1) Prove that the function $f(n) = n$ is a bound function for this recursive algorithm.

Proof: Consider the algorithm sCarin, and the function $f(n) = n$.

1) Since n is an integer input and $f(n) = n$, then the output is also the integer n, so this is an integer valued function.

2) The only time this algorithm is applied recursively are in steps 10 and 11. In step 10, there are three recursive calls of this algorithm with inputs $n-1$, $n-3$ and $n-4$ respectively. In step 11, there are four recursive calls of this algorithm with inputs $n-1$, $n-2$, $n-3$ and $n-4$ respectively.

It can now be seen that in both steps 10 and 11, the value of the function decreases by at least one every time the algorithm is executed recursively.

Hilroy

3) The precondition of this computational problem being solved by this algorithm requires that n is an integer such that $n \geq 0$. So the only case when $n \leq 0$ is when $n = 0$.

Then when the algorithm is executed with integer input $n = 0$, the test at step 0 is passed and the execution terminates after executing line 2. So the algorithm did not call itself recursively during this execution.

Thus, since these three conditions are met, this algorithm is a .bound function for the recursive algorithm sCarin.

- Citation (APA Format):

Eberly, W. (2019). CPSC 331: Data Structures, Algorithms and their analysis, Week 1, Lecture 2 Notes [Powerpoint Slides]

2) Prove that the sAvin algorithm correctly solves the "Avindevaled Aaggle computation" problem.

Claim: Suppose that n is a nonnegative integer and that the sAvin algorithm is executed with n as input. Then the execution of the algorithm eventually terminates with a_n being returned when that happens.

Proof: This claim will be proved by strong induction on n. The cases when $n=0$, $n=1$, $n=2$ and $n=3$ will be considered in the basis.

Basis: If $n=0$, then during the execution of the algorithm on input n, the test at step 1 succeeds and the value $1 = a_0 = a_n$ is returned as output, as required for this case.

If $n=1$, then when the algorithm is executed on input n, the test at step 2 fails. Step 3 is passed, and the value $2 = a_1 = a_n$ is returned as output in step 4, as required for this case

Milroy

If $n = 2$, then when the algorithm is executed on input n, the tests at step 1 and 3 are failed, and the test at step 5 is passed. At step 6, the value $3 = G_2 = G_n$ is returned as output, as required for this case.

If $n = 3$, then when the algorithm is executed on input n, the tests at step 1, step 3 and step 5 are failed, and the test at step 7 is passed. At step 8, the value $4 = G_3 = G_n$ is returned as output, as required for this case.

Inductive Step: Let $k \geq 3$ be an arbitrary integer. It is necessary and sufficient to use the following inductive hypothesis to prove the following inductive step:

> Inductive Hypothesis: Suppose m is a nonnegative integer such that $0 \leq m \leq k$. If the sGrin algorithm is executed with m as input, then this execution of the algorithm eventually ends with G_m returned as output.

Inductive Claim: If the Skrin algorithm is executed with $k+1$ as input, then this execution of the algorithm eventually terminates with G_{k+1} returned as output.

Now, Suppose that the Skrin algorithm is executed with $k+1$ as input. Since $k \geq 3$, then $k+1 \geq 4$. It also means that $k-1 \geq 2$, $k-2 \geq 1$, $k-3 \geq 0$, so the integers k, $k-1$, $k-2$ and $k-3$ are between 0 and k.

Since $k+1 \geq 4$, the tests at steps 1, 3, 5 and 7 fail, so the execution moves to step 9.

Note that with $k+1 \geq 4$, it means that $k+1$ can be either even or odd, so we have two different cases:

Case 1) Suppose that $k+1$ is even.

Then $k+1$ mod 2 will equal 0, so the test at step 9 will be passed and the execution will move on to step 10

Step 10 includes 3 recursive executions of the algorithm with each of their respective inputs being: Hilroy

$k+1-1 = k$, $k+1-3 = k-2$, and $k+1-4 = k-3$.

As noted before, k, $k-2$ and $k-3$ are all between 0 and k, so according to the inductive hypothesis, these three recursive executions of the algorithm eventually end with a_k, a_{k-2} and a_{k-3} returned as each of their respective outputs.

Now with all of the recursive executions terminated at line 10, we can see that this execution of the algorithm eventually ends.

The value being returned as output is

$2a_k - 2a_{k-2} + a_{k-3} = a_{k+1}$, as required to establish the inductive claim for this case.

Case 2) $k+1$ is odd.

If $k+1$ is odd, then $k+1$ mod 2 will not equal 0. So the test at step 9 will be failed, and the execution will move on to step 11.

