Comments on the timing results

When `bubbleSort` and `improvedBubbleSort` were run on almost unsorted arrays, the running time grew at more or less the same rate, though `improvedBubbleSort` was a little slower due to the overhead of checking whether a swap was performed in a phase. Note that `improvedBubbleSort` does not save at all in the number of phases it needs to run. The two plot corresponding to running the functions on almost unsorted arrays are shown in the Figure 1. It is clear that the rate of growth of the functions is faster than linear and may be qudratic.

When `bubbleSort` and `improvedBubbleSort` were run on almost sorted arrays, we get plots that are similar to those above. These are shown in Figure 2. In these plots the running times of the two functions are very close together, sometimes `bubbleSort` is better and sometimes `improvedBubbleSort` better. This may surprising initially, but on examining `improvedBubbleSort` carefully, we see that even if the input array was obtained by taking a sequence that is completely in order and then swapping the first and the last elements, `improvedBubbleSort` still requires $n-1$ phases. In other words, if we start with $n-1, 1, 2, 3, \ldots, n-2, 0$ and run `improvedBubbleSort`, 0 gets to its correct position in the very first phase. However, in each phase, $n-1$ moves just one step to the right, towards its correct position.

A different kind of small perturbation does reveal situations in which `improvedBubbleSort` is much faster than `bubbleSort`. Let us start with a sorted sequence and make a bunch of swaps but ensure that all the swapped elements are in the first 10 slots of the array. Thus the first 10 elements are disorderly, but the remaining $n-10$ elements in the array are all in their correct positions. For this version of an almost sorted array, `improvedBubbleSort` does outperform `bubbleSort` dramatically. In general, `improvedBubbleSort` performs very well compared to `bubbleSort` on arrays in which each element is not too far away from its correct position. See Figure 3, for plots obtained by running the sorting algorithms on almost sorted arrays obtained using this different kind of perturbation. `improvedBubbleSort` is so fast that its plot is along the x-axis is not too clearly visible.

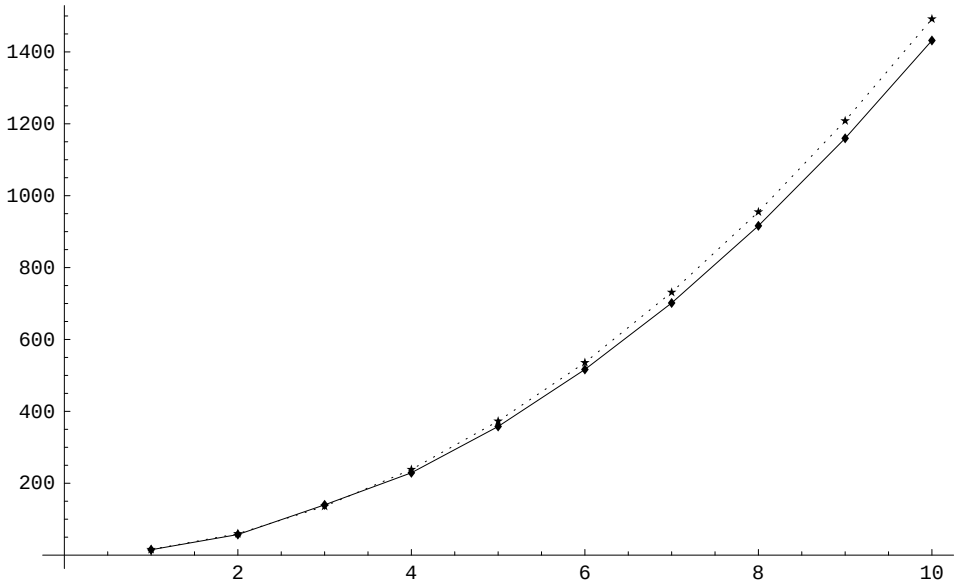

Figure 1: Shows the running times of `bubbleSort` and `improvedBubbleSort`, when these were run on almost unsorted arrays.

