CSCI 3104 Fall 2024
Instructor: Dr. Lijun Chen

Problem Set 1 [100 pts]

Due Date ..... September 10, 2024
Name ..... **Hyma Jujjuru**
Student ID **110743269**
Collaborators ..... **Apollo Hoffman, Lance Kluge**

Contents

Instructions

- The solutions **should be typed**, using proper mathematical notation. We cannot accept hand-written solutions. Here's a short intro to LaTeX.
- You should submit your work through the **class Gradescope page** only (linked from Canvas). Please submit one PDF file, compiled using this LaTeX template.
- You may not need a full page for your solutions; pagebreaks are there to help Gradescope automatically find where each problem is. Even if you do not attempt every problem, please submit this document with no fewer pages than the blank template (or Gradescope has issues with it).
- You are welcome and encouraged to collaborate with your classmates, as well as consult outside resources. You must **cite your sources in this document. Copying from any source is an Honor Code violation. Furthermore, all submissions must be in your own words and reflect your understanding of the material.** If there is any confusion about this policy, it is your responsibility to clarify before the due date.
- Posting to **any** service including, but not limited to Chegg, Reddit, StackExchange, etc., for help on an assignment is a violation of the Honor Code.

- You **must** virtually sign the Honor Code (see Section). Failure to do so will result in your assignment not being graded.

Honor Code (Make Sure to Virtually Sign) [10 pts]

Problem HC.
- My submission is in my own words and reflects my understanding of the material.
- Any collaborations and external sources have been clearly cited in this document.
- I have not posted to external services including, but not limited to Chegg, Reddit, StackExchange, etc.
- I have neither copied nor provided others solutions they can copy.

Agreed (signature here). I agree to the above, Hyma Jujjuru. □

1 Proof by Induction and Loop Invariants

1.1 Problem 1 [10 pts]

Problem 1. A student is trying to prove by induction that for all $n \geq 1$, the sum of the first n odd numbers is a perfect square.

Student's Proof. The proof is by induction on $n \geq 1$.

- **Base Case:** When $n = 1$, the sum of the first 1 odd number is 1, which is equal to 1^2. Thus, the statement holds for $n = 1$.
- **Inductive Hypothesis:** Now suppose that for some $k \geq 2$, the sum of the first k odd numbers is k^2.
- **Inductive Step:** We now consider the $k + 1$ case. We need to show that the sum of the first $k + 1$ odd numbers is $(k+1)^2$. By the inductive hypothesis, the sum of the first k odd numbers is k^2. Adding the next odd number $2k - 1$ to this sum gives:

$$k^2 + (2k - 1) = (k + 1)^2$$

Therefore, the sum of the first $k + 1$ odd numbers is $(k + 1)^2$, which is a perfect square. The result follows by induction.

However, there are two errors in this proof:

(a) The Inductive Hypothesis is not correct. Write an explanation to the student explaining why their Inductive Hypothesis is not correct. Rewrite a corrected Inductive Hypothesis. **[5 pts]**

Answer. The student assumes that the statement is true for all $k \geq 2$ which is circular reasoning because that is the statement she is trying to prove.
Corrected Induction Hypothesis: For any $k = n$, the sum of the first n odd numbers is a perfect square. □

(b) The Inductive Step is not correct. Write an explanation to the student explaining why their Inductive Step is not correct. Rewrite a corrected Inductive Step. **[5 pts]**

Answer. The student wrote that $k^2 + 2k - 1 = (k + 1)^2$ which is incorrect because $(k + 1)^2 = k^2 + 2k + 1$. Instead of $(2k - 1)$, the student should add $(2k + 1)$ as the next odd number.
Corrected Induction Step: $k^2 + (2k + 1) = (k + 1)^2$

□

1.2 Problem 2 [15 pts]

Problem 2. Consider the recurrence relation, defined as follows:

$$T_n = \begin{cases} 2 & : n = 0, \\ 22 & : n = 1, \\ -2T_{n-1} + 35T_{n-2} & : n \geq 2. \end{cases}$$

Prove by induction that $T_n = (-1) \cdot (-7)^n + 3 \cdot (5)^n$, for all integers $n \in \mathbb{N}$. [**Recall:** $\mathbb{N} = \{0, 1, 2, \ldots\}$ is the set of non-negative integers.]

