ICS 311: Algorithms Spring 2025

Problem Set 2

Kyle Berney *Due: Friday, January 24, 2025 at 4pm*

You may discuss the problems with your classmates, however **you must write up the solutions on your own** and **list the names** of every person with whom you discussed each problem.

Start **every** problem on a separate page, with the exception of Problems 1 (Peer credit assignment) – Problem 1 can be on the same page as any other problem. Any problem that starts on the same page as some other problem will receive 0 points!

1 Peer Credit Assignment (1 point extra credit for replying)

Please list the names of the other members of your peer group for this week and the number of extra credit points you think they deserve for their participation in group work.

- You have a total of 60 points to allocate across all of your peers.
- You can distribute the points equally, give them all to one person, or do something in between.
- You need not allocate all the points available to you.
- ***You cannot allocate any points to yourself!*** Points allocated to yourself will not be recorded.

2 Runtime analysis (25 pts)

Consider the following algorithm:

```
1: FancyLoop(A[1..n])
2:     for i = 1 to n
3:         for j = 1 to i − 1
4:             for k = 1 to 50
5:                 A[i] = A[j] + k
```

(a) **(15 pts)** Compute the exact number of times line 5 will execute. Show your work (using summations).

(b) **(10 pts)** State the asymptotic running time of FancyLoop$(A[1..n])$ using Θ notation.

3 MultiPrint (25 pts)

Consider the following algorithm:

```
1: MultiPrint(A, n)
2:     for (i = 0; i < n, i = i + 1)
3:         for (j = 0; j < n; j = j + ⌈n/10⌉)
4:             print(A[j])
```

(a) **(15 pts)** Compute the exact number of times line 4 will execute. Show your work (using summations).

(b) **(10 pts)** State the asymptotic running time of MultiPrint(A, n) using Θ notation.

4 Matrix Transpose (50 pts)

The transposition of a matrix flips the elements of the matrix across its diagonal. In other words, the rows of the matrix become the columns and the columns become the rows. Given a square $n \times n$ matrix, matrix transposition can be performed in-place (without the use of any additional matrices) by carefully swapping elements.

```
1: MatrixTranspose(A[1..n][1..n])
2:     for i = 1 to n − 1
3:         for j = i + 1 to n
4:             exchange A[i][j] with A[j][i]
```

To show that the above algorithm correctly computes matrix trasposition, we need to show for all $i \in \{1, 2, \ldots, n\}$ and $j \in \{1, 2, \ldots n\}$, such that $i \neq j$, that $A[i][j]$ has been swapped with $A[j][i]$.

(a) **(20 pts)** State precisely the loop invariant for the inner **for** loop (line 3), and prove that this loop invariant holds. Your proof should use the structure of the loop invariant proof presented in Chapter 2 of CLRS. *Hint: To come up with a good loop invariant, think about what exactly the algorithm is doing.*

(b) **(20 pts)** Using the termination condition of the loop invariant proved in part (a), state the loop invariant for the outer **for** loop (line 2). Again, your proof should use the structure of the loop invariant proof presented in Chapter 2 of CLRS.

(c) **(10 pts)** What is the worst-case running time of MatrixTranspose($A[1 \ldots n][1 \ldots n]$)? Show your work (using summations).

(d) **(OPTIONAL - 0 pts)** Think about why you cannot use your loop invariant (i.e., what would go wrong in your proof) if we changed the matrix transposition code to the following:

```
1: MatrixTranspose(A[1..n][1..n])
2:     for i = 1 to n
3:         for j = 1 to n
4:             exchange A[i][j] with A[j][i]
```

5 Algorithm analysis and correctness (OPTIONAL - 0 pts)

Consider the following algorithm:

```
1: COMPLEX(n)
2:     j = 1
3:     a = 0
4:     while j ≤ n
5:         a = a + j
6:         j = 2 · j
7:     return a
```

(a) Determine the **precise** number of iterations that lines 5-6 of the code above executes. Prove your answer using **strong** induction. *Hint: Observe how the value of j changes in each iteration.*

(b) Using your answer in part (a), determine the asymptotic running time of the COMPLEX(n) algorithm (use Θ notation).

(c) Determine the value that COMPLEX(n) returns (as a function of n) and prove your answer using loop invariants, as presented in Chapter 2 of CLRS. *Hint: Use the answer from part (a) and observe how it affects the value of a in each iteration.*

(d) Prove your answer in part (c) using ***strong*** *induction.* Observe the similarity between loop invariants and induction.

6 BubbleSort (OPTIONAL - 0 pts)

BubbleSort is a popular but inefficient sorting algorithm. It works by repeatedly swapping adjacent elements that are out of order, and doing so enough times that there can be no more elements out of order.

```
1: BUBBLESORT(A[1..n])
2:     for i = 1 to n − 1
3:         for j = n downto i + 1
4:             if A[j] < A[j − 1]
5:                 exchange A[j] with A[j − 1]
```

Since BubbleSort never removes or adds items to the array — it merely swaps them in line 5 — in order to prove that BUBBLESORT correctly sorts the input array, we only need to show that at the end of the algorithm $A[1] \leq A[2] \leq A[3] \leq \cdots \leq A[n-1] \leq A[n]$. You will do this using loop invariants.

(a) State precisely the loop invariant for the inner **for** loop (line 3), and prove that this loop invariant holds. Your proof should use the structure of the loop invariant proof presented in Chapter 2 of CLRS. *Hint: To come up with a good loop invariant, think about what exactly the algorithm is doing.*

(b) Using the termination condition of the loop invariant proved in part (a), state the loop invariant for the outer **for** loop (line 2) that will allow you to prove the inequality $A[1] \leq A[2] \leq A[3] ... \leq A[n-1] \leq A[n]$. Again, your proof should use the structure of the loop invariant proof presented in Chapter 2 of CLRS.

(c) What is the worst-case running time of BUBBLESORT? Show your work (using summations). How does it compare asymptotically to the running time of InsertionSort?