AlgoStat 2

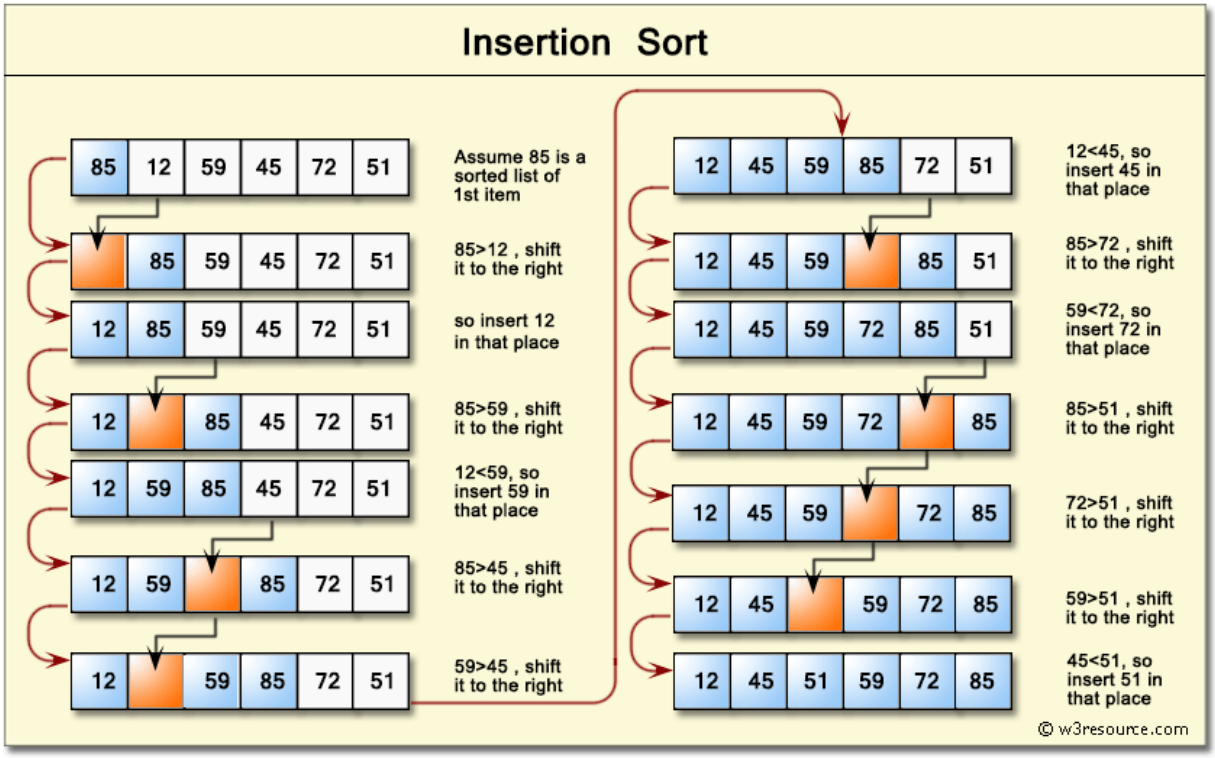
Sorting algorithms' time and cost comparison

