CS-GY 6033: Design and Analysis of Algorithms I

Fall 2024

Instructor: Nairen Cao **NYU Tandon School of Engineering**

Problems	**1**	**2**	**3**	**4**	**5 (Ungraded)**	**6 (Ungraded)**	**7 (Ungraded)**	**Total**
Max. Score	30	25	15	30	0	0	0	100

Table 1: Score Summary

Homework #1

Assigned: 9/13/2024 **Due: 9/26/2024**

General Instructions Write all solutions clearly, concisely, and legibly. It is okay to provide hand-written and scanned/photographed solutions as long as they are clearly legible. If typing, using LaTeX is recommended.

You should generally describe algorithms in words and high-level pseudocode. Specifically, emphasize the goal of each step in your algorithm rather than the coding details. After giving a high-level description, provide more detail on the implementation of each step where appropriate.

The purpose of having these problem sets is to enhance your mathematics and programming skills in the scope of designing algorithms. These problems may be quite challenging, so discussions with classmates are encouraged. However, you should spend at least half an hour thinking about the problems individually before discussing them. Although discussions (and consulting ChatGPT and WolframAlpha) are allowed, you need to write the solution in your own words and by yourself. Please acknowledge any person or reference you discuss with or consult from.

Special Instructions Problems 5, 6, and 7 are ungraded and are included for practice purposes. You are encouraged to attempt these problems to further improve your understanding of the material, but they will not be included in the overall grading of this problem set.

---

1 Asymptotic Notation Comparison

Take the following list of functions and arrange them in ascending order of growth rates. That is, if function $g(n)$ immediately follows function $f(n)$ in your list, then it should be the case that $f(n)$ is $O(g(n))$. You do not need to justify your answer for this problem part.

- $f_1(n) = \sqrt{2n}$
- $f_2(n) = 2^{\sqrt{\log n}}$
- $f_3(n) = \log n$
- $f_4(n) = 100n$
- $f_5(n) = n^2 \log n$
- $f_6(n) = n^{1/\log\log n}$
- $f_7(n) = n!$
- $f_8(n) = 2^n \cdot n!$
- $f_9(n) = n^{0.99n}$
- $f_{10}(n) = n^n$

Recall that $\log(n!) > 0.99n \log n$ for sufficiently large enough n.

2 Asymptotic Notation Properties

Assume both f, g, h, k are asymptotically positive functions. Decide if each of the following statements is true or false. If it is true, prove it; if it is false, give a counterexample.

(a) Does $f(n) = O(g(n))$ and $g(n) = O(f(n))$ imply that $f(n) = \Theta(g(n))$? Prove or disprove.

(b) Does $f(n) = O(g(n))$ imply $f(n)^2 = O(g(n)^2)$? Prove or disprove.

(c) Does $f(n) = O(g(n))$ and $h(n) = O(k(n))$ imply $f(n) \cdot h(n) = O(g(n) \cdot k(n))$? Prove or disprove.

(d) Does $f(n) = O(g(n))$ and $h(n) = O(k(n))$ imply $f(h(n)) = O(g(k(n)))$? Prove or disprove.

(e) Does $f(n) = O(g(n))$ imply $\log f(n) = O(\log g(n))$? Prove or disprove.

3 Euclidean Algorithm

Consider the following Euclidean algorithm for computing the greatest common divisor of two integers $a, b > 0$:

Algorithm 1 Euclid(a, b)

```
1: while $b > 0$ do
2:     $t \leftarrow b$
3:     $b \leftarrow a \mod b$
4:     $a \leftarrow t$
5: end while
6: return $a$
```

(a) Prove that the algorithm terminates in $O(\log a)$ iterations.

(b) Show that the $O(\log a)$ bound is tight. In other words, prove that there exists a constant $c > 0$ such that for every $n_0 > 0$, there exist two integers a, b such that $a > b > n_0$ and Euclid(a, b) takes at least $c \log a$ iterations to complete.

4 Two Sum with Simple Restriction

- An array $A[0, \ldots, n-1]$ of integers, where $n \geq 3$.
- An integer target value T.

Objective: Design an algorithm to find indices i, j such that:

$$|A[i] + A[j] - T|$$

is minimized, with the following constraint:

1. The indices i and j must satisfy $|i - j| > 1$.

Output:

- Return **one pair** of indices (i, j) that forms the sum closest to the target value T, subject to the constraint.

Requirements:

1. **Algorithm:** Provide a clear pseudocode that solves the problem.
2. **Correctness:** Prove the correctness of your algorithm by explaining why it works under the given constraint and guarantees the closest sum to the target.
3. **Running Time:** Analyze the running time of your algorithm and determine its time complexity. Full credit will be awarded for designing an algorithm with a time complexity of $O(n \log n)$.

Example

Input:

- $A = [1, 4, 2, 6, 9]$
- $T = 8$

Output:

- $(i, j) = (0, 3)$

Explanation:

- The closest valid sum to the target $T = 8$ is $A[0] + A[3] = 1 + 6 = 7$, with a difference of 1.
- The indices $i = 2$ and $j = 3$ do not satisfy the constraint since $|2-3| = 1$, even though $A[2]+A[3] = 2 + 6 = 8$ is exactly equal to T.

5 Asymptotic Notations Comparison Table (Ungraded)

For each pair of functions f and g in the following table, indicate whether $f = O(g)$, $f = \Omega(g)$, $f = \Theta(g)$, $f = o(g)$, and $f = \omega(g)$ respectively. You only need to give a yes/no answer for each question.

Problem	$f(n)$	$g(n)$	O	Ω	Θ	o	ω
(a)	$3n^2 + 10$	$5n^2 - 6n$					
(b)	$\log_{10} n$	$5\log_2(n^3) + 3$					
(c)	$10n^2 - n$	$n^2 \log n$					
(d)	$n^3 - 4n^2 + 10$	n^2					
(e)	2^{3n}	7^n					
(f)	$n^{\sin(n)}$	1					

Table 2: Comparison of Function Growth Rates (Ungraded)

6 Asymptotic Notation Proof (Ungraded)

Prove that $n^3 + 100n^2 = \Theta(n^3)$. (Deal with $O(n^3)$ and $\Omega(n^3)$ separately.)

7 Algorithm Warmup (Ungraded)

Consider the following basic problem. You're given an array A consisting of n integers $A[0], A[1], \ldots, A[n-1]$. You'd like to output a two-dimensional n-by-n array B in which $B[i, j]$ (for $i < j$) contains the sum of array entries $A[i]$ through $A[j]$. Here's a simple algorithm for this problem:

Algorithm 2 Sum Array Entries and Store Results

```
1: for $i = 0$ to $n - 1$ do
2:     for $j = i + 1$ to $n - 1$ do
3:         Add up the array entries $A[i]$ through $A[j]$
4:         Store the result in $B[i, j]$
5:     end for
6: end for
```

(a) Choose a function $f(n)$ and prove that the running time of this algorithm is $O(f(n))$ on any input of size n.

(b) For the same function, show that the running time of the algorithm on input size n is also $\Omega(f(n))$.

(c) Explain how you can improve the algorithm to get asymptotically better running time.