# CS3000: Algorithms & Data — Summer 2025 — Laney Strange

Exam 1 Practice Problems

- These practice problems are to help you prepare for Exam 1. We'll release the solutions on Tuesday (May 20) so you can go over them and ask questions in your recitation.
- Exam 1 takes places during lecture on May 22, 2025
- The exam will take the entirety of lecture (11:40am-1:20pm). You'll hand it in on paper, but we'll scan it later for grading on gradescope.
- You may bring one 8.5x11-inch paper as a cheat sheet, with anything written or typed on it (one side only). You will submit this cheat sheet along with your exam, and you will not be permitted to use any other materials or notes during the exams.
- If you have a DAS accommodation for exams, please arrange to take the exam at their center. Make sure you schedule that time ASAP if you haven't yet!

**Problem 1.** *Practice - Loop Invariant*

Below is the Pseudocode for a procedure called BUILD-MAX-HEAP, which turns an sorted array into a max-heap. You may assume that MAX-HEAPIFY works correctly to restore the heap properties on a structure where everything, except possibly the root at position $i$, is already a max-heap.

Prove that BUILD-MAX-HEAP correctly builds a max-heap by showing the following loop invariant: At the start of each iteration of the for loop of lines 2-3, each node $i+1, i+2, ..., n$ is the root of a max heap.

BUILD-MAX-HEAP(A, n)

1 $A.heapsize = n$
2 **for** $i = \lfloor n/2 \rfloor$ **downto** 1
3 MAX-HEAPIFY$(A, i)$

- Initialization
- Maintenance
- Termination

**Solution:**

**Problem 2.** *Practice - Binary Search*

Suppose you are given a sorted array $A$ of non-zero integers and you want to determine whether there exists two distinct indices $i \neq j$ such that $A[i] = -A[j]$. Give pseudocode for an algorithm that solves this problem using Binary Search as a subroutine. Your algorithm should return indices $(i, j)$ if they exist (any pair can be returned if there are multiple such indices) or NIL if there is no such pair.

**Solution:**

**Problem 3.** *Practice - Bounds*

Identify whether $f(n) = O(g(n))$, $g(n) = O(f(n))$, or both.

(a) $f(n) = 2n$, $g(n) = 4n$
**Solution:**

(b) $f(n) = \sqrt{n}$, $g(n) = \lg n$
**Solution:**

**Problem 4.** *Practice - Growth of Functions*

Order the following functions from slowest growing to fastest growing:

$$\lg n, \quad n^3, \quad n \lg n, \quad n!, \quad n^{500}, \quad 5^n, \quad n, \quad \lg \lg n$$

**Solution:**

**Problem 5.** *Practice - Heaps*

(a) Draw the heap represented by the array $\langle 16, 14, 10, 8, 7, 9, 3, 2, 4, 1 \rangle$ and state whether it is a valid min- or max-heap.

**Solution:**

(b) Given the heap in the previous portion, draw what it would look like after the root is removed, the final element is put into the root, and the structure is re-heapified.

**Solution:**

**Problem 6.** *Practice - Mystery Run-Time*

Given the below pseudocode, give a tight bound.

```
MYSTERY(A, n, B, m)
1  result = 0
2  for i = 1 to n
3        for j = 1 to m
4              result = result + A[i] · B[j]
5  return result
```

**Solution:**

**Problem 7.** *Practice - Loop Invariant*

Below is the pseudocode for Insertion Sort.

INSERTIONSORT$(A, n)$

```
1  for i = 2 to n
2      key = A[i]
3      j = i − 1
4      while j > 0 and A[j] > key
5          A[j + 1] = A[j]
6          j = j − 1
7      A[j + 1] = key
```

Prove the correctness of Insertion Sort by showing the following loop invariant: At the start of iteration $i$ of the for loop of lines 1-7, the subarray $A[1..i-1]$ is in sorted order.

(For this proof, you can assume that the inner **while** loop correctly finds where *key* should be placed among the already-sorted subarray, and places it there in line 7.)