Types of sorts

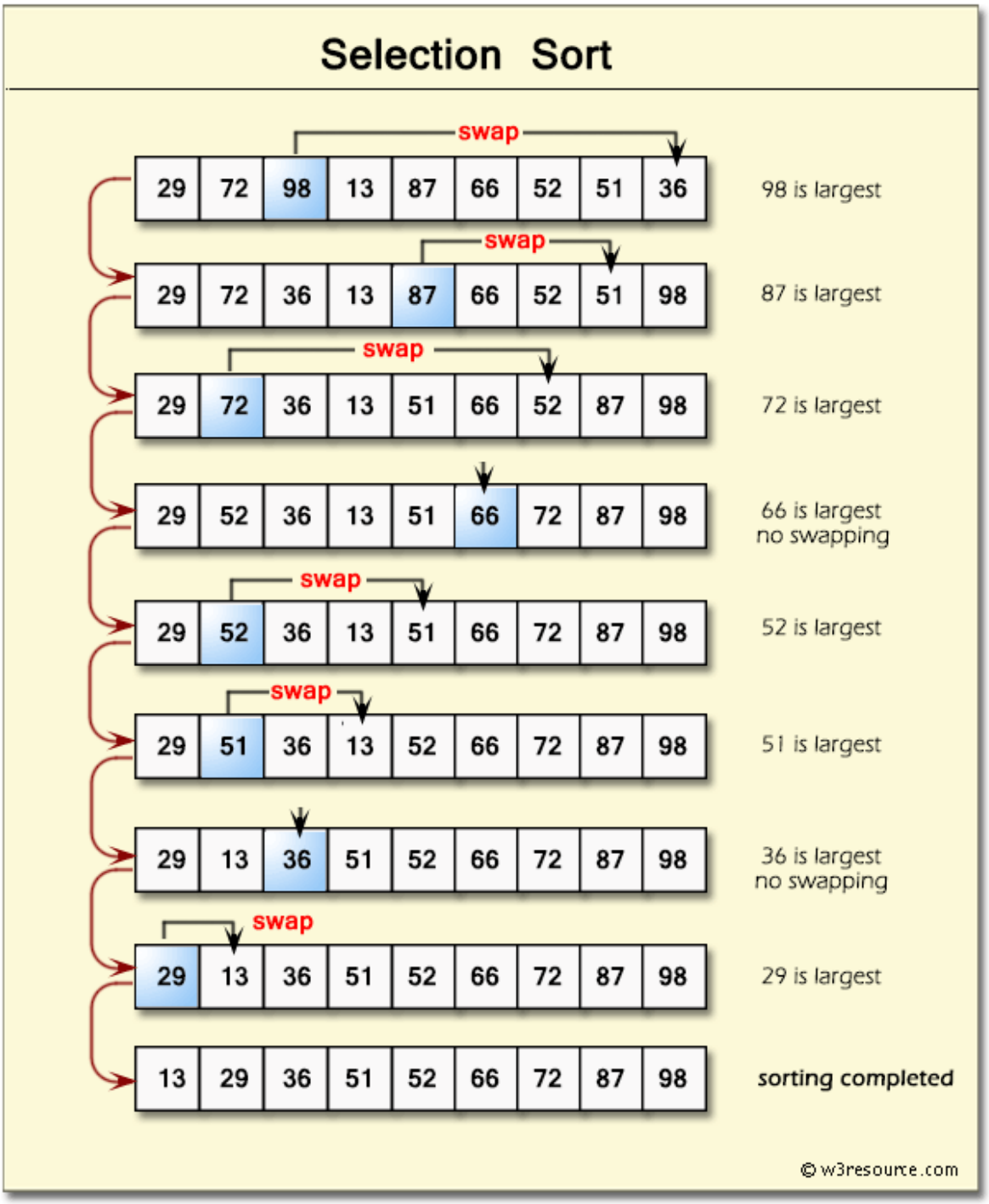
1. Insertion
2. Selection
3. Bubble
4. Shell
5. Quick
6. Comb
7. Merge

