# Algorithm Design and Analysis
# Assignment 01: Algorithm Analysis

| | |
|---|---|
| **Total Points:** | 40 |
| **Assigned:** | Saturday, 13th January 2024 |
| **Due:** | Friday, 19th January 2024, 23.59pm. |
| **Type of Assignment:** | Individual |
| **Submission format:** | PDF on Canvas |

**Instructions**

- Submit individual solutions to this assignment. Remember that you may brainstorm with your peers (e.g. your study group), but everyone should write up their solution independently. Remember also that you may not search for solutions to assignment questions online, nor may you refer to such solutions if you happen to come across them.
- As mentioned in the course syllabus, remember to list anyone you brainstorm or discuss with on the assignment. Refer to other course policies in the syllabus.
- All algorithms must be written in a well-structured way, like the examples in the problems and in the textbook.

**Preparation**

Study Chapter 2 (up to Section 2.3) as well as Chapter 3 (at least up to Section 3.1) of the Levitin textbook. You are encouraged to study with your study group. Discuss key ideas as well as things you do not understand and bring questions to office hours.

**Problem 1*[8pts]***

*(Reference: Levitin textbook, Exercise 2.1, #3)*

For each of the following functions, indicate the class $\Theta(g(n))$ the function belongs to. (Use the simplest g(n) possible in your answers.) Prove your assertions in two different ways: (1) using lower bound & upper bound arguments(i.e. definitions of big-oh/big-omega, and big-theta), and (2) using limits.

**a.** $10n^4 + 7n^2 + 3n$ **b.** $2n \log n + 10n$

**c.** $\sqrt{10n^4 + 7n^2 + 3n}$ **d.** $(n^3 + 1)^6$

**Problem 2** ***[4pts]***

***Part A****: (Reference: Levitin textbook, Exercise 2.1, #2)*

Determine whether the following statements are true or false. Justify your answers.

**a.** $2n(n-1)/2 \in O(n^3)$ **b.** $2n(n-1)/2 \in O(n^2)$

**c.** $2n(n-1)/2 \in \Theta(n^3)$ **d.** $2n(n-1)/2 \in \Omega(n)$

*Part B ;[5pts]*

For each of the following functions, indicate the class $\Theta(g(n))$ the function belongs to. (Use the simplest $g(n)$ possible in your answers.) Prove your assertions.

**a.** $(n^2+1)^{10}$ **b.** $\sqrt{10n^2+7n+3}$

**c.** $2n \lg(n+2)^2 + (n+2)^2 \lg \frac{n}{2}$ **d.** $2^{n+1} + 3^{n-1}$

**e.** $\lfloor \log_2 n \rfloor$

**Problem 3*[10pts]***

***Counting sort*** assumes that each of the $n$ input elements is an integer in the range 0 to $k$, for some integer $k$. Counting sort determines, for each input element $x$, the number of elements less than $x$. It uses this information to place element $x$ directly into its position in the output array. For example, if 17 elements are less than $x$, then $x$ belongs in output position 18. In the code below for counting sort, we assume that the input is an array A[1...$n$], and thus A.length = $n$. We require two other arrays: the array B[1...$n$] holds the sorted output, and the array C[1...$n$] provides temporary working storage.

```
COUNTING-SORT(A, B, k)
 1  let C[0..k] be a new array
 2  for i = 0 to k
 3      C[i] = 0
 4  for j = 1 to A.length
 5      C[A[j]] = C[A[j]] + 1
 6  // C[i] now contains the number of elements equal to i.
 7  for i = 1 to k
 8      C[i] = C[i] + C[i − 1]
 9  // C[i] now contains the number of elements less than or equal to i.
10  for j = A.length downto 1
11      B[C[A[j]]] = A[j]
12      C[A[j]] = C[A[j]] − 1
```

The figures below show the operation of COUNTING-SORT on an input array A of length 8, where each element of A is a nonnegative integer no larger than $k$ = 5.

