SIENAcollege
Computer Science

Computer Science 385
Design and Analysis of Algorithms
Siena College
Spring 2025

Problem Set 1
Group Formation: 4:00 PM, Friday, January 31, 2025
Due: 4:00 PM, Wednesday, February 5, 2025

You may work alone or in a group of size 2 or 3 on this assignment. However, in order to make sure you learn the material and are well-prepared for the exams, you should work through the problems on your own before discussing them with your partner(s), should you choose to work with others. In particular, the "you do these and I'll do these" approach is sure to leave you unprepared for the exams.

There is a substantial amount of work to be done here, and you are sure to have questions. It will be difficult if not impossible to complete the assignment if you wait until the last minute. A slow and steady approach will be much more effective.

Submitting

Please submit a hard copy (typeset preferred, handwritten OK but must be legible) for all written questions. Only one submission per group is needed.

Written Questions: Discrete Math and Data Structures

Question 1: Indicate and briefly justify (in words, not mathematically) the "Big O" complexity for each of the operations below. Differentiate among best, average, and worst cases and specify under what circumstances they occur, where relevant. Specify the basic operation you are counting in each case. You may use any trustworthy resource, but any such resource must be cited. Be sure you understand any answers you need to look up, as you will see some of these or very similar questions as quiz or exam questions soon. (16 points)

For example, if the operation is "Find the largest value in an unsorted array of n integers", your response could be:

The basic operation is an integer comparison. Since the array is unsorted, there is no way to find the largest until you have compared with every one of the n values, so the best, average, and worst cases are all $O(n)$.

- Perform a binary search in a sorted array of n integers.
- Add an element to an `ArrayList` that contains n values, using the default `add` method (which would add at the end).
- Add an element to a singly-linked list that contains n values using the most efficient possible `add` method (which would add at the head).

- Add an element to a sorted `ArrayList` that contains n integers, and which has capacity available to store the new value.
- Sort an array of n integers using bubble sort (with the implementation we have seen in class and lab).
- Determine if a key exists in a binary search tree that contains n keys.
- Determine if a specific value is currently stored in a sorted array of n integers.
- And one that you might not have seen, but should be able to reason out: count the number of times a specific value occurs in a sorted array of n integers.

As we do more and more analysis of algorithms, we will encounter mathematical results in various forms, with logs, exponents, square roots, that represent familiar values expressed in unusual ways. It is helpful to be aware when different mathematical expressions represent the same value. For example, n and $\sqrt{n^2}$ are equivalent.

Question 2: For the expressions below, group them into sets of equivalent expressions. For logarithms, lg indicates base 2, and otherwise the base of the log will be indicated with a subscript. (14 points)

$\lg a + \lg b$	n^2	$\lg(2^{n+1})$	$8^{\lg n}$	$(2^n)^n$	$2^{\lg n}$
$n^{\lg 8}$	$\lg(2^n)$	n	3	2^{n+n}	2
$4^{\lg n}$	1	$\frac{\log_{10} n}{\log_{10} 2}$	$4(2^n)$	$\lg 2$	$\lg(2^n)$
$(2^n)^2$	2^{n+2}	n^3	4	2^{n+1}	$\lg(ab)$
$\log_{10} n$	n	$2^n 2^n$	$\lg n$	$\lg(a/b)$	$\lg a - \lg b$
4^n	$n+1$	2^{n+2}	$n \lg 2$	$n \lg n$	$\lg 16$
$2(2^n)$	$n^{\lg 16}$	2^{n+1}	$\lg n$	2^{2n}	

Written Questions: Summations and Counting

Question 3: Compute closed forms the following sums. Show your work. No credit if intermediate steps are not shown. Remember page 476 of your textbook! (16 points)

$$\sum_{i=1}^{8}(4n + 2i + 7)$$

$$\sum_{i=0}^{n-1}(4n + 2i + 7)$$

$$\sum_{i=0}^{n} 2^i \cdot n$$

$$\sum_{i=0}^{n} 2^{i+3}$$

For the next 5 questions, refer to this Java code fragment:

```
for ( i = 1; i <= n; i++ ) {
  for ( j = 1; j <= n; j++ ) {
    for ( k = 1; k <= j; k++ ) {
      System.out.println("Hello!");
      System.out.println("Hello!");
    }
  }
}
```