Step 11 includes four recursive executions of the algorithm with their respective outputs being: $k+1-1 = k$, $k+1-2 = k-1$, $k+1-3 = k-2$, $k+1-4 = k-3$. As noted before, k, $k-1$, $k-2$, $k-3$ are all between 0 and k, so according to the inductive hypothesis, these four recursive executions of the algorithm eventually end with a_k, a_{k-1}, a_{k-2} and a_{k-3} being returned as each of their respective outputs.

With all the recursive executions terminated at line 10, we can see that the execution of the algorithm eventually ends. The value returned as output is $a_k + 3a_{k-1} - 5a_{k-2} + 2a_{k-3} = a_{k+1}$, as needed to establish the inductive claim for this case.

Since we have established the inductive claim in both of the cases, it means that we have managed to complete the inductive step, thus proving this claim by strong induction on n.

Hilroy

Citation (APA Format):

- Ebery, U. (2019). CPSC 331: Data structures, Algorithms, and their analysis. Week 1, Lecture 2 Notes [Powerpoint slides]

* Certain parts of the answer has used paraphrased parts from this source.

4) We are assuming that the uniform cost criterion and that the only steps counted is the number of steps shown in Figure 1.

- The algorithm executes two steps (steps 1 and 2) when executed with input $n = 0$.
- The algorithm executes three steps (steps 1, 3, 4) when executed with input $n = 1$.
- The algorithm executes four steps (steps 1, 3, 5, 6) when executed with input $n = 2$
- The algorithm executes five steps (steps 1, 3, 5, 7, 8) when executed with input $n = 3$
- The algorithm executes Six steps (steps 1, 3, 5, 7, 9, 10) if it is executed when n is even and $n \geq 4$, but there are also recursive executions at line 10:

1) With input $n-1$
2) With input $n-3$
3) With input $n-4$

Hilroy

- The algorithm executes six steps (steps 1, 3, 5, 7, 9, 11) if it is executed when n is odd and $n \geq 4$, but there are also recursive executions at step 11:

1) with input $n-1$
2) with input $n-2$
3) with input $n-3$
4) with input $n-4$

Thus, $T_{savin}(n)$, which is the number of steps executed by this algorithm savin on a non-negative integer n, is given by the following recurrence:

$$T_{savin}(n) = \begin{cases} 2 & \text{if } n = 0 \\ 3 & \text{if } n = 1 \\ 4 & \text{if } n = 2 \\ 5 & \text{if } n = 3 \\ T_{savin}(n-1) + T_{savin}(n-3) + T_{savin}(n-4) + 6 & \text{if } n \geq 4 \text{ and } n \bmod 2 \text{ equals } 0 \\ T_{savin}(n-1) + T_{savin}(n-2) + T_{savin}(n-3) + T_{savin}(n-4) + 6 & \text{if } n \geq 4 \text{ and } n \bmod 2 \text{ does not equal } 0. \end{cases}$$

Citation (APA format):
Ebeyly, U. (2019). CPSC 331: Data Structures, Algorithms and their Analysis, Week 2, Lecture 5 Notes [Powerpoint Slides]

5) Claim: Suppose that T_{savin} is a function of the non negative integers such that for every integer $n \geq 0$,

$$T_{savin}(n) = \begin{cases} 2 & \text{if } n = 0 \\ 3 & \text{if } n = 1 \\ 4 & \text{if } n = 2 \\ 5 & \text{if } n = 3 \\ T_{savin}(n-1) + T_{savin}(n-3) + T_{savin}(n-4) + 6 & \text{if } n \geq 4 \text{ and } n \bmod 2 \text{ equals } 0 \\ T_{savin}(n-1) + T_{savin}(n-2) + T_{savin}(n-3) + T_{savin}(n-4) + 6 & \text{if } n \geq 4 \text{ and } n \bmod 2 \text{ does not equal } 0. \end{cases}$$

Then $T_{savin}(n) \geq \left(\frac{3}{2}\right)^n$ for every non-negative integer n.

Proof: We will prove this claim by strong induction on n. The cases when $n = 0$, $n = 1$, $n = 2$, $n = 3$ will be considered in the basis.

Basis: If $n = 0$, then $T_{savin}(n) = T_{savin}(0)$

$= 2$ (by the definition of T_{savin})

≥ 1

$= \left(\frac{3}{2}\right)^0$

$= \left(\frac{3}{2}\right)^n$ (because $n = 0$),

as required for this case.