(a) The array A and the auxiliary array C after line 5

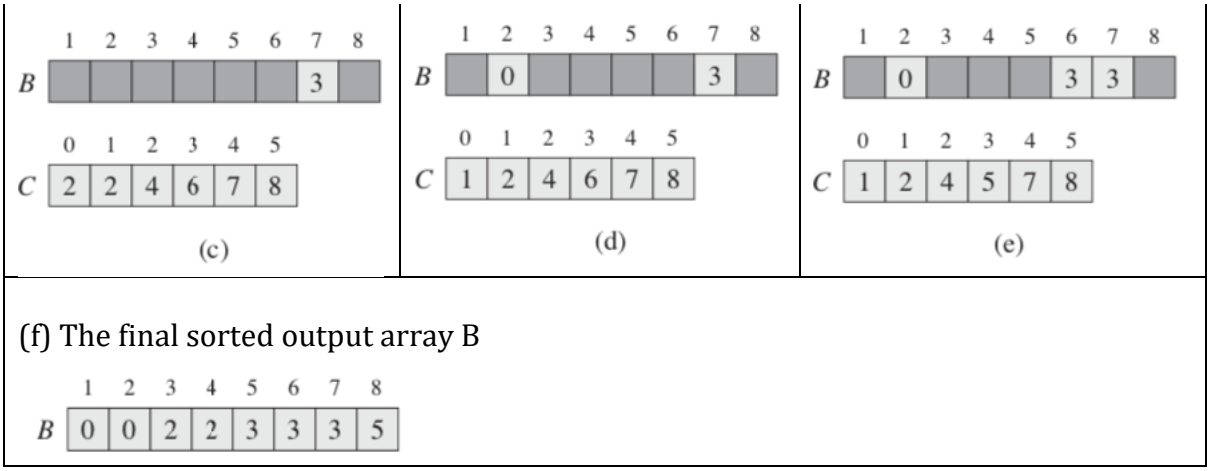
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| A | 2 | 5 | 3 | 0 | 2 | 3 | 0 | 3 |

| | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| C | 2 | 0 | 2 | 3 | 0 | 1 |

(b) The array C after line 8

| | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| C | 2 | 2 | 4 | 7 | 7 | 8 |

(c) – (e)
The output array B and the auxiliary array C after one, two and three iterations of the loop in lines 10-12, respectively. Only the lightly shaded elements of array B have been filled in




(f) The final sorted output array B




(i) Using the example above as a guide, illustrate the operation of counting sort on the array A = [6,0,2,0,1,3,4,6,1,3,2]

(ii) Analyze the runtime of counting sort. You can assume that the integer *k* is a constant < *n.*

- (*To answer this, you should identify the basic operation, write and simplify a summation formula, etc. Finally, you can simply state its efficiency class, i.e. Θ(…), without formally proving the efficiency class*)

(iii) What would be the implication if the assumption in part (ii) was not true? Why do you think counting sort is not generally used for most sorting tasks?

**Problem 4** ***[6pts]***

Below is an algorithm for finding the Gaussian Elimination of a matrix A. Analyze the runtime of this algorithm. You do not need to bother to understand how the algorithm works.

**ALGORITHM** *GaussElimination*$(A[1..n, 1..n], b[1..n])$

```
//Applies Gaussian elimination to matrix A of a system's coefficients,
//augmented with vector b of the system's right-hand side values
//Input: Matrix A[1..n, 1,..n] and column-vector b[1..n]
//Output: An equivalent upper-triangular matrix in place of A with the
//corresponding right-hand side values in the (n + 1)st column
for i ← 1 to n do A[i, n + 1] ← b[i]    //augments the matrix
for i ← 1 to n − 1 do
    for j ← i + 1 to n do
        for k ← i to n + 1 do
            A[j, k] ← A[j, k] − A[i, k] ∗ A[j, i] / A[i, i]
```

## Problem 5 *[7pts]*

Insertion sort is one of the sorting algorithms we will be studying. Below is an algorithm for Insertion sort.

**ALGORITHM** *InsertionSort*$(A[0..n-1])$
//Sorts a given array by insertion sort
//Input: An array $A[0..n-1]$ of $n$ orderable elements
//Output: Array $A[0..n-1]$ sorted in nondecreasing order
**for** $i \leftarrow 1$ **to** $n-1$ **do**
    $v \leftarrow A[i]$
    $j \leftarrow i-1$
    **while** $j \geq 0$ **and** $A[j] > v$ **do**
        $A[j+1] \leftarrow A[j]$
        $j \leftarrow j-1$
    $A[j+1] \leftarrow v$

Write a version of the insertion sort algorithm to work on a linked list rather than an array. Does this algorithm work with the same $\Theta(n^2)$ runtime as the array version? Why? Give reasons.