Driller
Software Design Document

Version <1.0>
Trent (Will) Hughes
OU-ID: 113392078
2019/10/03

Introduction
The purpose of this document is to give a short explanation on design decisions for the Driller 1.0 command line application.

Design Overview
Unit Testing
The first thing I did was setup Unit Testing in Xcode.
I wrote unit tests for every method in RezizableArray, DrillingRecordComparator, Search, Sort. In that order

DrillingRecordComparator
This was the first I completed because the Search, Sort classes would use the comparator to determine if a < b, a > b, or a == b

Read-In
We first grab the FIRST_DATESTAMP from the file to compare all consecutive datestamps to.
For each row after we check this comparison. Stop if invalid or continue
Then we grab the timestamp. We do not check if this is unique YET.
We first validate the numbers. If they are valid, we do a binary/linear search on the timestamp.

Search
Linear
Linear search is used only when the array is unsorted.
Time complexity of O(n)
Space complexity of O(1)
Modifications: Utilizes comparator to determine if search token equals item at index of for loop

Binary
This is the preferred searching method. Used when array is sorted.
Time complexity of O(log n)
Space complexity of O(1)

Modifications: Utilizes comparator. The middle else if statement checks if index mid > 0 and if item at mid-1 > searchKey OR item at mid-1 < searchKey. This modification allows us to find the first key instead of finding any key when there are duplicates found. Example {1,2,3,4,5,5,5} will result in index 4, not 5 or 6. Since index 4 is the first 5 in the array.

Total Search Time

Best case binary search time complexity of O(log N)
Worst case linear search time complexity of O(N)

Space complexity for search remains the same despite the two search algorithms O(1)

Sort

Quicksort

I decided quicksort would be the best algorithm because despite its worst time complexity of 0(n^2) the best case is O(nlog n). The worst time complexity only occurs when the array is already sorted. Which we have logic to safeguard against. The average case is O(nlogn). Quicksort does not require any extra storage, but Merge Sort requires O(n) extra storage. The extra time taken to allocate and deallocate increases the running time of the algorithm. Therefore, I decided that merge sort would not be a good option for our array.

isSorted Method

Used in Driller to decide whether or not to run linear search or binary search

Merge of Multiple Files

Initial Read-In

I decided to do data validation and merge into recordArray as separated concerns. This decision is based off the idea that a user's experience would be best considered if a sorting algorithm only had to run once after they have imported multiple files.

If we were to run the quick sort algorithm after every file is selected then the user would have to wait each time to enter another file. When it should only run once the user is finished importing files.

I allow the user to import all files.
Then, if they go directly to output. Then at this point, I sort it by timestamp.
If the user goes to sort. Then they can select which column to sort. Output will reflect their decision.

Best Time Complexity: O(N logN)
Space Complexity: O(1)