Milbrey

Now if $n=1$, then $T_{savin}(n) = T_{savin}(1)$

$= 3$ (by definition of T_{savin})

$= \frac{6}{2}$

$\geq \frac{3}{2}$

$= \left(\frac{3}{2}\right)^1 = \left(\frac{3}{2}\right)^n$ (since $n=1$),

as required for this case

Now if $n=2$, then $T_{savin}(n) = T_{savin}(2)$

$= 4$ (by definition of T_{savin})

$= \frac{16}{4}$

$\geq \frac{9}{4}$ (since $9 \leq 16$)

$= \left(\frac{3}{2}\right)^2 = \left(\frac{3}{2}\right)^n$ (since $n=2$),

as required for this case

Now if $n=3$, then $T_{savin}(n) = T_{savin}(3)$

$= 5$ (by definition of T_{savin})

$= \frac{40}{8}$

$\geq \frac{27}{8}$ (since $40 \geq 27$)

$= \left(\frac{3}{2}\right)^3 = \left(\frac{3}{2}\right)^n$ (since $n=3$),

as required for this case.

Inductive Step: Let k be an integer such that $k \geq 3$. It is necessary and sufficient to use the following

Inductive Step: $T_{savin}(m) \geq \left(\frac{3}{2}\right)^m$ for all integers m such that $0 \leq m \leq k$.

to prove the following

Inductive Hypothesis: $T_{savin}(k+1) \geq \left(\frac{3}{2}\right)^{k+1}$

Note that since $k \geq 3$, then $k-1 \geq 2$, $k-2 \geq 1$, $k-3 \geq 0$, so k, $k-1$, $k-2$ and $k-3$ are all between 0 and k.

Also note that there are two cases: 1) $k+1$ is even, 2) $k+1$ is odd.

Case 1: $k+1$ is even.

Then since $k \geq 3$, then $k+1 \geq 4$, so it follows by the definition of $T_{savin}(n)$ that

$$T_{savin}(k+1) = T_{savin}(k) + T_{savin}(k-2) + T_{savin}(k-3) + 6$$

As noted before k, $k-2$, $k-3$ are all between 0 and k.

So using the inductive hypothesis, we get

$$T_{savin}(k+1) \geq \left(\frac{3}{2}\right)^k + \left(\frac{3}{2}\right)^{k-2} + \left(\frac{3}{2}\right)^{k-3} + 6$$

Nilroy

$= \left(\frac{27}{8}\right)\left(\frac{3}{2}\right)^{k-3} + \frac{3}{2}\left(\frac{3}{2}\right)^{k-3} + \left(\frac{3}{2}\right)^{k-3} + 6$ (by factoring out $\frac{3}{2}$)

$= \left[\left(\frac{27}{8}\right) + \left(\frac{12}{8}\right) + \left(\frac{8}{8}\right)\right]\left(\frac{3}{2}\right)^{k-3} + 6$

$= \frac{47}{8}\left(\frac{3}{2}\right)^{k-3} + \frac{48}{8}$

$= \frac{95}{8}\left(\left(\frac{3}{2}\right)^{k-3} + 1\right)$

$= \frac{190}{16}\left(\left(\frac{3}{2}\right)^{k-3} + 1\right)$

$\geq \frac{190}{16}\left(\frac{3}{2}\right)^{k-3}$ (because $\left(\frac{3}{2}\right)^{k-3} + 1 \geq \left(\frac{3}{2}\right)^{k-3}$ for all integers $k \geq 3$)

$\geq \frac{81}{16}\left(\frac{3}{2}\right)^{k-3}$ (because $190 \geq 81$)

$= \left(\frac{3}{2}\right)^{4}\left(\frac{3}{2}\right)^{k-3}$

$= \left(\frac{3}{2}\right)^{k+1}$, as required to establish the inductive claim in this case.

Case 2: Now suppose $k+1$ is odd.

Then since $k \geq 3$, then $k \geq 4$, so it follows by the definition of $T_{\text{Savitch}}(n)$ that

$$T_{\text{Savitch}}(k+1) = T_{\text{Savitch}}(k) + T_{\text{Savitch}}(k-1) + T_{\text{Savitch}}(k-2) + T_{\text{Savitch}}(k-3) + 6$$

As noted before, k, $k-1$, $k-2$, $k-3$ are all between 0 and k.