Explanation of each type of sort

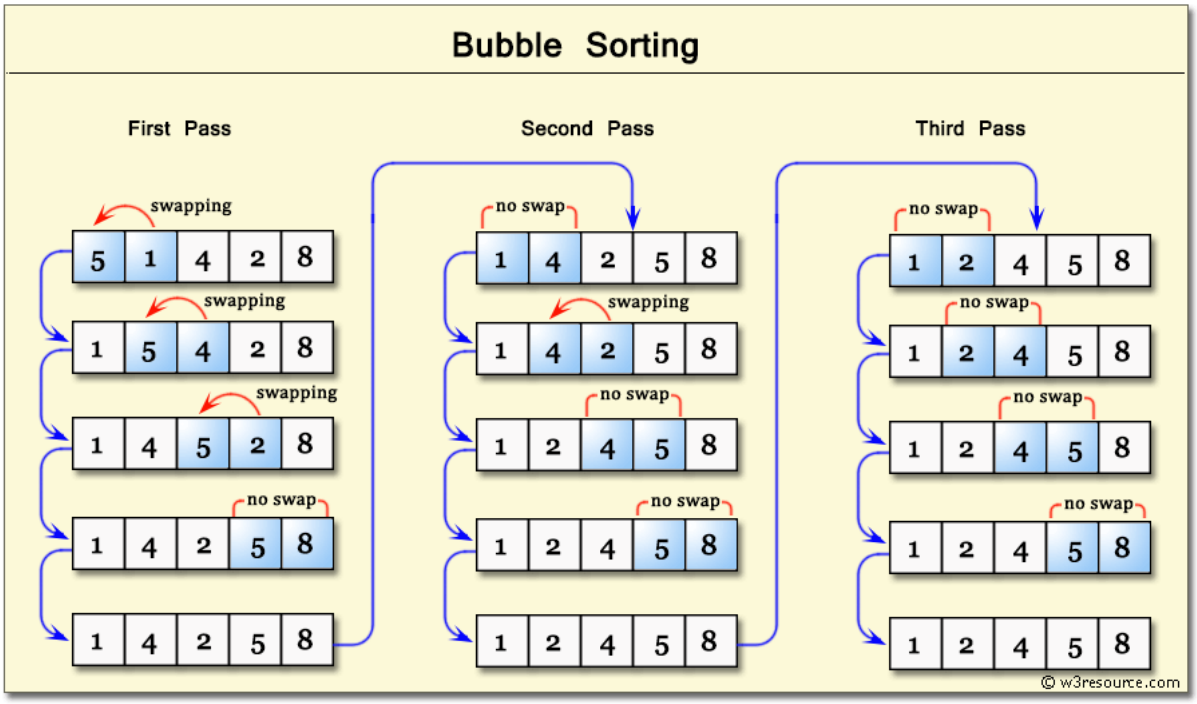
Insertion sort : Insertion sort is a simple sorting algorithm that is relatively efficient for small lists and mostly sorted lists, and is often used as part of more sophisticated algorithms. It works by taking elements from the list one by one and inserting them in their correct position into a new sorted list. In arrays, the new list and the remaining elements can share the array's space, but insertion is expensive, requiring shifting all following elements over by one.



Selection sort : Selection sort is an in-place comparison sort. It has O(n2) complexity, making it inefficient on large lists, and generally performs worse than the similar insertion sort. Selection sort is noted for its simplicity, and also has performance advantages over more complicated algorithms in certain situations.
The algorithm finds the minimum value, swaps it with the value in the first position, and repeats these steps for the remainder of the list. It does no more than n swaps, and thus is useful where swapping is very expensive.



Bubble sort : Sometimes referred to as sinking sort, is a simple sorting algorithm that repeatedly steps through the list to be sorted, compares each pair of adjacent items and swaps them if they are in the wrong order. The pass through the list is repeated until no swaps are needed, which indicates that the list is sorted. The algorithm, which is a comparison sort, is named for the way smaller elements "bubble" to the top of the list. Although the algorithm is simple, it is too slow and impractical for most problems even when compared to insertion sort. It can be practical if the input is usually in sort order but may occasionally have some out-of-order elements nearly in position.

