**Insertion sort**

Insertion sort is a pretty fast algorithm compared with other elementary sorting algorithms, which sorts a set of n elements in O(n2) worst and average case time. An iterative algorithm INSERTIONSORT-ITERATIVE is given in Figure 3. The algorithm has n−1

iterations. In iteration I ∈[1,n−1) , the array element A[i] will be inserted in a sorted position in the range A[0..i−1]. After each iteration i, the subarray A[0..i] will be sorted. The runtime complexity is data sensitive.

A recursive D&C insertion sort algorithm INSERTIONSORT is shown in Figure [3](javascript:;). The initial invocation to the algorithm is InsertionSort(A[0..n−1])InsertionSort(n[0..n−1])⁠. The recursive structure of the algorithm is different from that of bubble and selection sorts. The InsertionSortInsertionSort function calls itself twice to sort the left and right halves separately and simultaneously. Then it invokes the MergeMerge function to merge the elements from the two halves using the logic of the iterative insertion sort. After the merge, the entire array would be sorted.

The merge function MergeMerge calls itself a total of three times: the first two calls in parallel and then a third serial call. The first call Merge(A[ℓℓ..ℓm],A[rℓ..rm])Merge(n[ℓℓ..ℓn],n[nℓ..nn]) brings the smallest elements to A[ℓℓ..ℓm]n[ℓℓ..ℓn] in sorted order. The second call Merge(A[ℓm+1..ℓh],A[rm+1..rh])Merge(n[ℓn+1..ℓℎ],n[nn+1..nℎ]) brings the largest elements to A[rm+1..rh]n[nn+1..nℎ] in sorted order. The third call Merge(A[ℓm+1..ℓh],A[rℓ..rm])Merge(n[ℓn+1..ℓℎ],n[nℓ..nn]) brings the remaining elements to A[ℓm+1..rm]n[ℓn+1..nn] in the sorted order.

The base case kernel of InsertionSortInsertionSort function is equivalent to InsertionSort-IterativeInsertionSort-Iterative⁠. The base case kernel of the MergeMerge function merges two sorted subarrays to a sorted array. In iteration kn⁠, the knth element of the right subarray gets merged with its previous elements in the right subarray and with the elements of the left subarray. After rh−rℓ+1nℎ−nℓ+1 iterations, the elements in the two subarrays would be merged into a sorted array—the left subarray will be sorted, the right subarray will be sorted and the last element of the left subarray will be less than or equal to the first element of the right subarray.