So using the inductive hypothesis, we get

$$T_{\text{Savitch}} \geq \left(\frac{3}{2}\right)^{k} + \left(\frac{3}{2}\right)^{k-1} + \left(\frac{3}{2}\right)^{k-2} + \left(\frac{3}{2}\right)^{k-3} + 6$$

$$= \frac{27}{8}\left(\frac{3}{2}\right)^{k-3} + \frac{9}{4}\left(\frac{3}{2}\right)^{k-3} + \frac{3}{2}\left(\frac{3}{2}\right)^{k-3} + \left(\frac{3}{2}\right)^{k-3} + 6$$

$$= \left(\frac{27}{8} + \frac{18}{8} + \frac{12}{8} + \frac{8}{8}\right)\left(\frac{3}{2}\right)^{k-3} + 6$$

$$= \frac{55}{8}\left(\frac{3}{2}\right)^{k-3} + 6$$

$$= \left(\frac{55}{8} + \frac{48}{8}\right)\left(\left(\frac{3}{2}\right)^{k-3} + 1\right)$$

$$= \left(\frac{103}{8}\right)\left(\left(\frac{3}{2}\right)^{k-3} + 1\right)$$

$$= \left(\frac{206}{16}\right)\left(\left(\frac{3}{2}\right)^{k-3} + 1\right)$$

$$\geq \left(\frac{206}{16}\right)\left(\left(\frac{3}{2}\right)^{k-3}\right) \qquad \text{(because } \left(\frac{3}{2}\right)^{k-3} + 1 \geq \left(\frac{3}{2}\right)^{k-3} \text{ for all integers } k \geq 3\text{)}$$

$$\geq \left(\frac{81}{16}\right)\left(\frac{3}{2}\right)^{k-3} \qquad \text{(because } 206 \geq 81\text{)}$$

$$= \left(\frac{3}{2}\right)^{4}\left(\frac{3}{2}\right)^{k-3}$$

$$= \left(\frac{3}{2}\right)^{4+k-3}$$

$$= \left(\frac{3}{2}\right)^{k+1}$$, as required to establish the inductive claim in this case.

Thus, since the inductive claim has been established in both case 1 and case 2, it means that the inductive step has been completed and so we have proved this claim by strong induction.

Citation (APA format):

Eberly, L. (2019). CPSC 331: Data Structures, Algorithms and their Analysis, week 2, Lecture 5 notes [Powerpoint Slides] Hilroy

6) Consider the algorithm fGrin for the "Grindelwald Gaggle Computation Problem" and the while loop at lines 15-19 of this algorithm.

The following is a loop invariant of this while loop:
Loop Invariant:
1) n is an integer input such that $n \geq 4$

2) G is a (variable) integer array with length $n+1$

3) i is an integer variable such that $4 \leq i \leq n+1$

4) $G[j] = G_j$ for every integer j such that $0 \leq j \leq i-1$

Hilroy

7) Claim #1: Consider the algorithm fGvin for the "Grindelwald Gaggle Computation Problem" and the while loop at lines 15-19 of this algorithm. The following is a loop invariant for this while loop:

Loop Invariant:

1) n is an integer input such that $n \geq 4$

2) G is a (variable) integer array with length $n+1$

3) i is an integer variable such that $4 \leq i \leq n+1$

4) $G[j] = G_j$ for every integer j such that $0 \leq j \leq i-1$

Proof: We will prove this claim using loop theorem #1.

Loop theorem #1: Consider an algorithm for a given computational problem with a while loop: while (t) {
S
}
and an assertion A. If

a) an execution of the loop test does not change the value of any inputs, variables or global data (no side effects)

b) If the algorithm is executed when the problem's precondition is satisfied, then the assertion is satisfied whenever the loop is reached.

c) If this assertion is satisfied at the beginning of any execution of the

Hilroy

(assuming the precondition is satisfied)
loop body S, then the assertion is also satisfied when the execution of the loop body ends.

Then A is a loop invariant for this while loop.

Proof of a): Note that the loop test at step 7 is an inequality statement, so it does not change the value of any inputs, variables or global data (so no side effects).

Proof of b): To prove b) we will state and prove Claim # 2, which establishes that condition b) is satisfied.

Claim # 2: If the algorithm favin is executed with the problem's precondition satisfied, then this assertion is satisfied whenever the loop is reached.

Proof of Claim # 2: Consider an execution of this algorithm when it's problem's precondition is satisfied - that is, n is an integer input such that $n \geq 0$.

When $n = 0$ or $n = 1$ or $n = 2$ or $n = 3$, then the execution of the algorithm ends before the while loop is reached, so the claim

is trivial and satisfied in these cases.

Now, consider $n \geq 4$.

Since there is no step in the algorithm where the value of n is changed, then part 1) of the assertion holds when the while loop is reached during an execution of the algorithm.

Then after the execution of steps 1, 3, 5, 7, 9-14, the while loop is reached. One can see by inspection of the code that the while loop is never reached again, so it is sufficient to just consider this execution of the loop.

At step 9, a is declared to be a (variable) integer array with length $n+1$. Since steps 10 - 14 do not change the value of the length of a, it means that part 2 of the assertion holds when the while loop is reached.