Question 4: How many times will this print `Hello!` when `n=2`? (1 point)

Question 5: How many times will this print `Hello!` when `n=3`? (1 point)

Question 6: Write an expression involving three summations that counts the number of times it prints `Hello!` in terms of n. There should be one summation for each loop. (3 points)

Question 7: Simplify your expression from the previous question to get a closed form. Show all work. (3 points)

Question 8: Substitute `n=2` and `n=3` in your expression (show your work for this, even if it seems trivial) to see if the answers match what you counted above. (1 point)

Written Questions on Graph Representations

For the questions below, consider the graph representations discussed in class for storing directed graphs.

Suppose that the amount of memory required by the adjacency matrix graph representation is exactly $\frac{|V|^2}{8}$ bytes[1], and that the exact amount of memory required by the adjacency list graph representation is exactly $32|V| + 32|E|$ bytes[2].

[1]This assumes each Boolean value in the array is stored as a single bit.

[2]This assumes 8 bytes for each reference in the `vertices[]` array, 24 bytes for each `Vertex` object (8 bytes for the `head` reference plus 16 bytes needed by Java to store housekeeping information about each `Vertex` object), and 32 bytes per `Edge` object (8 bytes for the integer `dest`, 8 bytes for the `next` reference, and 16 bytes to store Java housekeeping information about each `Edge` object).

Question 9: If you have a graph with 2^{21} (just over 2,000,000) vertices and each vertex has 4 outgoing edges, exactly how much memory in gigabytes is needed to store the graph using each representation? For each representation, can it fit into the 32GB of main memory which you might find in a PC today? (4 points)

Question 10: Graphs that have only a few number of edges per vertex are known as sparse graphs. A graph with a high number of edges per vertex is said to be dense. Suppose you have a complete directed graph of 2^{21} vertices. Here every vertex has a directed edge to all the other vertices. This is the "densest" graph there is! Exactly how much memory is needed to store the graph using each representation? Express your answer in gigabytes. (5 points)

Question 11: For a graph with 2^{21} vertices, where is the "break even" point, measured by $|E|$, below which the adjacency list representation is more memory efficient, and above which the adjacency matrix representation is more efficient? (5 points)

Question 12: What is the average vertex out-degree corresponding to your answer from the previous question? (3 points)

Written Questions: Populating Arrays for Empirical Studies

In future problem sets, you will be asked to perform empirical analyses on the sorting algorithms we will be studying. In order to test the best, worst, and average cases of some of these algorithms, you will need to fill arrays with values with various characteristics. For simplicity, we will work with arrays of integers.

We will not be coding them yet for this problem set, but let's think about how this could be done. For the questions in this section, describe how you would generate a set of n integer values to store in an array that has each of the following characteristics.

Question 13: Random values within a given range. When you write this later as a method, the upper and lower bounds of the range of values will be given as parameters. (3 points)

Question 14: Values already sorted in ascending order. Note: it is important that you generate these efficiently – for example, you should not generate sorted input by generating random input then sorting it. Generate it in sorted order right from the start. Be sure to have at least some randomness to the values generated. (4 points)

Question 15: Values already sorted in descending order. Again, make sure you do this efficiently and have some randomness. (2 points)

Question 16: Values "nearly" sorted in ascending order. Here, when you write this as a method later, it will take a parameter that specifies the fraction of entries that are out of order. For example, if the parameter has a value of 0.05, approximately one of out of each 20 array slots should contain a value that's out of order. There are many reasonable ways to accomplish this. (4 points)

Grading

This assignment will be graded out of 85 points.

Feature	Value	Score
Q1	16	
Q2	14	
Q3	16	
Q4	1	
Q5	1	
Q6	3	
Q7	3	
Q8	1	
Q9	4	
Q10	5	
Q11	5	
Q12	3	
Q13	3	
Q14	4	
Q15	2	
Q16	4	
Total	85	