CIS121-204 – Fall 2007

Lab 2 Solution

Tuesday, September 18

Recall that when we analyze running time we analyze the *worst* case.
Count the exact number of steps for the following snips of pseudo-code and give a big-oh analysis.

Prints a square of '' of size* n+1.

PRINTSQUARE(n)

1 **for** $i \leftarrow 0$ **to** n $\quad\triangleright$ 1 initial assignment; $n+1$ increments; $n+2$ comparisons
2 $\quad$ **do for** $j \leftarrow 0$ **to** n $\quad\triangleright$ 1 initial assignment; $n+1$ increments; $n+2$ comparisons
3 $\quad\quad$ **do** PRINT $*$ $\quad\triangleright$ 1 operation
4 $\quad$ PRINT linebreak $\quad\triangleright$ 1 operation

For each inner **for** loop, line 3 executes 1 operation for $n+1$ times. Hence, the number of steps for one inner **for** loop is $1 + 2(n+1) + (n+2) + (n+1) = 4n + 6$. Note that each increment costs 2 steps: an addition and an assignment.

For each outer **for** loop, the inner **for** loop executes $4n + 6$ steps. Line 4 execute 1 operation. Hence, there are $4n + 7$ steps for each outer loop. The outer **for** loop executes $n + 1$ times, so the total number of steps executed is $(n+1)(4n+7)+1+2(n+1)+(n+2) = 4n^2+11n+7+3n+5 = 4n^2 + 14n + 12$. Therefore, PRINTSQUARE runs in $O(n^2)$. □

Prints a triangle of '' of height* n+1.

PRINTTRIANGLE(n)

1 **for** $i \leftarrow 0$ **to** n $\quad\triangleright$ 1 initial assignment; $n+1$ increments; $n+2$ comparisons
2 $\quad$ **do for** $j \leftarrow 0$ **to** i $\quad\triangleright$ 1 initial assignment; $i+1$ increments; $i+2$ comparisons
3 $\quad\quad$ **do** PRINT $*$ $\quad\triangleright$ 1 operation
4 $\quad$ PRINT linebreak $\quad\triangleright$ 1 operation

For each inner **for** loop, line 3 executes 1 operation for $i + 1$ times. Hence, the number of steps for one inner **for** loop is $1 + 2(i + 1) + (i + 2) + (i + 1) = 4i + 6$.

When the outer **for** loop takes on value i, the inner **for** loop executes $4i + 6$ steps. Line 4 execute 1 operation. Hence, there are $4i + 7$ steps for each outer loop. The outer **for** loop executes $n + 1$ times, so the total number of steps executed is

$$\begin{aligned}
\left(\sum_{i=0}^{n} 4i + 7\right) + 1 + 2(n+1) + (n+2) &= \left(4\sum_{i=0}^{n} i\right) + \left(\sum_{i=0}^{n} 7\right) + 1 + 2(n+1) + (n+2) \\
&= 2n(n+1) + 7(n+1) + 3n + 5 \\
&= 2n^2 + 2n + 7n + 7 + 3n + 5 \\
&= 2n^2 + 12n + 12.
\end{aligned}$$

Therefore, PRINTTRIANGLE runs in $O(n^2)$. □

Counts the number of positive integers in array A with $n+1$ elements.

```
COUNTPOSITIVE(A, n)
1  count ← 0                      ▷ 1 assignment
2  for i ← 0 to n                 ▷ 1 initial assignment; n + 1 increments; n + 2 comparisons
3      do if A[i] > 0             ▷ 1 array indexing; 1 comparison
4            then count ← count + 1   ▷ 1 arithmetic operation; 1 assignment
5  PRINT count                    ▷ 1 operation
```

Since this code fragment contains an **if** statement, which might evaluate to true or false, we need to consider the worst case possible. In this case, the worst case occurs when all integers in the array A are positive.

Hence, for each **for** loop, line 3 performs 2 steps and, assuming that the **if** statement evaluates to TRUE, line 4 executes for 2 steps. Thus, the number of steps for one **for** loop is $2+2=4$. The **for** loop executes $n+1$ times, so the number of steps executed by the **for** loop is $4(n+1)+1+2(n+1)+(n+2)=7n+9$.