Now, since i is declared to be an integer variable with value of 4 at step 14 and $n \geq 4$, then $n+1 \geq 5$, so it follows that $4 \leq i \leq n+1$. So part 3) of the assertion holds when the loop is reached.

Hilroy

Now, since steps 10 - 14 is executed before the while loop is reached, it means that i is an integer variable with value 4 and $a[0] = 1 = a_0$, $a[1] = 2 = a_1$, $a[2] = 3 = a_2$, $a[3] = 4 = a_3$, so $a[j] = a_j$ for every integer j such that $0 \leq j \leq i-1$. So part 4) of the assertion holds when the loop is reached, as needed to complete the proof of the claim.

Since we have proved claim #2, we have also proved condition b) of loop theorem #1.

Proof of c): To prove c) we will state and prove claim #3, which establishes that condition c) is satisfied.

Claim #3: If this assertion is satisfied at the beginning of any execution of the body of the while loop (assuming the precondition is satisfied), then it is also satisfied when the execution of the loop body ends.

Proof of claim #3: Consider an execution of the body of the while loop of the fawin algorithm. Suppose that the assertion of claim #1 is satisfied when the execution of the loop body begins.

Then at the beginning of this execution of the loop body, n is an integer input such that $n \geq 4$, as stated in part 1) of the assertion. Since there is no step within the body of the loop that changes the value of n, it means that n is still an integer variable such that $n \geq 4$ at the end of the body of the loop, so part a of the assertion holds at the end of this execution of the loop body.

According to the second part of the assertion, A is a (variable) integer array of length n+1 at the beginning of the execution of the while loop body. By inspecting the code, we can see that the length or type of A is not modified during the loop test at step 15 or the loop body (which is composed of steps 16-19). So F is still a (variable) integer array of length n+1 at the end of the execution of the loop body, so part 2) of the assertion holds at the end of the execution of the loop body.

According to the third part of the assertion, i is an integer variable such that $4 \leq i \leq n+1$ at the beginning of the execution of the loop

Hilroy

body. Now the loop test at step 15 must have been passed in order to reach the loop body, so we know that $i \leq n$. Then since $4 \leq i \leq n+1$ and $i \leq n$, it follows that $4 \leq i \leq n$ at the beginning of the loop body execution. This is also true from steps 16-18, where neither i or n is modified in any way. At step 19, the value of i is modified by 1, so it follows that at the end of step 9 (which is the last step in the loop body), i is an integer variable such that $4 \leq i \leq n+1$. So part 3) of the assertion holds at the end of the execution of the body of the while loop.

According to part 4) of the assertion, $F[j] = F_j$ for every integer j such that $0 \leq j \leq i-1$ at the beginning of this execution of the loop body. Also, as noted above, $4 \leq i \leq n$ at this point of the execution. So G[i] will have the value

$$G[i] = 2 \times G[i-1] - 2 \times G[i-3] + G[i-4]$$

$$= 2G_{i-1} - 2G_{i-3} + G_{i-3} = G_i$$

if i mod 2 is equal to 0 and step 17 is executed.

And if i mod 2 is not equal to 0, then step 18 is executed and

G[i] will have the value

$$G[i] = G[i-1] + 3G[i-2] - 5G[i-3] + 2G[i-4]$$

$$= G_i + 3G_{i-1} - 5G_{i-2} + 2G_{i-3} = G_{i+1}$$

So after step 17 or 18, $G[j] = G_j$ for every integer j such that $0 \leq j \leq i$

In step 19, the value of i is increased by 1, so now $F[j] = F_j$ for every integer j such that $0 \leq j \leq i-1$

Thus, all of the assertions in claim #1 is satisfied at the end of the execution of the loop body, as needed to complete the proof of claim #3.

Since we have proved claim #3, we have also proved condition b) of loop theorem #1.

Thus, since we have satisfied conditions a), b) and c) of the loop theorem #1, we can conclude that claim #1, and the stated loop invariant, are both correct.

* Certain parts of the answer has used paraphrased parts from this source:
Citation: (APA format):
Eberly, W. (2019). CPSC 331: Data Structures, Algorithms, and their analysis, Week 2, Lecture 3 notes [Powerpoint Slides]

8) Claim: The fGrin algorithm is partially correct, when considered as an algorithm for the "Grindelwald Gaggle Computation" problem.

Proof: Consider an execution of the algorithm with the precondition for the "Grindelwald Gaggle Computation" problem being satisfied. Then n is a non-negative integer input.

