COSC 302: Analysis of Algorithms — Spring 2018
Prof. Darren Strash
Colgate University

Problem Set 1 — Invariants and Induction

Due by 4:30pm Friday, Feb. 2, 2018 as a single pdf via Moodle (either generated via LaTeX, or concatenated photos of your work). Late assignments are not accepted.

This is an *individual* assignment: collaboration (such as discussing problems and brainstorming ideas for solving them) on this assignment is highly encouraged, but the work you submit must be your own. Give information only as a tutor would: ask questions so that your classmate is able to figure out the answer for themselves. It is unacceptable to share any artifacts, such as code and/or write-ups for this assignment. If you work with someone in close collaboration, you must mention your collaborator on your assignment.

Suggested practice problems (not to be turned in): 2.1-2, 2.1.4, 2.2-2, 2.2-4

1. Problem 2.1-3 in CLRS, 3rd edition.

```
LINEAR-SEARCH(A, v)
1: for j = 1 to A.length
2:     if A[j] = v then
3:         return j
4: return NIL
```

Invariant 1 (Loop invariant for LINEAR-SEARCH). *Before iteration j of the **for** loop on line 1, v is not contained in $A[1..j-1]$.*

We now show that this invariant holds during initialization, that it is maintained throughout the algorithm, and that it implies correctness of the algorithm at termination.

Initialization: $j = 1$ and the subarray $A[1..0]$ is empty. Therefore v trivially is not in $A[1..j-1]$.

Maintenance: Suppose just before loop j that v is not in $A[1..j-1]$, then if we get to the start of the loop after executing the loop body it must be that the **if** condition on line 2 failed, and therefore $A[j] \neq v$. Thus, at the beginning of the next loop (loop $j+1$) v is not contained in $A[1..j]$.

Termination: The algorithm terminates either when it finds v on line 3 or if it does not find v on line 4. On line 3 it correctly returns if it finds v, and if line 4 is reached, then by our invariant v is not in $A[1..A.length]$, and therefore it correctly returns NIL.

2. Problem 2-2 in CLRS, 3rd edition.

   Solution:

   (a) We need to show that $A'[1..n]$ is a permutation of A. Otherwise, A' could just contain any elements in sorted order.

   (b) Invariant for the inner loop of BUBBLESORT:

   Invariant 2 (loop invariant for the inner loop of BUBBLESORT)**.** *Before executing the body of the* ***for*** *loop on line 2, $A[j-1]$ is the smallest element among $A[j-1..n]$ and $A[j-1..n]$ is a permutation of $A[j-1..n]$ from the previous loop.*

   We now show that this invariant holds during initialization, that it is maintained between iterations of the loop, and that it implies correctness of the algorithm at termination.

   Initialization: $j = n$ and the subarray $A[n..n]$ consists of one element that is trivially the smallest among $A[n..n]$ and a permutation.

   Maintenance: Suppose just before loop $A[j]$ is the smallest element in $A[j..n]$, then either $A[j-1] < A[j]$ or $A[j-1]$ and $A[j]$ are swapped on line 4, and then $A[j-1]$ is the smallest elements in $A[j-1..n]$ at the beginning of loop $j-1$. Furthermore, since elements are only swapped, $A[j-1..n]$ is still a permutation.

   Termination: Before the final iteration of the **for** line 4 (where $j = i+1$ and the loop body does not execute), then $A[i]$ is the smallest number in $A[i..n]$ and $A[i..n]$ is a permutation of $A[i..n]$ before the first iteration of the loop.

   (c) Invariant for BUBBLESORT:

   Invariant 3 (loop invariant for BUBBLESORT)**.** *Before executing the body of the* ***for*** *loop on line 1, $A[1..i-1]$ contains, in sorted order, the $i-1$ smallest elements originally in A, for all $a \in A[1..i-1]$ and all $b \in A[i..n]$ $a \leq b$. Furthermore, A is a permutation of the elements originally in A.*

   Initialization: $i = 1$ and the subarray $A[1..0]$ is empty; the invariant trivially holds.

   Maintenance: Suppose just before loop iteration i that $A[1..i-1]$ that the invariant holds. Then, by Invariant 2, $A[i]$ has the smallest value in $A[i..n]$ when the inner loop completes. Note that $A[i]$ is at least as large as the largest value in $A[1..i-1]$ (mainly, $A[i-1]$) since $a \in A[1..i-1]$ and $b \in A[i..n]$ $a \leq b$. Therefore $A[i]$ is the i-th smallest value from among the elements in the original array A and now $A[1..i]$ is sorted and contains the i smallest elements from the original array A. Furthermore, since elements are only swapped (in line 4) and never removed or added, A remains a permutation.

   Termination: Before the final iteration of the **for** loop, $i = n+1$ and by the invariant elements $A[1..n]$ are sorted, containing the first n elements from the original array A, and A is a permutation. Therefore, A is sorted.

(d) The worst-case running time of BUBBLESORT is $\Theta(n^2)$. The inner loop always executes $n - i$ times in one iteration of the outer loop, giving us the running time

$$\Theta\left(\sum_{j=1}^{n-1}(n-i)\right) = \Theta(n^2),$$

which is the same as selection sort and insertion sort.

3. Prove by induction that for every non-negative integer n

$$\sum_{k=0}^{n} k^2 = \frac{n(n+1)(2n+1)}{6}.$$

Solution:

Proof. Base case: $n = 0$, then both $\sum_{i=0}^{0} i^2 = 0^2 = 0 = \frac{0(0+1)(2 \cdot 0+1)}{6}$.

Inductive step: Assume that

$$\sum_{i=0}^{k-1} i^2 = \frac{(k-1)(k-1+1)(2(k-1)+1)}{6} = \frac{(k-1)k(2k-1)}{6} = \frac{2k^3 - 3k^2 + k}{6}.$$

Then,

$$\begin{aligned}
\sum_{i=0}^{k} i^2 &= \sum_{i=0}^{k-1} i^2 + k^2 \\
&= \frac{2k^3 - 3k^2 + k}{6} + k^2 && \textbf{I.H.} \\
&= \frac{2k^3 - 3k^2 + k}{6} + \frac{6k^2}{6} \\
&= \frac{2k^3 + 3k^2 + k}{6} \\
&= \frac{k(2k^2 + 3k + 1)}{6} \\
&= \frac{k(k+1)(2k+1)}{6}.
\end{aligned}$$

□

4. Prove that given an unlimited supply of 6-cent coins, 10-cent coins, and 15-cent coins, one can make any amount of change larger than 29 cents.[1]

Solution:

Proof. Let c be the amount of change we can make. We have two cases: $29 < c \leq 35$, and $c > 35$.

Case 1: We begin with $c > 35$. We inductively assume that we can make $c - 6$ cents worth of change (which is greater than 29). Then we add a 6-cent coin to get $c = (c - 6) + 6$ cents in change.

Case 2: We now show it is true for $29 < p \leq 35$, by manually showing that each value p is a sum of 6's 10's, and 15's.

$c = 30 = 15 + 15$

$c = 31 = 6 + 10 + 15$

$c = 32 = 6 + 6 + 10 + 10$

$c = 33 = 6 + 6 + 6 + 15$

$c = 34 = 6 + 6 + 6 + 6 + 10 + 10$

$c = 35 = 10 + 10 + 15$

□

[1] This is problem 1 from Jeff Erickson's lecture notes on induction: `http://jeffe.cs.illinois.edu/teaching/algorithms/notes/98-induction.pdf`.