering is sorted, and you'd like to see that ASAP.

## 3. Sorting: Algorithm Practice

(a) Demonstrate how you would use quick sort to sort the following array of integers. Use the first index as the pivot; show each partition and swap.

[6, 3, 2, 5, 1, 7, 4, 0]

(b) Show how you would use merge sort to sort the same array of integers.

## 4. Memory: Short Answer

(a) What are the two types of memory locality?

(b) Does this more benefit arrays or linked lists?

## 5. Memory: In Context

(a) Based on your understanding of how computers access and store memory, why might it be faster to access all the elements of an array-based queue than to access all the elements of a linked-list-based queue?

(b) Why might `f2` be faster than `f1`?

```
public void f1(String[] strings) {
    for (int i=0; i < strings.length; i++) {
        strings[i] = strings[i].trim();        // omits trailing/leading whitespace
    }
    for (int i=0; i < strings.length; i++) {
        strings[i] = strings[i].toUpperCase();
    }
}

public void f2(String[] strings) {
    for (int i=0; i < strings.length; i++) {
        strings[i] = strings[i].trim(); // omits trailing/leading whitespace
        strings[i] = strings[i].toUppercase();
    }
}
```

(c) Consider the following code:

```
public static int sum(IList<Integer> list) {
    int output = 0;
    for (int i = 0; i < 128; i++) {
        // Reminder: foreach loops in Java use the iterator behind-the-scenes
        for (int item : list) {
            output += item;
        }
    }
    return output;
}
```

You try running this method twice: the first time, you pass in an array list, and the second time you pass in a linked list. Both lists are of the same length and contain the exact same values.

You discover that calling `sum` on the array list is consistently 4 to 5 times faster then calling it on the linked list. Why do you suppose that is?

(d) Suppose you are writing a program that iterates over an `AvlTreeDictionary` – a dictionary based on an AVL tree. Out of curiosity, you try replacing it with a `SortedArrayDictionary`. You expect this to make no difference since iterating over either dictionary using their iterator takes worst-case $\Theta(n)$ time.

To your surprise, iterating over `SortedArrayDictionary` is consistently almost 10 times faster!

Based on your understanding of how computers organize and access memory, why do you suppose that is? Be sure to be descriptive.

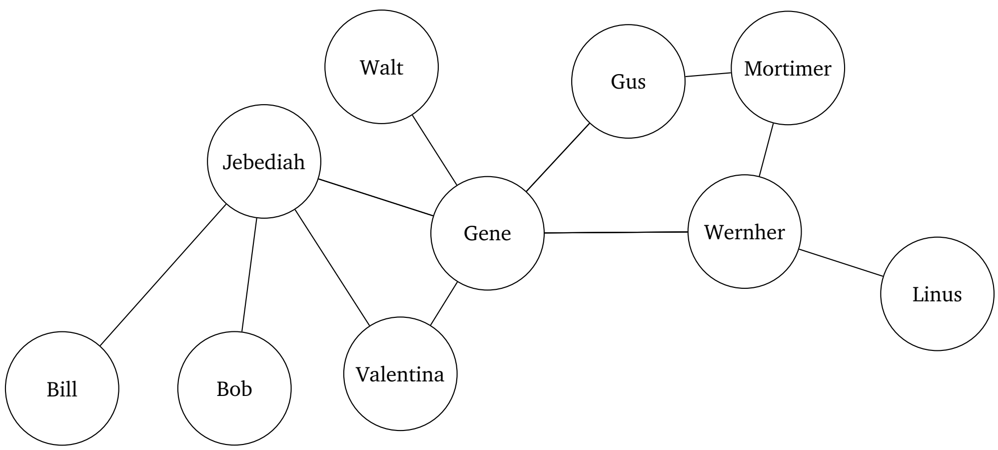
(e) Excited by your success, you next try comparing the performance of the `get(...)` method. You expected to see the same speedup, but to your surprise, both dictionaries' `get(...)` methods seem to consistently perform about the same.

Based on your understanding of how computers organize and access memory, why do you suppose that is?

(Note: assume that the `SortedArrayDictionary`'s `get(...)` method is implemented using binary search.)

## 6. An Introduction to Graphs

Consider the following graph that models friendships between people on Facebook.




(a) What do the vertices in this graph represent? What do the edges represent?

(b) Is this a directed or undirected graph? Why does this kind of graph make sense for Facebook friends?

(c) Which vertex has the highest degree? What degree does it have?

(d) Gus has an urgent message to send to Bob. What is the shortest path he can use to send his message?

(e) Using your data structures from the homework projects, create this graph as an adjacency list in Java.

**Hint:** The type of the graph should be `IDictionary<String, ISet<String>>`. For convenience, you can assume `ChainedHashSet` has a constructor that can initialize the set with a list of arguments. For example, `new ChainedHashSet<String>("Bill", "Bob", "Valentina")` would create the set $\{\text{Bill}, \text{Bob}, \text{Valentina}\}$.

(f) Write a method `String maxDegree(IDictionary<String, ISet<String>> graph)` that returns the vertex with the highest degree.