In order to prove partial correctness, we have to show that either

a) The execution of the algorithm terminates and the post condition being satisfied when that happens (with no undocumented inputs accessed or modified), or

b) the execution of the algorithm never ends at all

By inspecting the signature and each step in this algorithm, we can see that no undocumented inputs or global data are accessed or modified, so there are no "side effects"

Now if $n = 0$, then the test at step 1 is passed and the value $1 = G_0 = G_n$ returned as output, as required to satisfy condition a) for this case.

Hilroy

If $n = 1$, then the test at step 1 is failed, and the execution moves on and passes the test at step 2, with the value $2 = G_1 = G_n$ being returned as output. This satisfies condition a) in this case.

If $n = 2$, then the tests at step 1 and 3 are failed, and the execution moves on and passes the test at step 5, with the value $3 = G_2 = G_n$ returned as output. This satisfies condition b) in this case.

If $n = 3$, then the tests at step 1, step 3 and step 5 are failed, so the execution moves on and passes the test at step 5, with the value $4 = G_3 = G_n$ being returned as output. This satisfies condition b) in this case.

Now if $n \geq 4$, then steps 1, 3, 5, 7, 9 - 14 are executed and the loop is reached and executed in steps 15 - 18.

We now have 2 cases: 1) The loop terminates, 2) The loop does not terminate.

Case 1: If the execution of the loop terminates, and the loop invariant (which we stated and proved in parts 6 and 7) is

satisfied at that point, then the following is true: n is an integer input such that $n \geq 4$, a is a (variable) integer array with length $n+1$, i is an integer variable such that $4 \leq i \leq n+1$ and $a[j] = a_j$ for every integer j such that $0 \leq j \leq i-1$

In order for the execution of the while loop to have terminated, the loop test at line 15 must have been checked and failed. Then by the end of the loop execution, $i \geq n+1$, and to be more precise, $i = n+1$. So since $i = n+1$, it means that $a[j] = a_j$ for every integer j such that $0 \leq j \leq (n+1)-1$, or $0 \leq j \leq n$.

After the execution of the loop ends, at step 20 the execution of the algorithm ends with the value $a[n] = a_n$ returned as output, which satisfies the postcondition and thus satisfies condition a).

Case 2: Now if the execution of the loop never terminates, then the execution of the algorithm does not end either, so Hilroy

condition b) has been satisfied in this case

Since either condition a) or condition b) is satisfied in every possible case, we can conclude that this algorithm is partially correct, as claimed.

Citation (APA format):

- Eberly, W. (2019). CPSC 331: Data Structures, Algorithms and their Analysis, Week 2, Lecture 5 notes [PowerPoint Slides]
- Eberly, W. (2019). CPSC 331: Data Structures, Algorithms and their Analysis, Week 2, Tutorial Exercise #4 solutions [PDF Document]

* Certain parts of the answer has used paraphrased parts from the second source.

9) Claim: The function $f(n,i) = (n+1) - i$ is a bound function for the while loop in the fawin algorithm

Proof:

Since n is an integer input and i is an integer variable defined and initialized at step 14, this is certainly an integer valued total function on some of the inputs and variables that are defined when the loop is reached.

When the loop body is executed, the value of i increases by 1 at step 19 and the value of n does not change at all. So the value of the function $f(n,i) = (n+1) - i$ decreases by (at least) one each time the loop body is executed.

Suppose the value of f is less than or equal to zero. Then it would mean that $(n+1) - i \leq 0$. This in turn would mean that $i \geq n+1$. Note that when $i \geq n+1$, then the loop test, which requires that $i < n+1$, would fail and the execution of the loop would end.

Hilroy

Thus, the function f is a bound function for the while loop in an algorithm because it satisfies all the required properties of one.

So we have proved the claim.

Citation (APA format):

- Eberly, W. (2019). CPSC 331: Data Structures, Algorithms and their Analysis, Week 2, Lecture 3 Notes [Powerpoint Slides]
- Eberly, W. (2019). CPSC 331: Data Structures, Algorithms and their Analysis, Week 3, Tutorial Exercise 4 Answers [PDF Document]

* Certain parts of the answer has used paraphrased parts from the second source.

10) Claim: If the fGrin algorithm is executed when the precondition for the "Grindelwald Gaggle Computation" problem is satisfied, and the while loop is reached and executed, then the execution of the loop eventually ends.

Proof: We will prove this using Loop theorem # 2

Loop Theorem # 2: Consider a while loop

```
while (t)
do {S}
```

with a loop test t and a loop body S. If the following properties are satisfied:

a) an execution of the loop test t has no side effects - so no change in the value of any inputs, variables or global data. Also, every execution of this loop test terminates.

b) If the problem's precondition is satisfied when an execution of the algorithm including this loop body begins, then every execution of the loop body ends

c) A bound function for this while loop exists

Hilroy

If all three conditions are satisfied, then every execution of this while loop, included in an execution of the algorithm with the problem's precondition satisfied, terminates.

