CSC 225 Assignment 2

Dryden Bryson

March 12, 2025

Question 1:

a)

We count the following assignments and comparisons:

- 1 Assignment on line 1: $\boxed{p \leftarrow 1}$
- For loop, 1 Assignment to initialize i: $\boxed{i \leftarrow 1}$ and 1 comparison to terminate loop
- Within the for loop, $n^2 + 1$ times:
  - 1 Comparison to check loop condition $\boxed{i \leftarrow 1 \text{ to } n^2 + 1}$ or $\boxed{i \leq n^2 + 1}$
  - 1 Assignment to increment i, $\boxed{i \leftarrow i + 1}$
  - 1 Assignment in body of loop $\boxed{p \leftarrow p \cdot i}$

This gives us, $1A + 1A + 1C$ comparisons and assigmnets which occur once and $1C + 1A + 1A$ which occur $n^2 + 1$ times in the for loop. This gives us the following:

$$T_{A\&C}(n) = 3 + \sum_{i=1}^{n^2+1} 3$$

b)

We count the following assignments and comparisons:

- 1 Assignment on line 1: $\boxed{s \leftarrow 1}$
- First for loop, 1 Assignment to initialize i: $\boxed{i \leftarrow 0}$ and one comparison upon loop termination
- Within first for loop, $n^2 + 1$ times:
  - 1 Comparison to check loop condition, $\boxed{i \leftarrow 1 \text{ to } n^2 + 1}$ or $\boxed{i \leq n^2 + 1}$
  - 1 Assignment to increment i, $\boxed{i \leftarrow i + 1}$
  - Nested for loop, 1 assignment to initialize j, $\boxed{j \leftarrow 1}$ and one comparison upon loop termination
  - Within nested loop i times,
    * 1 Comparison to check loop condition $\boxed{j \leftarrow 1}$ to i or $\boxed{j \leq i}$
    * 1 Assignment to increment j, $\boxed{j \leftarrow j + 1}$
    * 1 Assignment within body of loop: $\boxed{s \leftarrow s + i}$

This gives us, $1A + 1A + 1C$ assignments and comparisons which occur once and $1C + 1A + 1A + 1C$ within the first loop occuring $n^2 + 1$ times and within that loop $1C + 1A + 1A$ which occur i times. This gives us the following:

$$T_{A\&C}(n) = 3 + \sum_{i=0}^{n^2+1} \left[4 + \sum_{j=1}^{i} 3 \right]$$

or equivalently, avoiding nested summations:

$$T_{A\&C}(n) = 3 + \sum_{i=0}^{n^2+1} \left[4 + 3i \right]$$

Question 2:

The following algorithm recursively finds the minumum and the maximum of an array:

```
Algorithm findMinMax(A,n)
    Input: An array A storing atleast 1 real number, length of array denoted by n
    Output: A pair (min,max) containing the minimum and maximum values of the array

    if n = 1 then           //if array only contains one item
        return (A[0],A[0]) //return only element as min and max
    else
        (min, max) = findMinMax(A,n-1)

        if A[n - 1] < min then
            min = A[n - 1]
        else if A[n - 1] > max then
            max = A[n - 1]

        return (min,max)
```

For a worst case analysis, we will asume with each recursive iteration finds a new max so the first if statement is false resulting in 2 comparisons. We count the following assignments and comparisons:

- Base Case $n = 1$
  - 1 Comparison for $\boxed{n = 1}$
  - 1 Assignment for $\boxed{\text{return(A[0],A[0])}}$
  - **Total 2 Assignments and Comparisons**
- Recursive case $n > 1$
  - 1 Comparison for $\boxed{n = 1}$
  - 2 Assignments and $+T(n-1)$ for $\boxed{\text{(min , max) = findMinMax (A ,n -1)}}$
  - 1 Comparison for $\boxed{\text{if A [n - 1]} < \text{min then}}$
  - 1 Comparison for $\boxed{\text{else if A [n - 1]} > \text{max then}}$
  - 1 Assignment for $\boxed{\text{min = A [n - 1]}}$ or $\boxed{\text{max = A [n - 1]}}$ because of if-else flow
  - 1 Assignment for $\boxed{\text{return (min , max)}}$

