



Figure 2.8 Comparison of growth rates of n and $\log n$.

EXAMPLE 2.3 Time complexity of power algorithms

The simple power algorithm's efficiency is given by equation (2.2):

$$\text{No. of multiplications} = n$$

The number of multiplications grows proportionately to n . We say that the simple power algorithm's time complexity is *of order n* , and write this as:

$$O(n)$$

(The analysis in this particular example was trivial, but if the number of multiplications had been $2n + 3$, the time complexity would still have been $O(n)$.)

The smart power algorithm's efficiency is given by equation (2.3):

$$\text{Maximum no. of multiplications} = 2 \text{ floor}(\log_2 n) + 2$$

Discarding the slower-growing term ($+ 2$) and the constant factor (2), we get $\text{floor}(\log_2 n)$. We can approximate this as $\log_2 n$. Thus the number of multiplications grows proportionately to $\log_2 n$. We say that the smart power algorithm's time complexity is *of order $\log n$* , and write this as:

$$O(\log n)$$