Also, the value of the bound function immediately before the execution of this loop is an upper bound of the number of times that the loop body is executed before this execution of the loop ends.

Back to proof of claim:

Suppose the algorithm fAvin is executed with the precondition of the problem satisfied. Then n is an integer input such that $n \geq 0$.

If $n = 0$ then the test at step 1 is passed and the execution of the algorithm ends after executing step 2, as required to establish the claim.

If $n = 1$ then the test at step 1 is failed, and so the execution moves on and passes the test at step 3 and it terminates after executing step 4, as required to establish the claim

If $n = 2$ then the tests at steps 1 and 3 are failed, so the execution moves on and passes the test at step 5 and it terminates after executing step 6, as required to establish the claim.

If $n = 3$ then the tests at steps 1, 3 and 5 are failed, so the execution moves on and passes the test at step 7, and then it terminates after executing step 8, as required to establish the claim.

Now, if $n \geq 3$ then the tests at steps 1, 3, 5 and 7 are failed and the steps 9 - 14 are executed before the while loop is reached. After that, the while loop is reached and executed

- It is clear upon inspection that since the loop test at line 15 is only comprised of a comparison between an integer and variable ($i \leq n$), the loop test does not change the value of any inputs, variables, global data. Moreover, every execution of the loop test certainly halts

- The body of the while loop is comprised of one if statement (at step 16) and two assignment statements (at steps 17, 18, 19), so every execution of the loop body will terminate.

- As noted in part 9) of the assignment, this while loop has a bound function $f(n,i) = (n+1) - i$

Since all the conditions of loop theorem #2 has been satisfied, it now follows that if the algorithm is executed with the precondition satisfied, and the while loop is reached and executed, then the execution of the loop will eventually terminate, as needed to establish the claim.

Also, using the loop theorem #2, we can establish $(n+i) - 4$ to be an upper bound for the number of times the loop body is executed before the execution of the loop ends.

Citation (APA Format):

Eberly, W. (2019). CPSC 331: Data Structures, Algorithms and their Analysis, Week 2, Lecture 3 Notes [Powerpoint slides]

11) Claim: The algorithm fGvin is correct.

Proof: In part 8) of the assignment, we proved that the algorithm fGvin is partially correct. That means the execution of this algorithm either:

1) ends with the problem's postcondition satisfied and no side effects

2) never ends at all.

In part 10) of the assignment, we proved that if the algorithm is executed (with precondition satisfied) and the while loop is reached, then the execution of the loop eventually ends.

Note that after the execution of the loop ends, the execution of the algorithm itself ends after executing line 20.

So it now follows that if the problem's precondition is satisfied and the algorithm is executed, then the execution of the algorithm ends with the postcondition satisfied (and with no side effects), as needed to prove the correctness of the fGvin algorithm.

Thus, we have managed to prove the claim. Hilroy

12) Suppose this algorithm is executed with non-negative integer n as input.

We are assuming the uniform cost criterion and counting only the numbered steps.

If n = 0 then steps 1 and 2 are executed and the algorithm ends. Since two steps have been executed, $T_{\text{fwin}}(0) = 2$.

If n = 1 then steps 1, 3 and 4 are executed and the algorithm ends. Since three steps have been executed, $T_{\text{fwin}}(1) = 3$.

If n = 2 then steps 1, 3, 5 and 6 are executed and the algorithm ends. Since four steps have been executed, $T_{\text{fwin}}(2) = 4$.

If n = 3 then steps 1, 3, 5, 7 and 8 are executed and the algorithm ends. Since five steps have been executed, $T_{\text{fwin}}(3) = 5$.

Now, suppose that $n \geq 4$.

- Then the steps 1, 3, 5, 7, 9-14 are executed before the while loop is reached. So 10 steps are executed before the while loop is reached.

Hilroy

- From part 9 of this assignment, we know that $(n+1)-i$ is a bound function for the while loop in this algorithm, which has initial value $n+1-4 = n-3$

- As noted in part 9 of this assignment, the loop body will be executed at most $n-3$ times, and the loop test will be executed at most $n-2$ times.

- If i is even, the loop body will execute steps 16, 17, 19. And if i is odd, the loop body will execute steps 16, 18, 19. So in both cases, each execution of the loop body will consist of three steps.

- The loop test consists of only one step, which is step 15.

