A = {1, 3, 2, 5}

recursiveMax(A, 4)
= max(recursiveMax(A, 3), A[3])
= max(max(recursiveMax(A, 2), A[2]), A[3])
= max(max(max(recursiveMax(A, 1), A[1]), A[2]), A[3])
= max(max(max(A[0], A[1]), A[2]), A[3])
= max(max(max(1, 3), 2), 5)
= max(max(3, 2), 5)
= max(3, 5)
= 5

```
max(a, b)

if a > b
   return a
else
   return b
```

```
Algorithm recursiveMax(A, n):
    Input: An array A storing n ≥ 1 integers.
    Output: The maximum element in A.
  if n = 1 then                                  1
      return A[0]                                2
  return max{recursiveMax(A, n − 1), A[n − 1]}   T(n-1) + 6
```

Algorithm 1.4: Algorithm recursiveMax.

return: 1, max: 2, operation (n-1): 2, recursive call: T(n-1), indexing A[n-1]: 1

T(n) = no. of primitive operations required to compute recursiveMax(A, n)

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

$$\begin{aligned} T(n) &= T(n-1) + 7 \\ &= T(n-2) + 7 + 7 \\ &= \dots \\ &= T(1) + 7(n-1) \\ &= 3 + 7(n-1) \\ &= 7n - 4 \end{aligned}$$