Which of the following sorting algorithms in its typical implementation gives best performance when applied on an array which is sorted or almost sorted (maximum 1 or two elements are misplaced).

Quick Sort

Insertion Sort

Merge Sort

Selection Sort

Consider a situation where swap operation is very costly. Which of the following sorting algorithms should be preferred so that the number of swap operations are minimized in general?

Quick Sort

Insertion Sort

Merge Sort

Selection Sort

Suppose we are sorting an array of eight integers using quicksort, and we have just finished the first partitioning with the array looking like this:

2 5 1 7 9 12 11 10

Which statement is correct?

The pivot could be either the 7 or the 9.

The pivot could be the 7, but it is not the 9

The pivot is not the 7, but it could be the 9

Neither the 7 nor the 9 is the pivot.

In a modified merge sort, the input array is splitted at a position one-third of the length(N) of the array. Which of the following is the tightest upper bound on time complexity of this modified Merge Sort.

N(logN base 3)

N(logN base 2/3)

N(logN base 1/3)

N(logN base 3/2)

A list of n string, each of length n, is sorted into lexicographic order using the merge-sort algorithm. The worst case running time of this computation is

$O\ (n \log n)$

$O\ (n^2 \log n)$

$O\ (n^2 + \log n)$

$O\ (n^2)$

Let P be a QuickSort Program to sort numbers in ascending order using the first element as pivot. Let t1 and t2 be the number of comparisons made by P for the inputs {1, 2, 3, 4, 5} and {4, 1, 5, 3, 2} respectively. Which one of the following holds?

t1 = 5

t1 < t2

t1 > t2

t1 = t2

What is the best sorting algorithm to use for the elements in array are more than 1 million in general?

Merge sort.

Quick sort.

Insertion sort.