SIENAcollege
Computer Science

Computer Science 385
Design and Analysis of Algorithms
Siena College
Spring 2017

Homework Set 1
Due: Start of Class, Friday, February 10, 2017

You may work alone or with a partnet 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 group members, should you choose to work in a group. Only one submission per group is needed.

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

This assignment has 103 points available, but the maximum score is 100. So you can get 3 points off and still earn a "perfect" 100.

Programming Task: Exhaustive Search

Sometimes an exhaustive search algorithm considers all permutations of a set. It is not obvious though how to generate these permutations in a program. Write a program that prints all permutations of integers $1..n$. Your program should take n as an input value, and it should work for any $n \geq 1$. If you would like to use a language other than Java, ask, and it will likely be approved. (10 points)

Programming Task 2: A New Graph Method

In an undirected graph, such as the `HighwayGraph` from Lab 2, the *degree* of a vertex is the number of adjacent (undirected) edges.

1. Download a new copy of the `HighwayGraph.java` starter you used for Lab 2.
2. Add a new method that computes the degree of the vertex with a given number. So the method call added in `main`:

   ```
   int d = g.degree(12);
   ```

   would compute and return the degree of vertex 12 and store it in `d` (4 points).
3. Remove the `System.out.println(g);` line from the given `main` method.
4. Add code in `main` that finds the largest degree of all the waypoints (vertices) in the graph, and then prints out a line containing the name of the input graph file, the largest degree, and the vertex label where you found that largest degree (4 points).

Print and attach or paste into your submission your new `degree` method, the code you added to the `main` method, and your program's output on the `DC-all.tmg`, `NY-all.tmg`, and `tm-master.tmg` graphs.

? Question 1:
What is the worst case running time of your `degree` method? (2 points)

Programming Task: Generating Example Arrays

In future homework sets, you will be asked to perform empirical analysis on the sorting algorithms we will be studying. As part of that task, you will be required to generate input arrays for the sorting algorithms. In order to test the best, worst, and average cases of some of these algorithms, you will need to generate input arrays with various characteristics. For simplicity, we will sort arrays of `int`.

- an array filled with n random values within a given range
- an array filled with n values sorted in ascending order
- an array filled with n values sorted in descending order
- an array filled with n values "nearly" sorted in ascending order

You may use any programming language. If you use Java, implement it within a class `IntArrayGenerator` that includes `static` methods to generate and return arrays of `int` with each of the above characteristics, and a `main` method that tests these methods for various values of n and ranges. Be sure you can achieve similar functionality if you choose a different language. 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. (10 points)

Homework Set Problems

Question 2:
Levitin Exercise 2.3.1, p. 67 (4 points)

Question 3:
Levitin Exercise 2.3.2, p. 67 (2 points)

Question 4:
Compare the asymptotic growth rates of the following pairs of functions and decide if one grows faster than the other or if they grow at the same rate. Prove your answers using limits. All logarithms are base 2, unless otherwise noted. (5 points)

a. $5n^2 + 2n + 5$ and n^2

b. $n \log n$ and n^2

c. $2^{\log n}$ and n

d. 2^n and 2^{2n}

e. n^r and n^s, where r and s are arbitrary fixed constants such that $r > s > 0$.

Question 5:
Prove the following using limits (4 points):

a. $n \in \Omega(\log n)$

b. $5n + 6 \in \Theta(n)$

c. $n \log n \in \Omega(n)$

d. $n \in \Theta(n - n^{\frac{1}{2}})$

Question 6:
For each of the following functions, indicate the class $O(g(n))$ to which the function belongs. Use the "smallest" $g(n)$ possible to obtain the tightest bound. You need not prove your assertions in this case, but justify your choice informally. (2 points)

a. $\frac{1}{n} + 12$

b. $\frac{\sin n}{n}$

Question 7:

