Algorithm recursiveMax(A, n):

Input: An array A storing $n \geq 1$ integers.

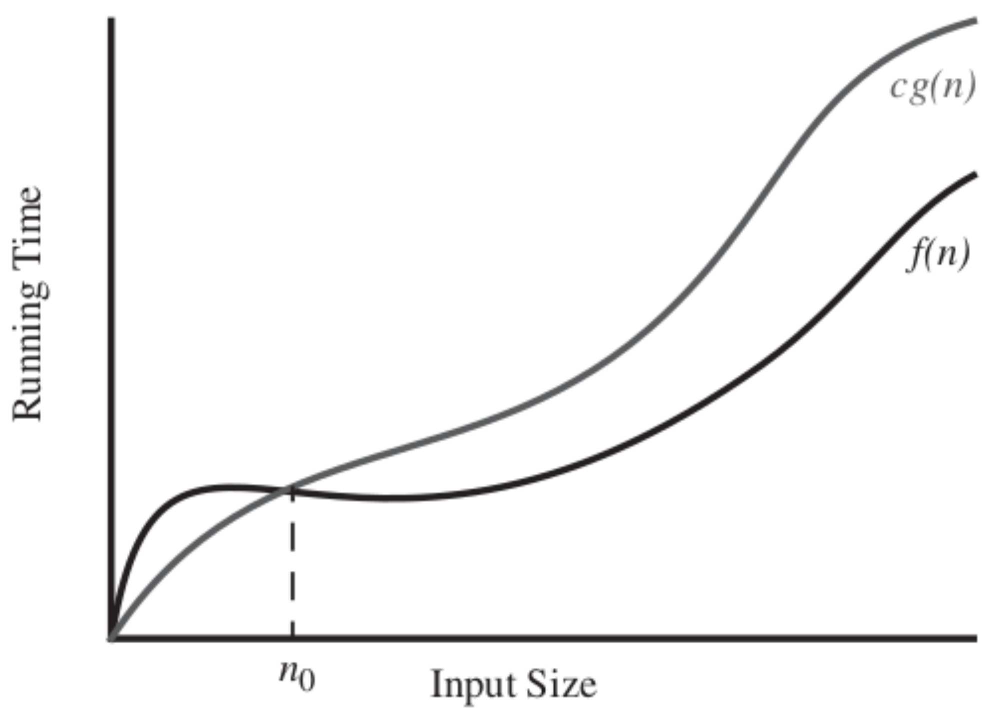
Output: The maximum element in A.

```
if n = 1 then
    return A[0]
return max{recursiveMax(A, n − 1), A[n − 1]}
```

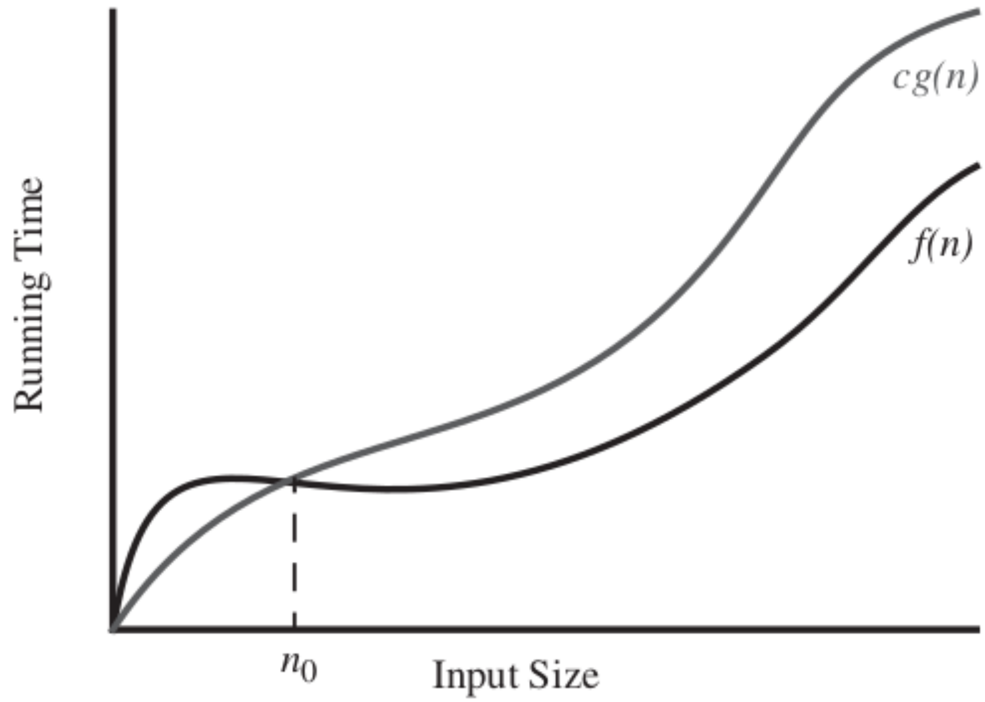
Algorithm 1.4: Algorithm recursiveMax.

$$T(n) = \begin{cases} 3 & \text{if } n = 1 \\ T(n-1) + 7 & \text{otherwise} \end{cases}$$

$$T(n) = 7(n-1) + 3 = 7n - 4$$



The function $f(n)$ is $O(g(n))$, for $f(n) \leq c \cdot g(n)$ when $n \geq n_0$.



The function $f(n)$ is $O(g(n))$, for $f(n) \leq c \cdot g(n)$ when $n \geq n_0$.

Let f(n) and g(n) be functions mapping non-negative integers to real numbers.

We say "f(n) is O(g(n))", or "f(n) is order of g(n)", if there exists a real constant c > 0 and an integer constant n0 >= 1 such that f(n) <= c.g(n) for every integer n >= n0.

Example 1.1: f(n) = 7n - 2 is O(n).

Proof: We need a real constant c > 0 and an integer constant n0 >= 1 such that (7n-2) <= c.n for every integer n >= n0. One possible choice is c = 7, and n0 = 1.

Corollary: The running time of arrayMax is O(n).

Example 1.3: $20n^3 + 10n \log n + 5$ *is* $O(n^3)$.

Proof: $20n^3 + 10n \log n + 5 \leq 35n^3$, *for* $n \geq 1$.

Note - If f(n) is a polynomial function of degree k, then f(n) will always be O(n^k).

Example 1.4: f(n) = 3.log(n) + log(log(n)) is O(log n).
Proof: We can choose c=4 and n0 = 2.

Example 1.5: 2^{100} *is* $O(1)$.

Proof: $2^{100} \leq 2^{100} \cdot 1$, *for* $n \geq 1$. *Note that variable* n *does not appear in the inequality, since we are dealing with constant-valued functions.* ■

We say "f(n) is Ω(g(n))", or "f(n) is big-Omega of g(n)", if there exists a real constant c > 0 and an integer constant n0 >= 1 such that f(n) >= c.g(n) for every integer n >= n0.

Example 1.9: $3\log n + \log\log n$ *is* $\Omega(\log n)$.

Proof: $3\log n + \log\log n \geq 3\log n$, *for* $n \geq 2$.

We say "f(n) is Θ(g(n))", or "f(n) is big-Theta of g(n)",
if there exists real constants c1, c2 > 0 and an integer constant n0 >= 1
such that f(n) <= c1.g(n) and f(n) >= c2.g(n) for every integer n >= n0.

Example 1.10: $3\log n + \log\log n$ *is* $\Theta(\log n)$.