CPSC 331 — Solution for Question #4 on the Practice Midterm Test

This question also concerned the "Slytherin Number Computation" problem and the `cSlytherin` algorithm shown on the *second* page of the lecture supplement.

(a) You were first asked to give a ***bound function*** for the `while` loop in this algorithm and to give a ***brief*** proof that your answer is correct.

Bound Function for Loop: The function

$$f(\mathtt{n}, \mathtt{i}) = \mathtt{n} - \mathtt{i}$$

is a bound function for the `while` loop in this algorithm.

Brief Proof That This is a Bound Function: Note first that `n` is an input for this algorithm and `i` is a variable that has been defined before the loop is reached. Thus the above function f is a (well-defined) function of the inputs and variables in this algorithm.

It is therefore sufficient to prove that this functions satisfies all three of the properties included in the definition of a "bound function for a `while` loop."

- Since `n` is an integer input, `i` is an integer input, and the difference between two integers is an integer, this is an ***integer-valued*** function.
- If the body of this `while` loop is executed then the value of `n` is not changed, but the value of `i` is *increased* in value by one, when the step at line 12 is executed. Thus an execution of the body of the `while` loop ***decreases the value of this function by at least one.***
- Finally, suppose that the precondition for the "Slytherin Number Computation" problem is satisfied when this algorithm is executed, and the value of this function is less than or equal to zero at some point when the loop's test is checked. Then, since $f(\mathtt{n} - \mathtt{i}) = \mathtt{n} - \mathtt{i} \leq 0$, $\mathtt{n} \geq \mathtt{i}$, and the loop's test (shown at step 8) fails, ***causing the execution of the loop to terminate***.

It follows that the above function is a bound function for the `while` loop in this algorithm, as claimed.

Note: For full marks, a student's proof should include at least a few references to the specific computational problem being solved and the `while` loop in this algorithm, so that the student is not just repeating the definition of a "bound function for a a `while` loop" and claiming that it is satisfied.

(b) You were asked to use this to give an ***upper bound*** for the number of steps that are executed during an execution of the `while` loop in this algorithm, using the *uniform cost criterion* to define this. This should be a function of `n`.

Upper Bound for Running Time of Loop: *Loop Theorem #2* can be applied to conclude that the *initial value* of this function is an upper bound for the number of times that the body of the `while` loop is executed. Since `i` $= 1$ immediately before the execution of this loop begins, this upper bound is `n` $- 1$.

It follows that the loop's *test* (at line 8) is executed at most one more times than this, that is, at most `n` times.

A formula included in the lecture on bounding the cost of `while` loops can now be applied:

- For $1 \leq j \leq$ `n` let $T_{\mathsf{Test}}(j)$ be an upper bound for the cost of the j^{th} execution of the loop's test. One can see by examination of the code that (since this is a simple comparison) $T_{\mathsf{Test}}(j) = 1$.
- For $1 \leq j \leq$ `n` $- 1$, let $T_{\mathsf{Body}}(j)$ be an upper bound for the cost of the j^{th} execution of the *body* of the loop. One can see by inspection of the code that the loop body consists of four (reasonably simple) assignment statements, so that $T_{\mathsf{Body}}(j) = 4$.
- It now follows by an application of the formula given in the lecture notes that the number of steps included in an execution of this `while` loop is at most

$$\sum_{j=1}^{\mathtt{n}} T_{\mathsf{Test}}(j) + \sum_{j=1}^{\mathtt{n}-1} T_{\mathsf{Body}}(j) = \sum_{j=1}^{\mathtt{n}} 1 + \sum_{j=1}^{\mathtt{n}-1} 4 = \mathtt{n} + 4(\mathtt{n} - 1) = 5\mathtt{n} - 4.$$

(c) Finally, you were asked to used this to given an ***upper bound*** for the number of steps executed during an execution of this algorithm, using the *uniform cost criterion* to define this. This should also be a function of `n`.

Upper Bound for Running Time of Algorithm: Suppose that the precondition for the "Slytherin Number Computation" problem is satisfied, so that `n` is a nonnegative (input) integer. Let $T_{\mathtt{cSlytherin}}(\mathtt{n})$ be the number of steps executed by the `cSlytherin` algorithm on such an input `n`.

- If `n` $= 0$ then two steps — the steps at lines 1 and 2 — are executed when the algorithm is executed on input `n`. Thus $T_{\mathtt{cSlytherin}}(0) = 2$.
- If `n` $= 1$ then three steps — at lines 1, 3 and 4 — are executed when the algorithm is executed on input `n`. Thus $T_{\mathtt{cSlytherin}}(1) = 3$.
- If `n` ≥ 2 then six steps — the steps at lines 1, 3, 5–7 and 13 — are executed, *along with an execution of the* `while` *loop*, during an execution of the algorithm on input `n`. It follows by the result of the previous question that $T_{\mathtt{cSlytherin}}(\mathtt{n}) \leq 6 + (5\mathtt{n} - 4) = 5\mathtt{n} + 2$ in this case.

Thus if n is a nonnegative integer then

$$T_{\texttt{cSlytherin}} \leq \begin{cases} 2 & \text{if } \texttt{n} = 0, \\ 3 & \text{if } \texttt{n} = 1, \\ 5\texttt{n} + 2 & \text{if } \texttt{n} \geq 2. \end{cases}$$

Note: It follows by the remarks given at the end of the solution for Question #2 and the solution for part (c), above, that the `cSlytherin` algorithm is ***much*** faster than the `slytherin` algorithm, when the input n is large.