Finally, line 1 and 5 execute 2 steps. Therefore, the total number of steps executed is $7n+11$. That is, COUNTPOSITIVE runs in $O(n)$. □

Assume that the function MIN(a,b) *is O(1).*

```
WEIRD(A, n)
1  count ← 0                      ▷ 1 assignment
2  for i ← 0 to n                 ▷ 1 initial assignment; n + 1 increments; n + 2 comparisons
3      do if A[i] ≥ 0             ▷ 1 array indexing; 1 comparison
4            then count ← count + 1   ▷ 1 arithmetic operation; 1 assignment
5            else  for j ← 0 to MIN(A[i], n)
                   ▷ 1 initial assignment; 0 increments
                   ▷ 1 array indexing; 1 procedure call; 1 comparison
6                      do PRINT *         ▷ 1 operation
7                         j ← j + 1       ▷ 1 arithmetic operation; 1 assignment
8  PRINT count                            ▷ 1 operation
```

Again, we consider the worst case. In this problem it is unclear which case of the **if** statement gives the worst case, so we will consider both cases. In case that the **if** statement evaluates to TRUE, line 4 executes 2 steps. Otherwise, $A[i]$ must be negative, and the value of MIN$(A[i],n)$ is always $A[i]$. Hence, the **for** loop never executes, but the overhead of setting up the loop is 4 steps: the initial assignment, the array indexing, the MIN procedure call, and the comparison between j and the value returned from the MIN procedure. Thus, the **else** case executes for a more number of steps, so we will assume that in the worse case, the array A contains all negative integers. Line 3 performs 2 steps for each iteration. The number of steps in each outer **for** loop is $4+2=6$.

The outer **for** loop executes $n+1$ times, so the number of steps executed by the outer **for** loop is $6(n+1)+1+2(n+1)+(n+2)=9n+11$. Finally, line 1 and 8 execute 2 steps. Therefore, the total number of steps executed is $9n+13$. That is, WEIRD runs in $O(n)$. □

Assume that the function MOD(a, b) *is O(1). Note that* MOD*(a,b) returns the remainder of the division of a by b.*

WEIRD2(n)

```
1  i ← 0                         ▷ 1 assignment
2  while i < n                   ▷ n + 1 comparisons
3      do if MOD(i, 2) = 0       ▷ 1 procedure call; 1 comparison
4            then for j ← 0 to n ▷ 1 initial assignment; n + 1 increments; n + 2 comparisons
5                    do PRINT *  ▷ 1 operation
6         i ← i + 1              ▷ 1 arithmetic operation; 1 assignment
```

The **for** loop with 1 operation executes $n + 1$ times. The number of steps executed by each **for** loop is $(n + 1) + 1 + 2(n + 1) + (n + 2) = 4n + 6$. Now, the **for** loop is executed only if i is even. If n is even, the **if** statement evaluates to TRUE $n/2$ times. If n is odd, the **if** statement evaluates to TRUE $(n + 1)/2$ times. Therefore, the worst case occurs when n is odd. Thus, each **while** loop performs $\frac{n+1}{2}(4n + 6 + 2 + 2) = 2n^2 + 7n + 5$ steps. Hence, the number of steps executed by the **while** loop is $2n^2 + 7n + 5 + (n + 1) = 2n^2 + 8n + 6$. Finally, the total number of steps executed is $2n^2 + 8n + 6 + 1 = 2n^2 + 8n + 7$. That is, WEIRD2 runs in $O(n^2)$. □

WEIRD3(n)

```
1  i ← 1                         ▷ 1 assignment
2  while i < n                   ▷ n/2 + 1 comparisons
3      do if MOD(i, 2) = 0       ▷ 1 procedure call; 1 comparison
4            then for j ← 0 to n ▷ 1 initial assignment; n + 1 increments; n + 2 comparisons
5                    do PRINT *  ▷ 1 operation
6         i ← i + 2              ▷ 1 arithmetic operation; 1 assignment
```

This code fragment is quite similar to WEIRD2 except on line 1 and line 6, where i starts at 1 and gets incremented by 2 instead of 1. That is, i will always be odd, and so the **if** statement always evaluates to FALSE. Hence, in one iteration of **while** loop, there are 4 steps to be executed. The **while** loops executes $n/2$ times because i is incremented by 2. To be precise, if n is even, the loop executes $n/2$ times; if n is odd, the loop executes $(n - 1)/2$ times. We take the former case as the worst case. Therefore, the number of steps performed by the **while** loop is $2n + n/2 + 1 = 5n/2 + 1$. Finally, the total number of steps performed is $5n/2 + 1 + 1 = 5n/2 + 2$. That is, WEIRD3 runs in $O(n)$. □