

Introduction to Data Structures

Overview

The goal of this project is to learn more about what a data structure is and how we choose between different data structures for different tasks.

The choice of which data structure to use is usually based on the following two main factors:

- Whether the data structure will perform well for the given task.
- Whether the data structure will be easy to use and implement in the code.

Note that there is a trade-off here. A fancy data structure might be very fast but might take too long to code or be difficult to understand. However, in general that won't be an issue for us right now so we are going to concentrate on the performance of data structures.

“Big O” Notation

Among computer scientists, “Big O” notation is used to talk about the performance of data structures and the algorithms that are used to work with them. Big O notation is a summary of the number of steps necessary to do a certain task. It is similar to limits in precalculus/calculus.

In Big O notation, the variable **n** is used to denote the number of items in a data structure. Imagine an ArrayList with **n** items sorted in alphabetical order.

Consider the amount of steps it takes to do the following:

Find out the number of items n	$O(1)$ (a.k.a., constant time)
Look up the first item in the list	$O(1)$
See if some String s is in the list	$O(n)$
Add a new item to the list	$O(n)$

By “constant time,” we mean that the amount of work to do that task does not depend on the size of the list. However, looking up a String or adding a new String takes more work if there are more items in the list.

Now imagine that you have an ArrayList of Strings but they aren't sorted and you need to sort them. In this case, the performance will depend on what kind of sorting technique you use:

Sorting using a simple sort algorithm $O(n^2)$

Sorting using a complex sort algorithm $O(n \log n)$

Here you can see a simple example of the trade-off between the complexity of the code and the performance of the code. (This is not to say that all complex code is necessarily better.)

The Project

This project will be divided into two phases:

Phase 1

For this phase, you should research your assigned data structure and write a short (i.e., one page or so) analysis of the data structure. Your analysis should include the following information:

- Name of the data structure and its basic characteristics
- The performance of the data structure for at least the following operations (insert, delete, count number of entries, list entries in sorted order). You can add other operations if you choose.
- What tasks for which the data structure is or is not well-suited.
- Common variants of the data structure.

Phase 2

For this phase you should implement your data structure in Java. Your code should have at least two classes: the data structure class itself, and a class that contains the main() and which uses the data structure for some simple task and tests all its features.

Good style and comments are an important part of this coding project, because one goal of this work is to provide sample code that the Intermediate students can use (with help) later in the course.

Nuts and Bolts

Phase 1 Due Fri., Sept. 15th at the end of class 10 points

Phase 2 Due Wed., Sept. 25th at the end of class 15 points

Extra Credit: You can implement a variant of your data structure (e.g., a double-ended queue—known as a deque—or a balanced binary tree) for up to 3 points of extra credit.