Big Oh - Analyzing Scalability of Algorithms

Nada Basit and Mark Floryan

February 11, 2022

1 SUMMARY

For this homework, you will use the methods you wrote in the pre-lab to understand the scalability of algorithms, and the differences between common runtimes.

1. Reacquaint yourself with the five methods (and one provided method) from the Big-OH implementation homework.

2. Run each method on increasing array sizes. Use completion times to fill out the BigOh chart.

3. **FILES TO DOWNLOAD:** None (*Use your code from the previous assignment*)

4. **FILES TO SUBMIT:** BigOh.pdf

1.1 GRAB YOUR CODE FROM PREVIOUS ASSIGNMENT

In the previous assignment, you implemented five methods (one one was provided to you). The headers for these methods are provided again here for your convenience.

```
1 // Searches for item in sorted array a.
  // Returns true iff item is found in a
```

```
3  // Should run in Theta(logn) time
   public static binarySearch(int[] a, int item);
5
   // Finds the largest item in a (a might not be sorted)
7  // Should run in Theta(n) time
   public static int max(int[] a);
9
   // Invokes binarySearch() with a.length random numbers
11 // Returns how many times those random numbers were found in a
   // Should run in Theta(nlogn) time
13 public static int multipleBinarySearch(int[] a);

15 // Counts how many pairs of items in a sum to a multiple of 5
   // Should run in Theta(n^2) time
17 public static int allPairs(int[] a);

19 // Counts how many a,b,c combinations there are in a such that a+b=c
   // Should run in Theta(n^3) time
21 public static int allTriads(int[] a);

23 // loops through all the subsets of array a
   // e..g, {1,2,3} would print {},{1},{2},{3},{1,2},{1,3},{2,3},{1,2,3}
25 // Should run in Theta(2^n) time
   public static int allSubsets(int[] a);
```

1.2 Analyzing Scalability

Your task is to analyze how large inputs can reasonable get at each runtime. We have provided a main function to you that asks for two inputs: The method you would like to execute, and how large you'd like the input size (array) to be. The method also provides some code that times how long the operation takes and reports the time to you.

Run the code multiple times and fill out the chart below. Submit the chart in a pdf file (BigOh.pdf). As you are filling out this chart, think about how much larger the input can get as your code becomes more and more efficient. If a time takes more than 1 minute, don't wait for the code to finish, just report that the code took more than 60000 ms. Please report all times in ms (a time of 0 ms is fine if the code finishes quickly).

Input Size:	1	10	100	1,000	10,000	10^5	10^6	10^7	10^8
Binary Search (logn)									
Max (n)									
Multi Binary Search (nlogn)									
All Pairs (n^2)									
All Triads (n^3)									
All Subsets (2^n)									

You should submit one file for this homework: **BigOh.pdf**.

1.3 Analysis

In your report, write a couple of paragraphs about the results you see. Do they match what you expected? How large can inputs get before certain methods start to slow down to an unreasonable pace? Does anything about the times you recorded stick out as strange?

1.4 What if I didn't finish the previous homework?

For this homework, we do not care if your methods are returning the **correct answer**, but we DO care that they have the correct asymptotic complexity. So, you may have to do some work to at least ensure the latter. Make sure each method is doing the correct amount of work, even if it has a bug or small error in it. Then, do the analysis without worrying about the correctness of the output.

1.5 Gradescope

You should submit your pdf to *Gradescope*. There is no autograder for this assignment, and a grader will be manually checking your submission.