

# CS1020: DATA STRUCTURES AND ALGORITHMS I

## Tutorial 10 – Sorting and Searching

(Week 12, starting 4 April 2016)

### 1. High-Tech Library

A new high-tech library is opening near your home – And you have been tasked to help design a system that facilitates searching of titles. Being a smart CS1020 student, you know that you should sort the books in ascending order, so that they can be found quickly using Binary Search.

- (a) The librarian Mary suggests using Radix Sort to sort the books by their title. She argues that Radix sort is the best candidate because it is an  $O(n)$  algorithm. Do you agree? What are some potential problems of using this algorithm?
- (b) Suggest a modification to Mary's idea to make using Radix Sort feasible. Discuss any possible drawbacks in your solution.
- (c) After much brainstorming, you've narrowed down your choices of sorting algorithms to Quicksort and Merge Sort. Which would you pick, and why?

(Hint: Remember while answering this question note that speed is not the only concern!)

### 2. Taking Advantage of Java Sorting

In practical applications, we do not usually have to code our own sorting algorithms. Instead, we can simply make use of what is provided by Java, namely the `Collections.sort()` function. What's really interesting about this function is that it can sort a list of any generic type! In this question, we explore how and why that works.

To do this, we have a Coordinate Class, which holds x, y and z coordinates of a point in space. We want to be able to sort a list of Coordinates by their distance from the origin (0,0,0). The following is a simplified skeleton of the coordinate class:

```
public class Coordinate {
    private double _x;
    private double _y;
    private double _z;

    public Coordinate(double x, double y, double z) {
        _x = x;
        _y = y;
        _z = z;
    }

    public double getX() { return _x; }
    public double getY() { return _y; }
    public double getZ() { return _z; }
}
```

In order for a list of Coordinates to be sorted the coordinates need to be **Comparable** to each other. Look into the Java API for the **Comparable** class, and implement what is required to make the Coordinate class sortable!

### 3. Search Algorithms

After learning about Binary Search in CS1020, you decide to build a better version of the algorithm, which partitions the array into three parts instead of just two.

- (a) In English, explain what you would need to change about Binary Search to make it partition a list into three instead of two. Then, modify the following code snippet to implement your idea.

```
public static int binSearchRecur(int[] input, int searchTerm,
                                int low, int high)
    throws Exception {
    if (low > high) {
        throw new Exception("Item Not Found");
    }

    int mid = (low + high) / 2;
    if (searchTerm > input[mid]) {
        return binSearchRecur(input, searchTerm, mid+1, high);
    }
    else if (searchTerm < input[mid]) {
        return binSearchRecur(input, searchTerm, low, mid-1);
    }
    else {
        return mid;
    }
}
```

- (b) Since your algorithm is able to discard more information per iteration, it appears to be more efficient than Binary Search. Discuss whether this is the case using the big-O notation.
- (c) As we have now seen, Binary Search is fast and thus preferable over linear search. However, this is not always the case. Give an example of such a context, and support it with discussions on time complexity.