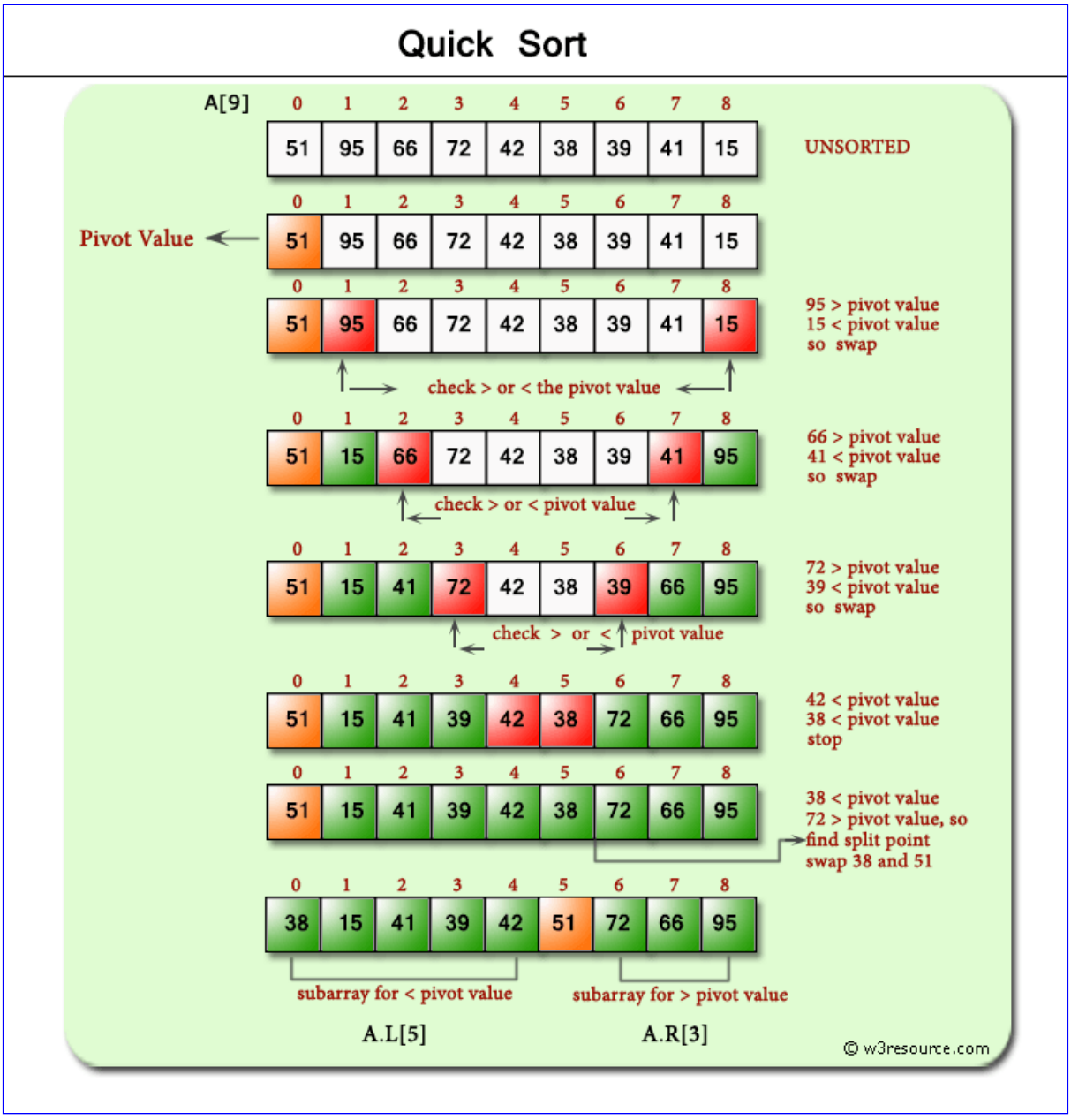


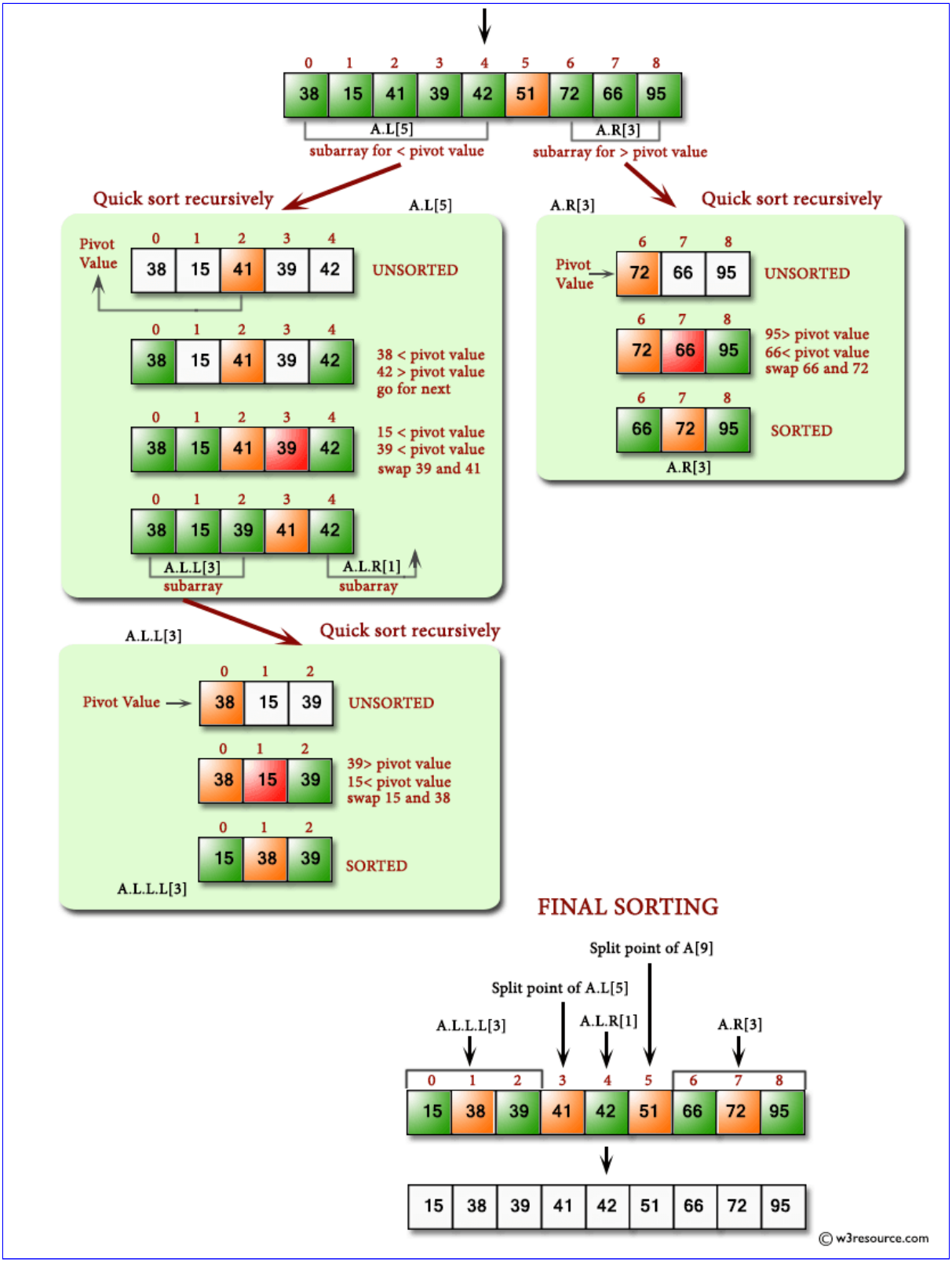
Shell sort : Shell sort or Shell's method, is an in-place comparison sort. It can be seen as either a generalization of sorting by exchange (bubble sort) or sorting by insertion (insertion sort). The method starts by sorting pairs of elements far apart from each other, then progressively reducing the gap between elements to be compared. Starting with far apart elements can move some out-of-place elements into position faster than a simple nearest neighbor exchange.

	a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8	a_9	a_{10}	a_{11}	a_{12}
input data:	62	83	18	53	07	17	95	86	47	69	25	28
after 5-sorting:	17	28	18	47	07	25	83	86	53	69	62	95
after 3-sorting:	17	07	18	47	28	25	69	62	53	83	86	95
after 1-sorting:	07	17	18	25	28	47	53	62	69	83	86	95

