Computer Science 385
Design and Analysis of Algorithms
Siena University
Spring 2026

Problem Set 2

Group Formation: 4:00 PM, Monday, February 9, 2026
Due: 4:00 PM, Friday, February 13, 2026

You may work alone or in a group of 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, 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, Monday, February 9, 2026. 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

In Canvas, you will find a link to follow to set up your GitHub repository, which will be named `ps2-yourgitname`, for this lab. Only one member of the group should follow the link to set up the repository on GitHub, then others should request a link to be granted write access.

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 $n \geq 1$. AI assistance is permitted but must be thoroughly documented. If you would like to use a language other than Java, ask, and it will likely be approved. (8 points)

Take a highly recursive approach: choose each next possible unused value and follow it by all permutations of the remaining values. Do not use algorithms like those in Levitin Section 4.3. We're focusing on brute force solutions here.

Question 1: Explain clearly how your program works, including the purpose of any major variables and/or parameters. (7 points)

Homework Set Problems

Question 2: Compute the following sums. (2 points)

$$\sum_{i=10}^{n+3} 1 \qquad \sum_{i=10}^{n+3} i \qquad \sum_{i=1}^{n} i(i-1) \qquad \sum_{j=0}^{n} 5^{j+2}$$

Question 3: Levitin Exercise 2.3.2, p. 67. You do not need to form a fully-computed closed form for these, but proceed far enough to know its Θ class. (2 points)

Question 4: Use limits to compare the asymptotic growth rates of the following pairs of functions. Fully simplify your limits to get an exact result of a number or ∞. Clearly state if one grows faster than the other or if they grow at the same rate. All logarithms are base 2, unless otherwise noted. (5 points)

a. $\frac{2}{9}n^3 + 6n^2 - 12$ and n^3

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

c. $4^{\log n}$ and n

d. 2^n and $2^{\frac{3n}{2}}$

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

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

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

b. $2n^4 + 16n \in \Theta(n^4)$

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

d. $\frac{2}{5}n^2 - 17n + 3 \in O(n^2 \log n)$

Question 6: For this question, consider the function $f(n) = 8n^3 - 17n + 19$. (6 points)

a. What is the simplest function $g(n)$ (that is, no unnecessary coefficients or extra terms) such that $f(n) \in \Theta(g(n))$?

b. State a simple function $g(n)$, with a different polynomial degree than your answer for part a, such that $f(n) \in O(g(n))$.

c. State a simple function $g(n)$, with a different polynomial degree than your answer for part a, such that $f(n) \in \Omega(g(n))$.

Question 7: For each of the following, prove the assertion using the definitions of $O(g(n))$, $\Omega(g(n))$, or $\Theta(g(n))$, as appropriate. Follow the procedures we used n class and lab to produce positive constants. (17 points, 2 each except h is worth 3)

a. $7n^2 + 9n - 1200 \in O(n^2)$

b. $(n^2 + 5n)^3 \in O(n^8)$

c. $n \log n \in O(n^2)$

d. $7n^2 + 9n - 1200 \in \Omega(n^2)$

e. $2^n + n^{100} \in \Omega(2^n)$

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

g. $7n^2 + 9n - 1200 \in \Theta(n^2)$ (hint: you've already done most of this, just bring it all together)

h. $5n^3 - 128n^2 - 50 \in \Theta(n^3)$

Question 8: For this question, consider this very wintry code segment to answer the parts below (6 points: a:1, b:2, c:2, d:1)

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

a. Trace the code to count how many messages it prints for both $n = 2$ and $n = 3$.

b. Write an expression involving two summations that counts the number of messages it prints for any value of n

c. Simplify your expression to get a closed form formula for the number of messages it prints. Show your work.

d. Check your formula from the previous question by substituting $n = 2$ and $n = 3$ to see if it matches your hand-traced answers to the first question in this section.

Question 9: You wouldn't know it here in Loudonville, but Major League Baseball spring training camps are just about to open. This inspired you and a group of friends to pick a game to attend. You have been chosen to acquire the group's tickets. Your group of size p is too large to get tickets all together but now you want to see if you can break into two subgroups such that two blocks of tickets are available to accommodate your group perfectly. Write a brute force algorithm that takes as input an array of the numbers of seats in contiguous blocks currently available, and the number of people in your group. It should output true if there are two distinct blocks in the array whose size sum to p. For example, suppose the array that contains the block sizes is S = {4, 2, 7, 10, 11, 8, 12, 3. 10}, and there are 20 people in your group. Then the output would be true because 8 + 12 = 20. 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 TicketGroups($S$, $p$)
    //Input: a set of available ticket block sizes, $S[0..n-1]$
    //Input: the size of your group, $n$
    //Output: true only if 2 distinct block sizes in $S$ add to exactly $p$
```

Question 10: Count the number of character comparisons made by the **BruteForceStringMatch** algorithm when applied to the pattern and text below. Explain how you arrived at your answer. (4 points)

text: `FROM_COMPUTER_SCIENCE_COMES_COMPUTING_FUN`

pattern: `COMPUTERS`

Note that the text is 41 characters long, and the pattern is 9 characters long.

Question 11: Levitin Exercise 3.2.6, p. 107 (3 points)

Question 12: For this question, suppose you wish to take an input string and count how many substrings you can form from it such that each substring starts with `[` and ends with `]`:. (19 points: a:3, b:5, c:4, d:7)

Examples: the input string `]385[` contains 0 such substrings, `a[b]x]` contains 2 such substrings, and the input string `B[][RA[[]CK][[E]T[S` contains 15 such substings.

a. List the 15 substrings of the longest input string above that start with `[` and end with `]`.

b. Write pseudocode for a very brute-force algorithm (nested loops, checking all possible start and end substring positions) that will solve this problem. Note that the algorithm returns a count, it does not need to print or return a list of all substrings. Do not try to be efficient.

c. Determine the efficiency class of your algorithm for an input string of length n, and briefly justify. Specify both best and worst cases if appropriate (don't worry about trying to nail down an average case).

d. Now write a more efficient (no nested loops, but still just brute-force) algorithm to solve this problem in $\Theta(n)$ time.

Question 13: Consider an instance of the Knapsack Problem in which there are n=4 items with weights 13, 7, 4, and 15 and the knapsack can hold up to a total weight of 24. The item values are 30, 15, 40, and 60. 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. (7 points)

Grading

This assignment will be graded out of 100 points.

Feature	Value	Score
Recursive Permutations	8	
Q1	7	
Q2	2	
Q3	2	
Q4	5	
Q5	4	
Q6	6	
Q7	17	
Q8	6	
Q9	10	
Q10	4	
Q11	3	
Q12	19	
Q13	7	
Total	100	