For each of the following functions, indicate the class $O(g(n))$ to which the function belongs. Use the "smallest" $g(n)$ possible to obtain the tightest bound, unless otherwise specified. Prove your assertions using the definition of $O(g(n))$ (*i.e.*, produce constants). (6 points)

a. $762n^2 - 56n + 37$

b. $(n^3 + 100)^5$

c. $n \log n$, use $g(n) = n^2$

Question 8:

For each of the following functions, indicate the class $\Omega(g(n))$ to which the function belongs. Use the "largest" $g(n)$ possible to obtain the tightest bound unless otherwise specified. Prove your assertions using the definition of $\Omega(g(n))$ (*i.e.*, produce constants). (6 points)

a. $762n^2 - 56n + 37$

b. $2^n + n^2$

c. $n \log n$, use $g(n) = n$

Question 9:

For each of the following functions, indicate the class $\Theta(g(n))$ to which the function belongs. Prove your assertions using the definition of $\Theta(g(n))$ (*i.e.*, produce constants). (5 points, 1 for the first, 4 for the second)

a. $762n^2 - 56n + 37$ (hint: you've already done this, just bring it all together)

b. $n^4 + 50n^2 - 23$

Question 10:

For questions below, consider this overly friendly code segment. (5 points)

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

a. Trace the code to count how many times it prints `Hello!` for both $n = 2$ and $n = 3$.

b. Write an expression involving two summations that counts the number of times it prints `Hello!` for any value of n.

c. Simplify your expression to get a closed form formula for the number of times it prints `Hello!` Show your work.

d. Check your formula from part c by substituting $n = 2$ and $n = 3$ to see if it matches your answers for parts a and b.

Note: you may also test your formula by implementing this code and then running it for different values of n.

Question 11:

Suppose you have an Amazon gift certificate worth $1000. You want to buy exactly two items from Amazon, and you want their cost to exactly equal $1000 so that you spend the full amount of the gift certificate. Write a brute force algorithm that takes as input an array of the Amazon item prices and outputs true if there are two distinct prices in the array whose sum is $1000. For example, suppose the array contains the prices A = {40, 225, 725, 10, 610, 812, 275, 350, 99}. Then the output would be true because 725 + 275 = 1000. For this exercise, design a brute force algorithm solving this problem using $\Theta(n^2)$ operations in the worst case. Use the pseudocode below to get started. (10 points)

```
// A[0..n-1] is an array of n prices.  This algorithm returns true if
// there are two distinct prices that sum to 1000, and false otherwise.
ALGORITHM GiftCertificate( A[0n-1] )
```

Question 12:

For the questions below, consider the graph representations discussed in class for storing directed graphs. (Total 6 points, 3 each part)

Suppose that the amount of memory required by the adjacency matrix graph representation is exactly $\frac{|V|^2}{8}$ bytes. (This assumes each Boolean value in the array is stored as a single bit.) In addition, assume the amount of memory required by the adjacency lists graph representation is exactly $16|V| + 16|E|$ bytes[1].

a. If you have a graph with 1,000,000 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 16GB of main memory which is typical of a PC today?

b. 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 1,000,000 vertices meaning that 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.

[1]Justification for why $16|V| + 16|E|$ is a reasonable ballpark estimate for the adjacency lists representation from class: It counts 4 bytes for each reference in the `vertices[]` array, 12 bytes for each `Vertex` object (4 bytes for the `head` reference plus 8 bytes needed by Java to store housekeeping information about each `Vertex` object), and 16 bytes per `Edge` object (4 bytes for the integer `dst`, 4 bytes for the `next` reference, and 8 bytes to store Java housekeeping information about each `Edge` object).

Question 13:
Levitin Exercise 1.3.1, p. 23 (4 points)

Question 14:
Levitin Exercise 3.1.8, p. 103 (2 points)

Question 15:
Levitin Exercise 3.1.9, p. 103 (2 points)

Question 16:
Levitin Exercise 3.1.11, p. 103 (2 points)

Question 17:
Levitin Exercise 3.1.13, p. 103 (2 points)

Question 18:
Levitin Exercise 3.2.4, p. 107 (3 points)

Question 19:
Levitin Exercise 3.2.6, p. 107 (3 points)