Sorted Runtimes

We want to sort an array of N **unique** numbers in ascending order. Determine the best case and worst case runtimes of the following sorts:

(a) Once the runs in merge sort are of $size <= N/100$, we perform insertion sort on them.

Best Case: $\Theta($), Worst Case: $\Theta($)

(b) We can only swap adjacent elements in selection sort.

Best Case: $\Theta($), Worst Case: $\Theta($)

(c) We use a linear time median finding algorithm to select the pivot in quicksort.

Best Case: $\Theta($), Worst Case: $\Theta($)

(d) We implement heapsort with a min-heap instead of a max-heap. You may modify heapsort but must maintain constant space complexity.

Best Case: $\Theta($), Worst Case: $\Theta($)

(e) We run an optimal sorting algorithm of our choosing knowing:

- There are at most N inversions

  Best Case: $\Theta($), Worst Case: $\Theta($)

- There is exactly 1 inversion

  Best Case: $\Theta($), Worst Case: $\Theta($)

- There are exactly $(N^2 - N)/2$ inversions

  Best Case: $\Theta($), Worst Case: $\Theta($)