3. Growth of functions and asymptotic notations

Computer Science Department,
College of Computer and Information Sciences,
King Saud University.

CSC 311: Design and Analysis of Algorithms\textsuperscript{1}
Dr. Waleed Alsalih

3.5 Time analysis of nonrecursive algorithms[Section 2.3]

How to analyze the time efficiency of an algorithm?
This can be done through two steps:

1. Assuming the worst case, count the number of major comparisons as a function $f$ of the input size.
2. Find the order of growth of $f$ using $O$ and/or $\Theta$.

Example:
Consider the problem of finding the largest value in a list.
Number of major comparisons$ = n - 1 \in \Theta(n)$.

**Algorithm 1**: An algorithm that finds the largest value in a list.

\begin{verbatim}
Algorithm MaxElement(A[0..n-1])
  maxval:=A[0];
  for i:=1..n-1 do
    if A[i] > maxval then
      maxval:=A[i];
  end
  return maxval;
\end{verbatim}

\textsuperscript{1}This is a summary of the material we cover from the textbook: *Introduction to the Design & Analysis of Algorithms*, A. Levitin, Second Edition, Pearson Addison-Wesley, 2006.
3. Growth of functions and asymptotic notations

Example:
Consider an algorithm that finds the product of two $n$-by-$n$ matrices $A$ and $B$.
Note that it suffices to focus on the most inner loops in counting major comparisons. Why?

Algorithm 2: An algorithm that finds the product of two $n$-by-$n$ matrices.

```plaintext
Algorithm MatrixMultiplication($A[0.. n-1, 0.. n-1], B[0.. n-1, 0.. n-1]$)
for $i:=0..n-1$ do
  for $j:=0..n-1$ do
    $C[i, j]:=0$;
    for $k:=0..n-1$ do
    end
  end
end
return $C$;
```

Number of major comparisons $=\sum_{i=0}^{n-1} \sum_{j=0}^{n-1} \sum_{k=0}^{n-1} 1 = n^3 \in \Theta(n)$.

Example:
Consider an algorithm that finds the number of binary digits in the binary representation of a positive decimal integer.

Number of major comparisons $=\lceil \log_2 n \rceil + 1 \in \log_2 n^1$.

Algorithm 3: Finding the number of binary digits in the binary representation of a positive decimal integer.

```plaintext
Algorithm Binary($n$)
count:=1;
while $n > 1$ do
  count:=count + 1;
  $n:=\lfloor n/2 \rfloor$;
end
return count;
```
3. Growth of functions and asymptotic notations

\footnote{To be precise, the input size here is the number of digits representing \( n \), and the algorithm is linear in the input size.}