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 $n+1$.

\begin{verbatim}
PRINT SQUARE(n)
  for i ← 0 to n    ▷ 1 initial assignment; $n + 1$ increments; $n + 2$ comparisons
    do for j ← 0 to n   ▷ 1 initial assignment; $n + 1$ increments; $n + 2$ comparisons
      do PRINT *      ▷ 1 operation
      PRINT linebreak ▷ 1 operation
  ▷ 2 initial assignment; $n + 1$ increments; $n + 2$ comparisons
\end{verbatim}

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 $n+1$.

\begin{verbatim}
PRINT TRIANGLE(n)
  for i ← 0 to n    ▷ 1 initial assignment; $n + 1$ increments; $n + 2$ comparisons
    do for j ← 0 to i   ▷ 1 initial assignment; $i + 1$ increments; $i + 2$ comparisons
      do PRINT *      ▷ 1 operation
      PRINT linebreak ▷ 1 operation
  ▷ 2 initial assignment; $n + 1$ increments; $n + 2$ comparisons
\end{verbatim}

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

\[
\left(\sum_{i=0}^{n} 4i + 7\right) + 1 + 2(n + 1) + (n + 2) = \left(4 \sum_{i=0}^{n} i\right) + \left(\sum_{i=0}^{n} 7\right) + 1 + 2(n + 1) + (n + 2)
\]

\[
= 2n(n + 1) + 7(n + 1) + 3n + 5
\]

\[
= 2n^2 + 2n + 7n + 7 + 3n + 5
\]

\[
= 2n^2 + 12n + 12.
\]

Therefore, PRINTTRIANGLE runs in $O(n^2)$.
Counts the number of positive integers in array $A$ with $n + 1$ elements.

\[
\text{COUNT_POSITIVE}(A, n)
\]

1. \(\text{count} \leftarrow 0\) \hspace{1cm} \triangleright 1 \text{ assignment}
2. \(\text{for } i \leftarrow 0 \text{ to } n\) \hspace{1cm} \triangleright 1 \text{ initial assignment; } n + 1 \text{ increments; } n + 2 \text{ comparisons}
3. \hspace{1cm} \text{do if } A[i] > 0 \hspace{1cm} \triangleright 1 \text{ array indexing; } 1 \text{ comparison}
4. \hspace{1cm} \text{then } \text{count} \leftarrow \text{count} + 1 \hspace{1cm} \triangleright 1 \text{ arithmetic operation; } 1 \text{ assignment}
5. \hspace{1cm} \text{print count} \hspace{1cm} \triangleright 1 \text{ 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 \text{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, \text{COUNT_POSITIVE} runs in $O(n)$.

Assume that the function \text{MIN}(a, b)$ is $O(1)$.

\[
\text{WEIRD}(A, n)
\]

1. \(\text{count} \leftarrow 0\) \hspace{1cm} \triangleright 1 \text{ assignment}
2. \(\text{for } i \leftarrow 0 \text{ to } n\) \hspace{1cm} \triangleright 1 \text{ initial assignment; } n + 1 \text{ increments; } n + 2 \text{ comparisons}
3. \hspace{1cm} \text{do if } A[i] \geq 0 \hspace{1cm} \triangleright 1 \text{ array indexing; } 1 \text{ comparison}
4. \hspace{1cm} \text{then } \text{count} \leftarrow \text{count} + 1 \hspace{1cm} \triangleright 1 \text{ arithmetic operation; } 1 \text{ assignment}
5. \hspace{1cm} \text{else for } j \leftarrow 0 \text{ to } \text{MIN}(A[i], n) \hspace{1cm} \triangleright 1 \text{ initial assignment; } 0 \text{ increments}
\hspace{1cm} \triangleright 1 \text{ array indexing; } 1 \text{ procedure call; } 1 \text{ comparison}
\hspace{1cm} \text{do \text{print} } \ast \hspace{1cm} \triangleright 1 \text{ operation}
6. \hspace{1cm} \text{end do}
7. \hspace{1cm} j \leftarrow j + 1 \hspace{1cm} \triangleright 1 \text{ arithmetic operation; } 1 \text{ assignment}
8. \hspace{1cm} \text{print count} \hspace{1cm} \triangleright 1 \text{ 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 \text{true}, line 4 executes 2 steps. Otherwise, $A[i]$ must be negative, and the value of \text{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 \text{MIN} procedure call, and the comparison between $j$ and the value returned from the \text{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, \text{WEIRD} runs in $O(n)$. □
Assume that the function $\text{MOD}(a, b)$ is $O(1)$. Note that $\text{MOD}(a, b)$ returns the remainder of the division of $a$ by $b$.

**WEIRD2($n$)**

1. $i \leftarrow 0$  \hspace{1cm} \triangleright 1 \text{ assignment}
2. while $i < n$  \hspace{1cm} \triangleright $n + 1 \text{ comparisons}$
3. \hspace{1cm} do if $\text{MOD}(i, 2) = 0$ \hspace{1cm} \triangleright 1 \text{ procedure call; 1 comparison}$
4. \hspace{2cm} then for $j \leftarrow 0 \text{ to } n$  \hspace{1cm} \triangleright 1 \text{ initial assignment; } n + 1 \text{ increments; } n + 2 \text{ comparisons}$
5. \hspace{3cm} do PRINT $\ast$  \hspace{1cm} \triangleright 1 \text{ operation}$
6. \hspace{1cm} i \leftarrow i + 1$  \hspace{1cm} \triangleright 1 \text{ 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. $i \leftarrow 1$  \hspace{1cm} \triangleright 1 \text{ assignment}
2. while $i < n$  \hspace{1cm} \triangleright $\frac{n}{2} + 1 \text{ comparisons}$
3. \hspace{1cm} do if $\text{MOD}(i, 2) = 0$ \hspace{1cm} \triangleright 1 \text{ procedure call; 1 comparison}$
4. \hspace{2cm} then for $j \leftarrow 0 \text{ to } n$  \hspace{1cm} \triangleright 1 \text{ initial assignment; } n + 1 \text{ increments; } n + 2 \text{ comparisons}$
5. \hspace{3cm} do PRINT $\ast$  \hspace{1cm} \triangleright 1 \text{ operation}$
6. \hspace{1cm} i \leftarrow i + 2$  \hspace{1cm} \triangleright 1 \text{ 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 $O(n)$.