Name: Jeremy M. Hein
ID: 810952267

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: Jeremy M. Hein
ID: 810952267

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. We will prove by induction on n that $\sum_{k=1}^{n} \frac{1}{k^2} \leq 2 - \frac{1}{n}$ is true.

Base Case: Let n = 1, we have:

$\frac{1}{1^2} \leq 2 - \frac{1}{1} = 1 \leq 1$

Since 1 is less than or equal to itself, our base case is true.

Inductive Hypothesis: Assume that for some $n = j$ where $j \in \mathbb{Z}^+$,

$$\sum_{k=1}^{j} \frac{1}{k^2} \leq 2 - \frac{1}{j},$$

then we need to prove:

$$\sum_{k=1}^{j+1} \frac{1}{k^2} \leq 2 - \frac{1}{j+1}$$

Name: Jeremy M. Hein
ID: 810952267

Inductive Step:

$$\begin{aligned}
\sum_{k=1}^{j+1} \frac{1}{k^2} &\leq 2 - \frac{1}{j} + \frac{1}{(j+1)^2} && \text{(By Inductive Hypothesis)} \\
&\leq 2 - \left(\frac{(j+1)^2}{j\,(j+1)^2} + \frac{j}{j\,(j+1)^2} \right) && \text{(Common Denominator)} \\
&\leq 2 - \left(\frac{j^2 + 2j + 1 + j}{j\,(j+1)^2} \right) && \text{(Expanding Numerator)} \\
&\leq 2 - \frac{j^2 + j}{j\,(j+1)^2} - \frac{2j+1}{j\,(j+1)^2} && \text{(Split Fractions / Distribute Negative)} \\
&\leq 2 - \frac{1}{j+1} - \frac{2j+1}{j\,(j+1)^2} && \text{(Factoring)} \\
\sum_{k=1}^{j+1} \frac{1}{k^2} &\leq 2 - \frac{1}{j+1} - \frac{2j+1}{j\,(j+1)^2} \leq 2 - \frac{1}{j+1} && \text{(Conclusion)}
\end{aligned}$$

Because $j \in \mathbb{Z}^+$, we can conclude $2 - \frac{1}{j+1} - \frac{2j+1}{j(j+1)^2} \leq 2 - \frac{1}{j+1}$, thus $\sum_{k=1}^{j+1} \frac{1}{k^2} \leq 2 - \frac{1}{j+1}$ and our hypothesis is proven.

□

Name: Jeremy M. Hein
ID: 810952267

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

Initialization: It holds prior to the first iteration/cycle of the loop.

Maintenance: Assuming it was true before an iteration of the loop, it remains true after the iteration (before the next iteration).

Termination: After the loop exits, the invariant gives a useful property that helps show the correctness of the algorithm.

Source: Instructor lecture slides

Name: Jeremy M. Hein
ID: 810952267

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: Yes, it's valid because it is true for each iteration of the loop, so by definition it is invariant. Before we have entered the loop, $i = 1$ and we have completed 0 iterations. When $i = 2$, we have completed 1 iteration, so on and so forth.

(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: No, it's not useful in proving that the algorithm is correct. A more useful invariant would be "Before the start of an iteration, the variable "ret" holds the minimum element in the sub-array $A[n, n-1, ...n-i]$." This is useful because it's true, and that it implies when our loop terminates we have both checked every element in array A, and that ret holds the minimum element in that array, which is precisely the goal of this algorithm.

Name: Jeremy M. Hein
ID: 810952267

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.

Before each iteration, the variable "product" holds the product of all elements in the sub-array $A[1, ..., i-1]$.

(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. Before the first iteration, product $= 1$, which is correct since the first iteration should be the first element in array A, and $1 * A[1] = A[1]$. □

(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. Assuming our Loop Invariant holds just before the i-th iteration, product $= A[1] * A[2] * .... * A[i-1]$, meaning product holds the product of all elements in sub-array $A[1, ..., i-1]$.

During the i-th iteration product = product $*A[i] = A[1] * A[2] * ... * A[i-1] * A[i]$, so product now holds the product of all elements in sub-array $A[1, ..., i-1, i]$.

Since we need product to hold the product of $A[1, ..., i-1]$ just before the i-th iteration, we need product to hold the product of $A[1, ..., i-1, i]$ just before the $(i+1)$-st iteration $(i+1-1=i)$. Since product $= A[1] * A[2] * ... * A[i-1] * A[i]$ after the i-th iteration, the invariant does hold just before the $(i+1)$-st iteration. □

(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. Assuming our Loop Invariant holds just before the final iteration, product = $A[1]*A[2]*....*A[n-1]$, meaning product holds the product of all elements in sub-array $A[1, ..., n-1]$.

During the n-th iteration $i = n$ and product = product $*A[i] = A[1] * A[2] * ... * A[n-1]*A[n]$, so product now holds the product of all elements in sub-array $A[1, ..., n-1, n]$.

Since product holds the product of all elements in sub-array $A[1, ..., n-1, n]$, we can see that the sub-array is now the same as array A. After the iteration i is now also $> n$, so we exit the loop and our algorithm returns the variable product which holds the product of all elements in array A. □