

1 Identifying Sorts

Below you will find intermediate steps in performing various sorting algorithms on the same input list. The steps do not necessarily represent consecutive steps in the algorithm (that is, many steps are missing), but they are in the correct sequence. For each of them, select the algorithm it illustrates from among the following choices: insertion sort, selection sort, mergesort, quicksort (first element of sequence as pivot), and heapsort. When we split an odd length array in half in mergesort, assume the larger half is on the right.

Input list: 1429, 3291, 7683, 1337, 192, 594, 4242, 9001, 4392, 129, 1000

- (a) 1429, 3291, 7683, 192, 1337, 594, 4242, 9001, 4392, 129, 1000
1429, 3291, 192, 1337, 7683, 594, 4242, 9001, 129, 1000, 4392
192, 1337, 1429, 3291, 7683, 129, 594, 1000, 4242, 4392, 9001
- (b) 1337, 192, 594, 129, 1000, 1429, 3291, 7683, 4242, 9001, 4392
192, 594, 129, 1000, 1337, 1429, 3291, 7683, 4242, 9001, 4392
129, 192, 594, 1000, 1337, 1429, 3291, 4242, 4392, 7683, 9001
- (c) 1337, 1429, 3291, 7683, 192, 594, 4242, 9001, 4392, 129, 1000
192, 1337, 1429, 3291, 7683, 594, 4242, 9001, 4392, 129, 1000
192, 594, 1337, 1429, 3291, 7683, 4242, 9001, 4392, 129, 1000
- (d) 1429, 3291, 7683, 9001, 1000, 594, 4242, 1337, 4392, 129, 192
7683, 4392, 4242, 3291, 1000, 594, 192, 1337, 1429, 129, 9001
129, 4392, 4242, 3291, 1000, 594, 192, 1337, 1429, 7683, 9001

In all these cases, the final step of the algorithm will be this:

129, 192, 594, 1000, 1337, 1429, 3291, 4242, 4392, 7683, 9001

- (f) You will be given an answer bank, each item of which may be used multiple times. You may not need to use every answer, and each statement may have more than one answer.
- A. QuickSort (in-place using Hoare partitioning and choose the leftmost item as the pivot)
 - B. MergeSort
 - C. Selection Sort
 - D. Insertion Sort
 - E. HeapSort
 - N. (None of the above)

List all letters that apply. List them in alphabetical order, or if the answer is none of them, use N indicating none of the above. All answers refer to the entire sorting process, not a single step of the sorting process. For each of the problems below, assume that N indicates the number of elements being sorted.

----- Bounded by $\Omega(N \log N)$ lower bound.

----- Has a worst case runtime that is asymptotically better than Quicksort's worstcase runtime.

----- In the worst case, performs $\Theta(N)$ pairwise swaps of elements.

----- Never compares the same two elements twice.

----- Runs in best case $\Theta(\log N)$ time for certain inputs

3 Bears and Beds

The hot new Cal startup AirBearsnBeds has hired you to create an algorithm to help them place their customers in the best possible homes to improve their experience. They are currently in their alpha stage so their only customers (for now) are bears. Now, a little known fact about bears is that they are very, very picky about their bed sizes: they do not like their beds too big or too little - they like them just right. Bears are also sensitive creatures who don't like being compared to other bears, but they are perfectly fine with trying out beds.

The Problem:

Given a list of Bears with unique but unknown sizes and a list of Beds with corresponding but also unknown sizes (not necessarily in the same order), return a list of Bears and a list of Beds such that that the i th Bear in your returned list of Bears is the same size as the i th Bed in your returned list of Beds. Bears can only be compared to Beds and we can get feedback on if the Bed is too large, too small, or just right. In addition, Beds can only be compared to Bears and we can get feedback if the Bear is too large for it, too small for it, or just right for it.

The Constraints:

Your algorithm should run in $O(N \log N)$ time on average. It may be helpful to figure out the naive $O(N^2)$ solution first and then work from there.