Proof. **Base Case:** Consider three cases: $k = 0, k = 1, k = 2$ where $k \in n$.

- Let $k = 0$.

  By the recurrence relation, $T_0 = 2$.

  By the equation, $T_0 = (-1) \cdot (-7)^0 + 3 \cdot (5)^0 = -1 \cdot 1 + 3 \cdot 1 = -1 + 3 = 2$.

  Thus, the statement holds for $k = 0$.

- Let $k = 1$.

  By the recurrence relation, $T_1 = 22$.

  By the equation, $T_1 = (-1) \cdot (-7)^1 + 3 \cdot (5)^1 = -1 \cdot -7 + 3 \cdot 5 = 7 + 15 = 22$.

  Thus, the statement holds for $k = 1$.

- Let $k = 2$.

  By the recurrence relation, $T_2 = -2T_1 + 35T_0 = -2 \cdot 22 + 35 \cdot 2 = -44 + 70 = 26$.

  By the equation, $T_2 = (-1) \cdot (-7)^2 + 3 \cdot (5)^2 = -1 \cdot 49 + 3 \cdot 25 = -49 + 75 = 26$.

  Thus, the statement holds for $k = 2$.

Induction Hypothesis: Assume that $T_k = -2T_{k-1} + 35T_{k-2}$ for $k \geq 2$
Induction Step: We show that the statement holds for $k+1$ or that $T_{k+1} = -1 \cdot (-7)^{k+1} + 3 \cdot (5)^{k+1}$ holds.

$$\begin{aligned} T_{k+1} &= -2T_{k+1-1} + 35T_{k+1-2} \\ &= -2T_k + 35T_{k-1} \\ &= -2(-1 \cdot (-7)^k + 3 \cdot (5)^k) + 35(-1 \cdot (-7)^{k-1} + 3 \cdot (5)^{k-1}) \\ &= 2 \cdot (-7)^k - 6 \cdot (5)^k + 35(\frac{1}{7} \cdot (-7)^k + \frac{3}{5} \cdot (5)^k) \\ &= 2 \cdot (-7)^k - 6 \cdot (5)^k + 5 \cdot (-7)^k + 21 \cdot (5)^k \\ &= 7 \cdot (-7)^k + 15 \cdot (5)^k \\ &= -1 \cdot -7 \cdot (-7)^k + 3 \cdot 5 \cdot (5)^k \\ &= -1 \cdot (-7)^{k+1} + 3 \cdot (5)^{k+1} \end{aligned}$$

Therefore, $T_{k+1} = -1 \cdot (-7)^{k+1} + 3 \cdot (5)^{k+1}$. So, by induction, $T_n = (-1) \cdot (-7)^n + 3 \cdot (5)^n$, for all integers $n \in \mathbb{N}$. $\square$

1.3 Problem 3 [15 pts]

Problem 3. Use mathematical induction to show that when $n \geq 4$ is an exact power of 3, the solution of the recurrence

$$T(n) = \begin{cases} 3 & \text{if } n = 3, \\ 3T(n/3) + n & \text{if } n > 3 \end{cases}$$

is $T(n) = n \log_3 n$.

Proof. **Base Case:** Let $k = 9$ which is the first exact power of 3 that is greater than or equal to 4.
By the recurrence relation, $T(9) = 3T(\frac{9}{3}) + 9 = 3T(3) + 9 = 3 \cdot 3 + 9 = 9 + 9 = 18$
By the equation, $T(9) = 9 \log_3 9 = 9 \cdot 2 = 18$
Thus the statement holds for $k = 9$.

