SIENAcollege
Computer Science

Computer Science 385
Design and Analysis of Algorithms
Siena College
Spring 2018

Problem Set 2

Group Formation: 4:00 PM, Wednesday, February 7, 2018
Due: 4:00 PM, Friday, February 16, 2018

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

All GitHub repositories must be created with all group members having write access and all group member names specified in the `README.md` file by 4:00 PM, Wednesday, February 7, 2018. This applies to those who choose to work alone as well!

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 assignment if you wait until the last minute. A slow and steady approach will be much more effective.

Getting Set Up

You will receive an email with the link to follow to set up your GitHub repository, which will be named `ps2-yourgitname`, for this problem set. One member of the group should follow the link to set up the repository on GitHub, then that person should email the instructor with the other group members' GitHub usernames so they can be granted access. This will allow all members of the group to clone the repository and commit and push changes to the origin on GitHub.

Again, this must be complete and all group member names listed in the `README.md` file by 4:00 PM, Wednesday, February 7, 2018.

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.

Your submission requires that all required code deliverables are committed and pushed to the master for your repository's origin on GitHub. If you see everything you intend to submit when you visit your repository's page on GitHub, you're set.

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)

Homework Set Problems

Question 1:

Levitin Exercise 2.3.1, p. 67 (4 points)

Question 2:

Levitin Exercise 2.3.2, p. 67 (4 points)

Question 3:

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. (10 points)

a. $\frac{1}{5}n^2 + 17n + 1$ and n^2

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

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

d. 2^n and $2^{\frac{n}{2}}$

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

Question 4:

Prove the following using limits (5 points):

a. $n^2 \in \Omega(n \log n)$

b. $5n^3 + 6 \in \Theta(n^3)$

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

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

e. $25n^2 - 7n + 23 \in O(n^2 \log n)$

Question 5:

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

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

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, unless otherwise specified. Prove your assertions using the definition of $O(g(n))$ (*i.e.*, produce constants). (6 points)

a. $23n^2 - 99n + 12$

b. $(n^4 + 56)^4$

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

Question 7:

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. $23n^2 - 99n + 12$

b. $3^n + n^3$

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

Question 8:

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). (6 points, 2 for the first, 4 for the second)

a. $23n^2 - 99n + 12$ (hint: you've already done this, just bring it all together)

b. $n^3 + 12n^2 - 5$

Question 9:

For questions below, consider this overly enthusiastic code segment. (6 points)

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

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

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

c. Simplify your expression to get a closed form formula for the number of times it prints `Algorithms!` 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 10:

Suppose you are a contestant on *The Price is Right* and your pricing game requires that you select two from a collection of items whose prices add up to exactly \$100. If you're right, you win... **a new car!!!!** Write a brute force algorithm that takes as input an array of the item prices and outputs true if there are two distinct prices in the array whose sum is \$100. For example, suppose the array contains the prices A = {40, 22, 72, 10, 65, 81, 27, 35, 99}. Then the output would be true because 65 + 35 = 100. 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. You do not need to write an implementation, just a pseudocode algorithm. (10 points)

ALGORITHM BOBBARKER(P)
//Input: a set of item prices $P[0..n-1]$
//Output: true only if 2 distinct elements of P add to exactly 100

Question 11:

Levitin Exercise 1.3.1, p. 23 (4 points)

Question 12:
Using the selection sort algorithm we looked at in class, show the steps it will follow to sort the array { G, O, S, I, E, N, A}. (3 points)

Question 13:
What does it mean for a sorting algorithm to be *stable*? (3 points)

Question 14:
Describe two circumstances where sorting of data is needed, such that for one of the circumstances it is important that the algorithm used for sorting is stable, and one where it does not matter. (4 points)

Question 15:
Is bubble sort as we studied in class a stable sorting algorithm? Explain briefly. (2 points)

Question 16:
Is selection sort as we studied in class a stable sorting algorithm? Explain briefly. (2 points)

Question 17:
Count the number of character comparisons made by the **BruteForceStringMatch** algorithm when applied to the pattern and text below. (3 points)

text: `THESE_ARE_NOT_THE_DROIDS_YOU_ARE_LOOKING_FOR`

pattern: `ARTOO`

Note that the text is 44 characters long, including spaces (but was incorrectly shown as 42 earlier).

(When you are done, you may `MOVE_ALONG,_MOVE_ALONG` to the next question.)

Question 18:
Levitin Exercise 3.2.6, p. 107 (3 points)

Question 19:
Levitin Exercise 3.4.8, p. 121 (3 points)

Question 20:
Consider an instance of the Knapsack Problem in which there are n=4 items with weights 7, 3, 12, and 8 and the knapsack can hold up to a total weight of 20. The item values are 6, 3, 8, and 12. Enumerate all the packings of the knapsack that would be considered using an exhaustive search algorithm solving this problem, and circle the packing that would be selected as the best solution. (4 points)