CS 260 100P
Searching

Revision Date: January 10, 2013

Printable version

Tasks to perform

To your dynamic circular array class, add two search methods. The first method performs a linear search, while the second method performs a binary search.

For the linear and binary search methods, the worst case behavior should occur when one looks for a value not in the array. Create two graphs that plot the input size versus the time it takes to determine a value is not in the array. One graph will contain data from linear searches while the other will contain data from binary searches. Please read up on plotting these kinds of plots here.

Your main program should take the following command-line arguments:

- a flag that specifies whether a linear or binary search should be used
- a time value
- a list of file names containing data

The output should be a *gnuplot* file. The time value is used as the upper limit on the *y*-axis of the generated plot (so that the two plots can be compared). Use your dynamic array class to read in the data from the files given as command-line arguments.

The main can be written as a group and the code shared.

Generating test data

To test your sorts, write a program (class project) to generate sorted and unsorted data. Your program should take the following command line arguments:

```
python3 makeIntegers <count> <start> <step> <swaps>
```

The first command-line argument, `<count>`, refers to the number of integers to generate. The second, `<start>`, refers to the starting number (subsequent numbers increase from this starting number). The third, `<step>`, refers to the spacing between the generated integers.The last argument, `<swaps>`, refers to the number of random swaps that should be performed to permute the set of integers. For example, the call:

```
makeIntegers 10 3 2 0
```

should generate the numbers:

```
3 5 7 9 11 13 15 17 19 21
```

where as:

```
makeIntegers 10 3 2 1
```

might produce the output:

```
3 5 11 9 7 13 15 17 19 21
```

A large value for swaps will essentially randomize the output. The generated numbers should be written to stdout; they can be captured in a file via redirection:

```
makeIntegers 10 3 2 1 > data.out
```

You may share the *makeIntegers* code.

Other requirements

Be able to describe to your mentor the complexity of each operation in the public interface of your classes using order notation.