algorithm insertion\_sort(A: array of Comparable)

for i from 1 to length(A) - 1 do

j := i

while j > 0 and A[j-1] > A[j] do

swap(A[j], A[j-1])

j := j - 1

end while

end for

end algorithm  
  
  
algorithm selection\_sort(A: array of Comparable)

for i from 0 to length(A) - 2 do

min\_index := i

for j from i+1 to length(A) - 1 do

if A[j] < A[min\_index] then

min\_index := j

end if

end for

swap(A[i], A[min\_index])

end for

end algorithm  
  
  
algorithm parallel\_merge\_sort(A: array of Comparable)

if length(A) <= 1 then

return A

end if

// Divide the array into two sub-arrays of approximately equal size.

mid := length(A) / 2

left := A[0..mid-1]

right := A[mid..length(A)-1]

// Sort the two sub-arrays in parallel.

spawn sort(left)

spawn sort(right)

// Wait for both sorts to finish.

sync

// Merge the two sub-arrays into a single sorted array.

result := []

i := 0

j := 0

while i < length(left) and j < length(right) do

if left[i] <= right[j] then

append result, left[i]

i := i + 1

else

append result, right[j]

j := j + 1

end if

end while

// Append any remaining elements in left or right to the result array.

while i < length(left) do

append result, left[i]

i := i + 1

end while

while j < length(right) do

append result, right[j]

j := j + 1

end while

return result

end algorithm

algorithm sort(A: array of Comparable)

if length(A) <= 1 then

return A

end if

// Divide the array into two sub-arrays of approximately equal size.

mid := length(A) / 2

left := sort(A[0..mid-1])

right := sort(A[mid..length(A)-1])

// Merge the two sub-arrays into a single sorted array.

result := []

i := 0

j := 0

while i < length(left) and j < length(right) do

if left[i] <= right[j] then

append result, left[i]

i := i + 1

else

append result, right[j]

j := j + 1

end if

end while

// Append any remaining elements in left or right to the result array.

while i < length(left) do

append result, left[i]

i := i + 1

end while

while j < length(right) do

append result, right[j]

j := j + 1

end while

return result

end algorithm