- Initialization:

- Maintenance:

- Termination:

**Problem 8.** *Practice - Best Case Run-Time*

(a) Given the below pseudocode, calculate the runtime $T(n)$ for MYSTERY-2$(A, n)$ and a tight bound on $T(n)$ for the best case scenario. For simplicity, you can assume $n$ will always be even. Show all of your work, including the table.

```
MYSTERY-2(A, n)
1  for i = 1 to ⌊n/2⌋
2       if A[i] mod 2 == 0
3            found = LINEAR-SEARCH(A, n, 2 · A[i])
4            if found ≠ NIL
5                 return FALSE
6  return TRUE
```

| line | cost | times |
|---|---|---|
| 1 | | |
| 2 | | |
| 3 | | |
| 4 | | |
| 5 | | |
| 6 | | |

**Solution:**

(b) Give an array example for A that would satisfy the best case condition you found in the previous part

**Solution:**

**Problem 9.** *Practice - Sorting*

Consider the problem of sorting an Array $A[1, ..., n]$. In general, sorting algorithms are known to have a lower bound of $\Omega(n \lg n)$. For this problem, we're going to try to beat that bound in certain cases.

(a) Give an efficient algorithm for sorting a Boolean array $B[1, ..., n]$ that is more efficient than $O(n \lg n)$. Assume that you want FALSE to come before TRUE.

**Solution:**

(b) Give a tight bound on the worst-case run-time for your algorithm.

**Solution:**

**Problem 10.** *Practice - Rotation*

Prior to being passed to your function, a sorted array of integers $A = \langle A_1, A_2, \ldots, A_n \rangle$ is possibly rotated at an unknown pivot index $1 \leq k < n$, such that the resulting array is $\langle A_{k+1}, A_{k+2}, \ldots, A_n, A_1, A_2, \ldots, A_k \rangle$. For example, if $A = \langle 0, 1, 2, 4, 5, 6, 7 \rangle$ is rotated and becomes $\langle 4, 5, 6, 7, 0, 1, 2 \rangle$, then the pivot index $k = 3$.

Give pseudocode for an algorithm that runs in linear time which calculates the value of $k$ given a rotated input array. Describe your approach in a few sentences and write pseudocode.

**Solution:**

**Problem 11.** *Practice - Insertion Sort*

Given the following array $A = \langle 1, 10, 2, 3, 15, 4 \rangle$ write out what the array would look like after each iteration of the for loop of insertion sort.

**Solution:**

**Problem 12.** *Practice - Recursive Sequences*

(a) Given that $a_k = 2k + 1$, find $a_0$, $a_1$, ..., $a_5$ and determine the recurrence relation $a_n$:

**Solution:**

(b) Given the recurrence relation $a_n = a_{n-1} + 4$ with base case $a_1 = 2$, find $a_2$, $a_3$, ..., $a_5$ and find the closed form for $a_k$.

**Solution:**

(c) Given the recurrence relation $a_n = 2a_{n-1} - 1$ with the base case $a_1 = 3$, find the terms $a_2$, $a_3$, ..., $a_5$ and a closed form for $a_k$.

**Solution:**

(d) Given the following sequence, determine the recurrence relation $a_n$:

$$\begin{aligned} a_1 &= 20 \\ a_2 &= -10 \\ a_3 &= 5 \\ a_4 &= -2.5 \\ a_5 &= 1.25 \end{aligned}$$

**Solution:**

(e) Given the following sequence, determine the reccurence relation $a_n$:

$$a_1 = 4$$
$$a_2 = 11$$
$$a_3 = 25$$
$$a_4 = 53$$
$$a_5 = 109$$

**Solution:**

**Problem 13.** *Practice - Solve a Recurrence*

(a) Use the iteration method to solve the recurrence given by $T(n) = T(n-1) + n + c$ with base case $T(1) = 1$

**Solution:**

(b) Give a tight bound on the recurrence.

**Solution:**

**Problem 14.** *Practice - Solve a Recurrence*

(a) Use the iteration method to determine the closed form of the recurrence given by $T(n) = n^3 + 2T\left(\frac{n}{2}\right)$ with the base case $T(1) = 1$.

**Solution:**

(b) Give a bound on the recurrence. You may find it helpful to remember that any finite geometric sequence with $r \neq 1$ sums to a constant as shown in the below equation

$$\sum_{k=0}^{n-1} ar^k = \begin{cases} a\left(\frac{1-r^n}{1-r}\right) & r \neq 1 \\ an & otherwise \end{cases}$$

**Solution:**