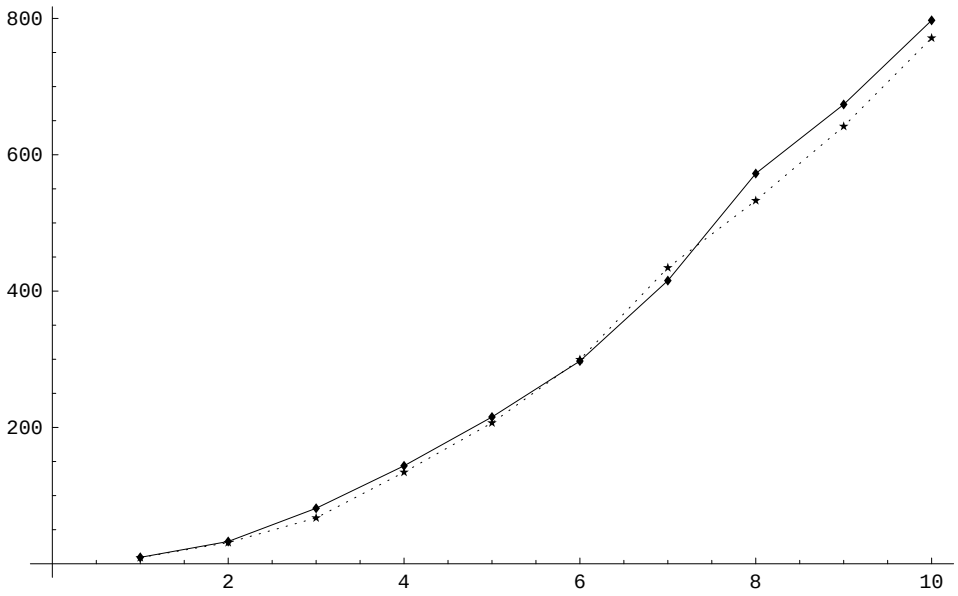

Figure 2: Shows the running times of `bubbleSort` and `improvedBubbleSort`, when these were run on almost sorted arrays.

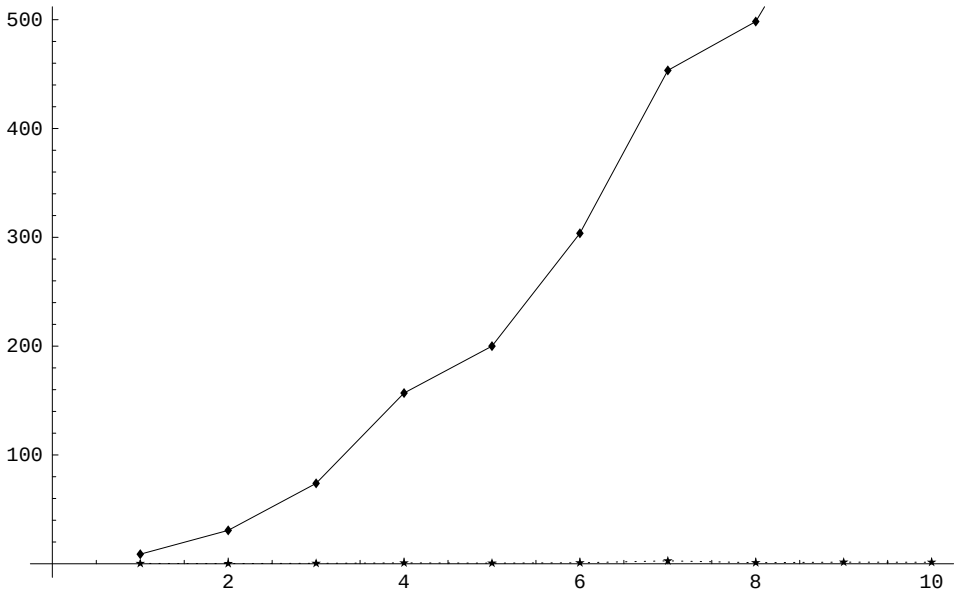

Figure 3: Shows the running times of `bubbleSort` and `improvedBubbleSort`, when these were run on almost sorted arrays, in which no element is too far away from its correct place.