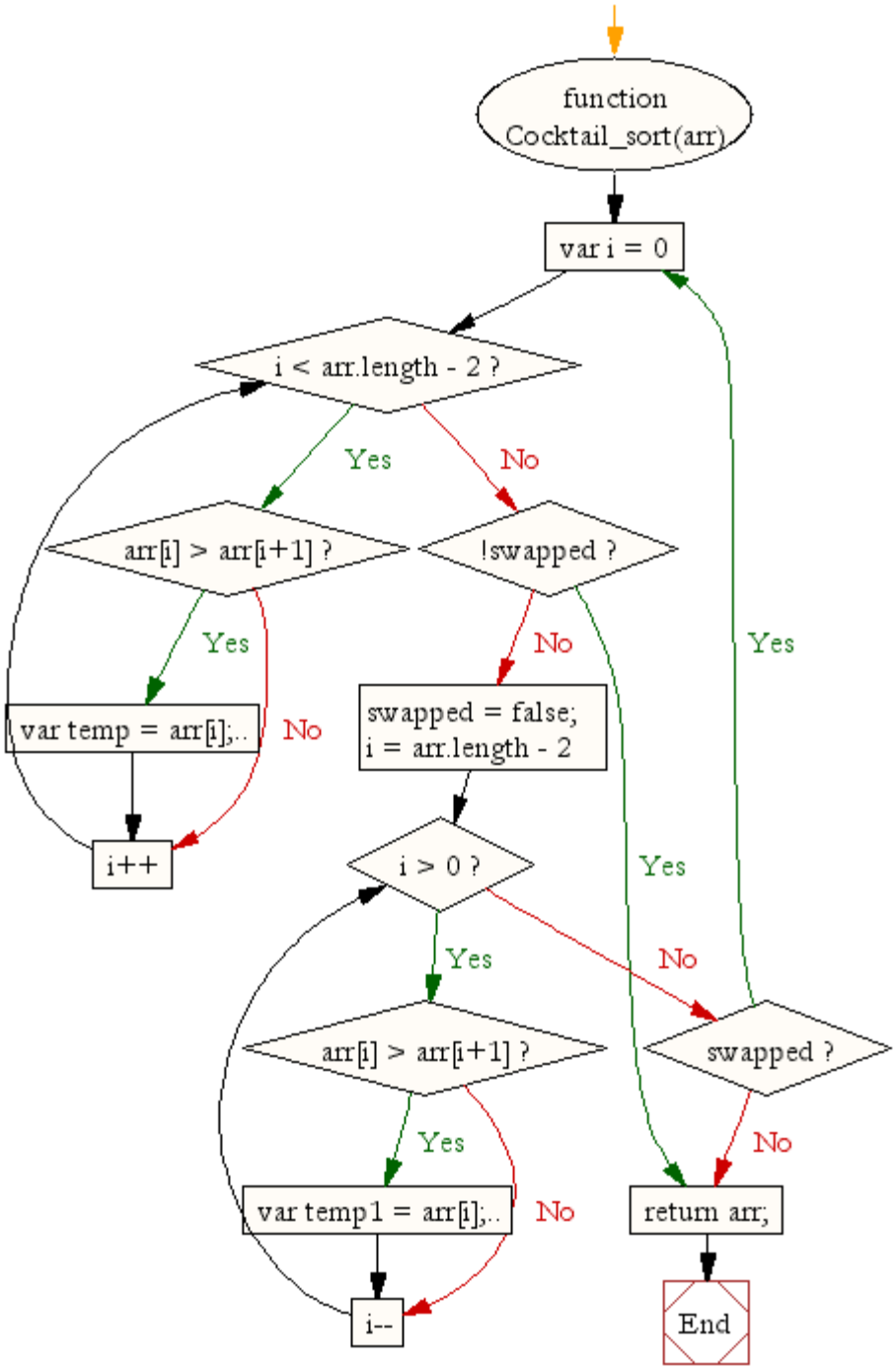
Quicksort : Quicksort is a comparison sort, meaning that it can sort items of any type for which a "less-than" relation (formally, a total order) is defined. In efficient implementations it is not a stable sort, meaning that the relative order of equal sort items is not preserved. Quicksort can operate in-place on an array, requiring small additional amounts of memory to perform the sorting. It is very similar to selection sort, except that it does not always choose worst-case partition.

Mathematical analysis of quicksort shows that, on average, the algorithm takes O(n log n) comparisons to sort n items. In the worst case, it makes O(n2) comparisons, though this behavior is rare.