Induction Hypothesis: Assume that $T_k = 3T(k/3) + k = k \log_3 k$.
Induction Step: The $(k+1)^{th}$ term is the next exact power of 3 so $(k+1)^{th}$ term is $3k$. So, we show that the statement holds for $k+1$ or that $T(k+1) = 3k \log_3 3k$.

$$\begin{aligned} T(k+1) &= T(3k) \\ &= 3T(3k/3) + 3k \\ &= 3T(k) + 3k \\ &= 3[k \log_3 k] + 3k \\ &= 3k \log_3 k + 3k \log_3 3 \\ &= 3k \log_3 (k \cdot 3) \\ &= 3k \log_3 (3k) \end{aligned}$$

Therefore, $T(k+1) = 3k \log_3 (3k)$. So, by induction, $T_n = n \log_3 n$, for all integers $n \geq 4$ that are exact powers of 3. □

1.4 Problem 4 [25 pts]

Problem 4. Consider the following algorithm which computes the mean of a non-empty array of integers. The mean of a nonempty list $a_1, \ldots, a_k$ of k numbers is defined to be $\frac{1}{k}\sum_{i=1}^{k} a_i$. Please state any assumptions you make.

```
procedure mean(A):
    mu = A[0]
    n = len(A)
    for (i = 1; i<n; ++i)
        mu = (i*mu + A[i]) / (i+1)
    return mu
```

(1a) State a valid loop invariant for the `for` loop in the algorithm that is useful for proving the correctness of the algorithm. **[10 pts]**

(1b) Provide the initialization component of a proof of correctness of this algorithm that uses your loop invariant from (1a). **[5 pts]**

(1c) Provide the maintenance component of a proof of correctness of this algorithm that uses your loop invariant from (1a). That is, assume that your loop invariant holds just before the i-th iteration of the `for` loop and use this assumption to show that your loop invariant holds just before iteration $(i+1)$. **[5 pts]**

(1d) Provide the termination component of a proof of correctness of this algorithm that uses your loop invariant from (1a). **[5 pts]**

Proof.

(1a) Loop Invariant: At each iteration where $1 \leq i < n$, *mu* contains the mean of the integers $a_1, ...a_i$.

(1b) Initiation of LI: At the first iteration, *mu* contains the value of the first integer of the array A. So, *mu* contains the mean of the first integer because any integer divided by 1 is itself.

(1c) Maintenance: During each iteration, *mu* is set to the mean of the integers $a_1, ...a_i$ at each iteration where i represents the index of the array A that is updated each iteration.

(1d) Termination: When the loop ends, *mu* contains the mean of the non-empty array of integers, A.

□

1.5 Problem 5 [25 pts]

Problem 5. Consider an array A of n numbers. Begin sorting the array by repeatedly comparing adjacent elements and swapping them if they are in the wrong order. This process should start at the beginning of the array and continue until you reach the end. After the first pass, the largest element will be in its correct position at the end of the array. Continue this process for the first $n-1$ elements of the array.

(a) Write pseudocode for this sorting algorithm, which is commonly known as the bubble-sort algorithm. **[15 pts]**

(b) Identify and explain the loop invariant maintained by this algorithm. Use it to prove correctness of the algorithms. **[5 pts]**

(c) Explain why the algorithm only needs to run for the first $n-1$ elements rather than for all n elements. **[5 pts]**

Proof. (a) Pseudocode:

```
1:  BubbleSort(A)
2:      for(i = 0 to i < n - 1)
3:          swapped = false
4:          for(j = 0 to j < n - i - 1)
5:              if(A[j] > A[j+1])
6:                  swap(A[j], A[j+1])
7:                  swapped = true
8:          if(swapped = false) break
9:      return A
```

(b) **Loop Invariant:** At each iteration, the right-most x integers are sorted.
Initiation: At the first iteration, the largest integer in the array is placed at the index $(n-1)$. So, the rightmost integer is sorted.
Maintenance: During each iteration, we ensure that the values are placed in ascending order by pushing a value to the right-most index as long as the integer value is greater than the one before it and less than the one after.
Termination: After the loop is terminated, all the integers or the right-most integers from indices 0 to n are sorted.
So, the loop invariant shows that the array is gradually sorted from the right-most index to the first index (or index 0) during loop execution and produces an array whose integers are sorted in ascending order.

(c) The algorithm only needs to run $n-1$ times because we are comparing j and $j+1$ which means each index is compared to its right neighbor. So, when $j = n-1, j+1 = n$, the n^{th} term is compared to the $(n-1)^{th}$ term which means we are checking all n elements in the array. □