- So the total number of steps used in the execution of the while loop is at most:

$$\sum_{j=1}^{n-3} T_{body}(j) + \sum_{j=1}^{n-2} T_{test}(j) = \sum_{j=1}^{n-3} 3 + \sum_{j=1}^{n-2} 1$$

$$= 3(n-3) + n-2 = 3n-9+n-2$$

$$= 4n-11$$

- An additional step (step 20) is executed after the while loop ends

- So when $n \geq 4$, $T_{fact2in}(n) = 4n - 11 + 10 + 1$
$= 4n$

So the number of steps executed by the algorithm on input n is at most $T_{fact2in}(n)$, where

$$T_{fact2in}(n) = \begin{cases} 2 & \text{if } n = 0 \\ 3 & \text{if } n = 1 \\ 4 & \text{if } n = 2 \\ 5 & \text{if } n = 3 \\ 4n & \text{if } n \geq 4 \end{cases}$$

Citation (APA Format):

Eberly, W. (2019). CPSC 331: Data Structures, Algorithms and their Analysis, Week 2, Lecture 3 Notes [Powerpoint Slides]

Hilroy

14) Claim: $a_n = n+1$ for all integers $n \geq 0$

Proof: We will prove this by strong induction on n. The cases when $n=0$, $n=1$, $n=2$, $n=3$ will be considered in the basis.

Basis: If $n=0$, then

$$\begin{aligned} a_n &= a_0 \\ &= 1 \quad \text{(by the definition of } a_n\text{)} \\ &= 0+1 \\ &= n+1 \quad \text{(since } n=0\text{)}, \end{aligned}$$

as required for this case.

If $n=1$, then

$$\begin{aligned} a_n &= a_1 \\ &= 2 \quad \text{(by the definition of } a_n\text{)} \\ &= 1+1 \\ &= n+1 \quad \text{(since } n=1\text{)}, \end{aligned}$$

as required for this case.

If $n=2$, then

$$\begin{aligned} a_n &= a_2 \\ &= 3 \quad \text{(by the definition of } a_n\text{)} \\ &= 2+1 \\ &= n+1 \quad \text{(since } n=1\text{)}, \end{aligned}$$

as required for this case.

Hilroy

If $n=3$, then

$$\begin{aligned} a_n &= a_3 \\ &= 4 \qquad \text{(by the definition of } a_n\text{)} \\ &= 3+1 \\ &= n+1 \qquad \text{(since } n=3\text{)}, \end{aligned}$$

as required for this case.

Inductive Step: Let k be an integer such that $k \geq 3$. It is necessary and sufficient to use the following

Inductive Hypothesis: $a_m = m+1$ for all integers m such that $0 \leq m \leq k$

to prove the following

Inductive Claim: $a_{k+1} = k+2$

Now, note that since $k \geq 4$, then $k+1 \geq 5$.

Also note that since $k \geq 4$, then $k-1 \geq 3$, $k-2 \geq 2$, $k-3 \geq 1$, so k, $k-1$, $k-2$, and $k-3$ are all between 0 and k.

We also have 2 cases.

Case 1) $k+1$ is even.

Then since $k+1$ is even, and $k+1 \geq 5$, we can use the definition of a_{k+1} to get

$$a_{k+1} = 2a_k - 2a_{k-2} + a_{k-3}$$

As noted before, k and $k-2$ and $k-3$ are all between 0 and k, so we can use the inductive hypothesis to get

$$\begin{aligned} a_{k+1} &= 2(k+1) - 2(k-2+1) + (k-3+1) \\ &= 2k+2 - 2k+4-2 + k-2 \\ &= (k+2k-2k) + (4+2-2-2) \\ &= k+2, \text{ as required to establish the inductive claim for this case.} \end{aligned}$$

Case 2) $k+1$ is odd

Then since $k+1$ is odd and $k+1 \geq 5$, we can use the definition of a_{k+1} to get

$$a_{k+1} = a_k + 3a_{k-1} - 5a_{k-2} + 2a_{k-3}$$

As noted before, k, $k-1$, $k-2$ and $k-3$ are all between 0 and k, so we can use the inductive hypothesis to get

Hilroy

$$a_{k+1} = (k+1) + 3(k) - 5(k-1) + 2(k-2)$$

$$= k + 1 + 3k - 5k + 5 + 2k - 4$$

$$= (3k + 2k + k - 5k) + (5 + 1 - 4)$$

$= k + 2$, as required to establish the inductive claim in this case.

Since we have established the inductive claim in both cases, we have completed the inductive hypothesis.

Thus, $a_n = n + 1$ for all integers $n \geq 0$, so we have proved the claim.

- (No resources or notes were consulted when answering this question)