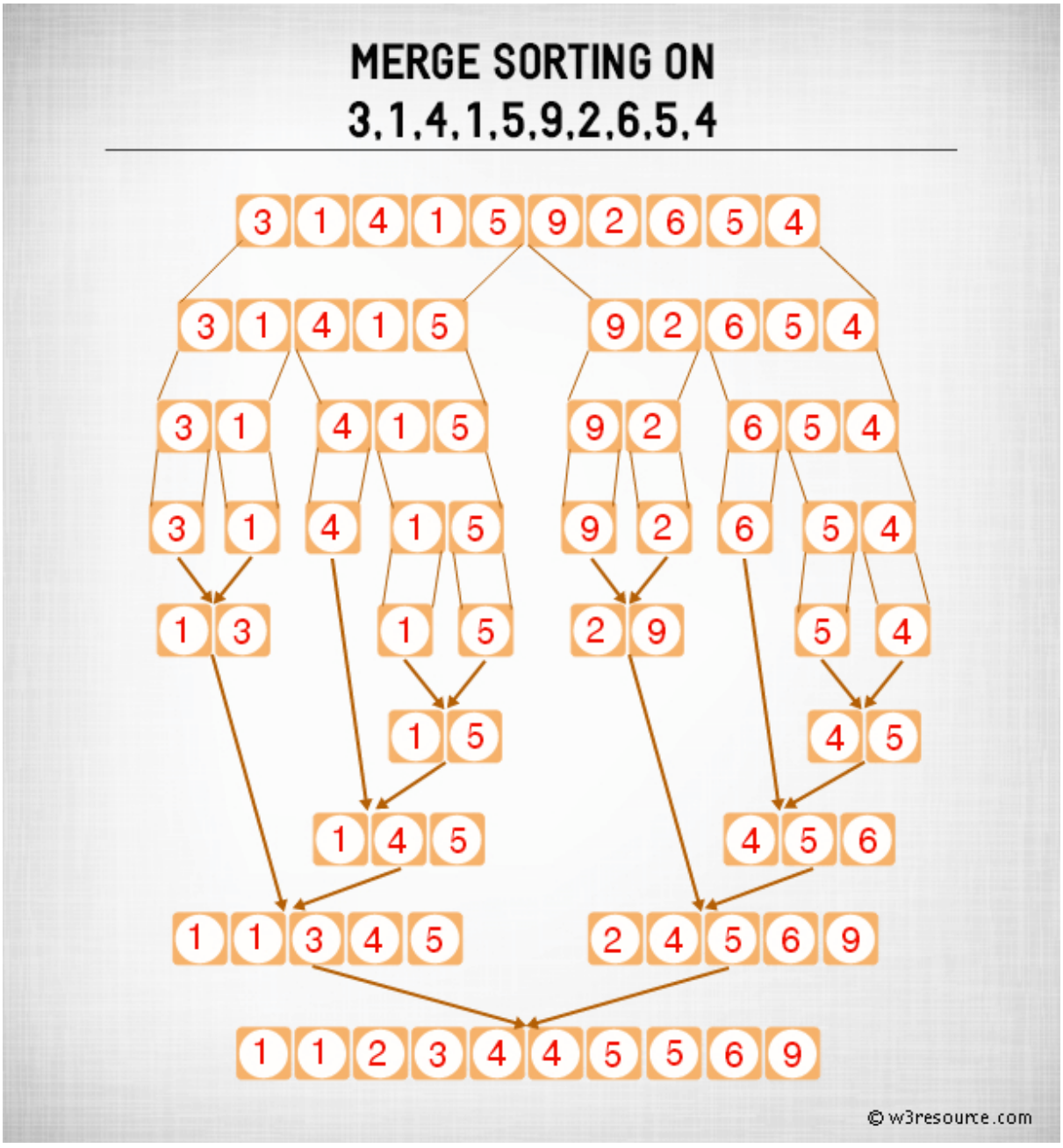



0 1 2 3 4 5 6 7 8
38 15 41 39 42 51 72 66 95
A.L[5]
subarray for < pivot value
A.R[3]
subarray for > pivot value
Quick sort recursively
A.L[5]
Pivot Value
UNSORTED
38 < pivot value
42 > pivot value
go for next
15 < pivot value
39 < pivot value
swap 39 and 41
A.L.L[3]
subarray
A.L.R[1]
subarray
A.L.L[3]
Quick sort recursively
Pivot Value
UNSORTED
39> pivot value
15< pivot value
swap 15 and 38
SORTED
A.L.L.L[3]
A.R[3]
Quick sort recursively
Pivot Value
UNSORTED
95> pivot value
66< pivot value
swap 66 and 72
SORTED
A.R[3]
FINAL SORTING
Split point of A[9]
Split point of A.L[5]
A.L.L.L[3]
A.L.R[1]
A.R[3]
15 38 39 41 42 51 66 72 95
15 38 39 41 42 51 66 72 95
© w3resource.com

Comb sort : The Comb sort is a variant of the Bubble Sort. Like the Shell sort, the Comb Sort increases the gap used in comparisons and exchanges. Some implementations use the insertion sort once the gap is less than a certain amount. The basic idea is to eliminate turtles, or small values near the end of the list, since in a bubble sort these slow the sorting down tremendously. Rabbits, large values around the beginning of the list do not pose a problem in bubble sort. In bubble sort, when any two elements are compared, they always have a gap of 1. The basic idea of comb sort is that the gap can be much more than 1.



Merge sort : Merge sort (also commonly spelled mergesort) is an O (n log n) comparison-based sorting algorithm. Most implementations produce a stable sort, which means that the implementation preserves the input order of equal elements in the sorted output.



Sources

www.w3resource.com

Wikipedia