Thus the recurrence equation counting the number of assignments and comparison is as follows:

$$T(n) = \begin{cases} 2 & \text{if } n = 1 \\ T(n-1) + 7 & \text{if } n > 1 \end{cases}$$

Question 3:

Let us first define the following for simplicity:

- $T(n-1) = T(n-2) + 2^{n-1}$
- $T(n-2) = T(n-3) + 2^{n-2}$
- $T(n-3) = T(n-4) + 2^{n-3}$

Then we can proceed with the definition of $T(n)$ and make repeated top down substitutions:

$$\begin{aligned} T(n) &= T(n-1) + 2^n \\ &= T(n-2) + 2^{n-1} + 2^n \\ &= T(n-3) + 2^{n-2} + 2^{n-1} + 2^n \\ &= T(n-4) + 2^{n-3} + 2^{n-2} + 2^{n-1} + 2^n \end{aligned}$$

Thus we can see the pattern that arrises which yeilds the reccurence:

$$T(n) = T(0) + 2^1 + 2^2 + 2^3 + \cdots + 2^n$$

When we apply the base case $T(0) = 1$ the equation becomes:

$$T(n) = 1 + \sum_{i=1}^{n} 2^i$$

Since $\sum_{i=1}^{n} 2^i$ is a geometric series it can be represented by the closed form $\frac{2(2^n-1)}{2-1}$ which simplifies to $2(2^n - 1)$ Thus we have that:

$$T(n) = 1 + 2(2^n - 1)$$

Question 4:

i. **Basis**
We want to show that the statement holds when $n = 1$, that is that the given formula we aim to prove yeilds 1 when $n = 1$ since $T(1) = 1$ is given, when $n = 1$:

$$\frac{1(1+1)}{2} = \frac{2}{2} = 1$$

Thus the base case holds.

ii. **Inductive Hypothesis**
Assume that the frmula holds true for $n > 1$ i.e:

$$T(n) = \frac{n(n+1)}{2}$$

iii. **Inductive Step**
We want to show that the statement holds true for $n + 1$ so we have that: The above matches the formula we aim to

$$\begin{aligned} T(n+1) &= T(n) + (n+1) && \text{From the given reccurence relation} \\ &= \tfrac{n(n+1)}{2} + (n+1) && \text{From the Inductive Hypothesis} \\ &= \tfrac{n(n+1)}{2} + \tfrac{2(n+1)}{2} \\ &= \tfrac{n(n+1)+2(n+1)}{2} \\ &= \tfrac{(n+1)(n+2)}{2} && \text{Factor } (n+1) \end{aligned}$$

prove when $n + 1$ or that $T(n+1) = \frac{(n+1)(n+2)}{2}$.

iv. **Conclusion**
By induction, $T(n) = \frac{n(n+1)}{2}$ holds for all $n \geq 1$

Question 5:

Loop Invariant Proof:
Loop Invariant: Throughout the loop it is true that at all times the variable sum holds the sum of the first k elements of the array.

i. **Initialization**
   Before the loop starts $k = 1$, and sum $\leftarrow$ A[0], which is indeed the sum of the first $k = 1$ elements of A thus the loop invariant holds.

ii. **Maintenance**
   We assume that the invariant holds at the start of an interation, that is that sum holds the sum of the first $k - 1$ elements. During the loop the value of the k'th element of A is added to sum so the loop invariant holds since sum now stores the sum of the first k elements.

iii. **Termination**
   The loop terminantes when $k = n - 1$ or once all of the elements of A have been indexed by the loop and added to the sum variable, thus sum holds the sum of the first k elements of the array which at this point is all of the elements in the array, sum is then returned which is exactly the desired output of the algorithm

The loop invariant holds true before, during and after every iterations of the loop. Therefore, the algorithm arraySum(A,n) correctly computes the sum of the elements of the array A. $\square$