function insertion-sort(A, n)
Input: Array A of length at least n.
Output: $A[0]..A[n-1]$ are permuted into sorted order.
if $n < 2$, return.
insertion-sort$(A, n-1)$.
insert(A, n).
return.

function insert(A, n)
Input: Array A of length at least n such that $A[0..A[n-2]$ are sorted.
Output: $A[0]..A[n-1]$ are permuted into sorted order.
if $n < 2$, return.
if $A[n-1] \geq A[n-2]$, return.
swap$(A[n-1], A[n-2])$.
insert$(A, n-1)$.
return.

Let $T_{in}(n)$ be the cost of `insert(A, n)`. Then

$$T_{in}(n) \leq T_{in}(n-1) + c, \text{ for } n > 1.$$

Thus by the muster theorem, $T_{in}(n)$ is in O(n).

Let $T_{is}(n)$ be the cost of `insertion-sort(A, n)`. Then

$$T_{is}(n) \leq T_{is}(n-1) + O(n), \text{ for } n > 1.$$

In other words,

$$T_{is}(n) \leq T_{is}(n-1) + c * n, \text{ for } n > 1 \text{ and for some constant } c.$$

Thus by the muster theorem, $T_{is}(n)$ is in $O(n^2)$.

Let n be given and let $T_m(n)$ be the number of multiplications used in `modexp` when the exponent e has n bits. This T_m satisifes

$$T_m(n) \leq T_m(n-1) + 2.$$

Thus by the muster theorem $T_m(n)$ is in $O(n)$.

Let $T(n)$ be the runtime cost of `modexp(a, e, N)` on n-bit inputs. If we use classical multiplication, each multiplication costs O(n^2) so $T(n)$ is in $O(n^3)$. If we use karatsuba multiplication (the divide and conquer approach of chapter 2.1), $T(n)$ is in $O(n^{2.59})$.