Name: $\boxed{\text{Connor O'Reilly}}$
ID: $\boxed{107054811}$

CSCI 3104, Algorithms **Charlie Carlson & Ewan Davies**
Problem Set 1 – Due Setpember 4th **Fall 2020, CU-Boulder**

Advice 1: For every problem in this class, you must justify your answer: show how you arrived at it and why it is correct. If there are assumptions you need to make along the way, state those clearly.

Advice 2: Informal reasoning is typically insufficient for full credit. Instead, write a logical argument, in the style of a mathematical proof.

Instructions for submitting your solutions:

- The solutions **should be typed using** LaTeXand we cannot accept hand-written solutions. Here's a short intro to LaTeX.
- You should submit your work through the **class Canvas page** only.
- 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 a document with at least as many pages as the blank template (or Gradescope has issues with it). **We will not accept submissions with fewer pages than the blank template**. Submissions with more pages are fine.
- **You must CITE any outside sources you use, including websites and other people with whom you have collaborated. You do not need to cite a CA, TA, or course instructor.**
- **Posting questions to message boards or tutoring services including, but not limited to, Chegg, StackExchange, etc., is STRICTLY PROHIBITED. Doing so is a violation of the Honor Code.**

Quicklinks: 1 2 3 4

Name: Connor O'Reilly

ID: 107054811

CSCI 3104, Algorithms **Charlie Carlson & Ewan Davies**

Problem Set 1 – Due Setpember 4th **Fall 2020, CU-Boulder**

Problem 1. Prove by induction that for each $n \in \mathbb{Z}^+$,

$$\sum_{k=1}^{n} \frac{1}{k^2} \leq 2 - \frac{1}{n}.$$

Proof.

- **Base Case:**
  Consider the case of n = 1 so that $\Sigma_{k=1}^{n} \frac{1}{k^2} = \Sigma_{k=1}^{1} \frac{1}{k^2} = 1$ and the RHS $2 - \frac{1}{n} = 2 - \frac{1}{1} = 1$

- **Inductive Hypothesis:**
  Assume there exists an $n \geq 1$ such that

$$\Sigma_{k=1}^{n} \frac{1}{k^2} \leq 2 - \frac{1}{n}$$

- **Inductive step:**
  Stepping from a hypothesis on n to $n+1$

$$\Sigma_{k=1}^{n+1} \frac{1}{k^2} \leq \Sigma_{k=1}^{n} \frac{1}{k^2} + \frac{1}{(n+1)^2}$$

$$\leq (2 - \frac{1}{n}) + \frac{1}{(n+1)^2}$$

$$2 - \frac{1}{(n+1)} \leq (2 - \frac{1}{n}) + \frac{1}{(n+1)^2}$$

$$2 - \frac{1}{(n+1)} \neq 2 - \frac{n^2+n+1}{n\,(n+1)^2}$$

The above inequality does not hold true for all values meaning the inductive step failed proving the claim false.

□

Name: Connor O'Reilly
ID: 107054811

CSCI 3104, Algorithms **Charlie Carlson & Ewan Davies**
Problem Set 1 – Due Setpember 4th **Fall 2020, CU-Boulder**

Problem 2. What are the three components of a loop invariant proof? Write a 1–2-sentence description for each one.

The three components of a loop invarient proof are initialization, Maintenance and Termmination.

- Initialization: The initialization includes a claim that is true prior to the first iteration of the loop.
- Maintenance: The maintenance includes a claim that is true prior to every iteration and still holds true after that iteration.
- Termination: The termination includes a claim in which after the loop exits the invariant should give a useful property to show the correctness of the algorithm.

Problem 3. Consider the following algorithm.

```
FindMinElement(A[1, ..., n]) : //array A is not empty
    ret = A[n]
    for i = 1 to n-1 {
        if A[n-i] < ret{
            ret = A[n-i]
    }}
    return ret
```

Do the following.

(a) Suppose a student provides the following: *At the start of each iteration, i is one more than the number of iterations that have occurred.* Is this a valid loop invariant? Justify your answer in light of Problem 2.

Answer:
According to the definitions given in the previous answer it completes both the initialization and maintenance step for an invariant proof but does not provide any useful information to prove the correctness of the algorithm. Basically it would fail the Termination phase for the loop invarient proof.

(b) Is the above invariant in part (a) *useful* in proving that the FindMinElement algorithm is correct? If so, explain why. If not, give a *useful* loop invariant and explain why your invariant is useful in proving the algorithm correct. **Note that this question is *not* asking you to prove that the algorithm is correct.**

Answer:
As stated above the invariant is not useful in proving that the FindMinElement algorithm is correct. A useful invariant would be *At the start of each iteration i, the variable ret should contain the smallest element in subarray $A'[n-i, ..., n]$ such that ret $\leq A'[n-i, ..., n]$*

Name: Connor O'Reilly
ID: 107054811

CSCI 3104, Algorithms — **Charlie Carlson & Ewan Davies**
Problem Set 1 – Due Setpember 4th — **Fall 2020, CU-Boulder**

Problem 4. Consider the following algorithm. We seek to prove that the algorithm is correct using a loop invariant proof.

```
ProductArray(A[1, ..., n]) : //array A is not empty
    product = 1
    for i = 1 to n {
        product = product * A[i]
    }
    return product
```

(a) Provide a loop invariant that is *useful* in proving the algorithm is correct.
Loop Invariant: At the start of each iteration i of the loop, the variable *product* should contain the product of all numbers in the subarray $A'[1..i]$.

(b) Using the loop invariant above, provide the **initialization** component of the loop invariant proof. That is, show that the loop invariant holds before the first iteration of the loop is entered.

Proof. **Initilization:** Prior to the start of the first iteration, $i = 1$ and the loop invariant states that the variable *product* should contain the product of all numbers in the sub array $A'[1, .., 1]$. the array only contains a single element 1, and $product = 1$. □

(c) Using the loop invariant above, provide the **maintenance** component of the loop invariant proof. That is, assume the loop invariant holds just before the i-th iteration of the loop, and use this assumption to show that it still holds just before the $(i+1)$-st iteration.

Proof. **Maintenance:** Assuming the loop invariant is true at the start of every i-th iteration, the variable *product* would be equal to the product of all elements in the sub-array $A'[1, ..., i]$. During the i-th iteration the value of $A[i]$ is multiplied with product so at the start of the $i + 1$ iteration the variable *product* will be equal to the product of all elements in sub-array $A'[1, ..., i + 1]$. □

(d) Using the loop invariant above, provide the **termination** component of the loop invariant proof. That is, assume the loop invariant holds just before the last iteration. Then argue that the loop invariant holds after the loop terminates, based on what happens in the last iteration of the loop. Finally, use this to argue that the algorithm overall is correct.

Proof. **Termination:** At the termination of the loop, the loop exits with $i = n$ and the loop invariant states that the variable *product* will contain the product of all elements in the sub-array $A'[1, ..., i]$. Which in this case is the whole array $A'[1, ..., n] = A[1, ..., n]$ Proving our algorithm is correct. □