Corey Kipp (Student ID: 57723335)

kippc@uci.edu

CS 161

Homework 3

R-8.5

Consider again the modification of the deterministic version of the quick-sort algorithm so that, instead of selecting the last element in an n-element sequence as the pivot, we choose the element at index $\lfloor n/2 \rfloor$. Describe the kind of sequence that would cause this version of quick-sort to run in $\Theta(n^2)$ time.

The kind of sequence that would cause the deterministic version of the quick-sort algorithm to run in $\Theta(n^2)$ would be if the sequence was unsorted and the pivot that $\lfloor n/2 \rfloor$ chooses ends up being the maximum or minimum value in that sequence.

C-8.10

Give an example of a sequence of n integers with $\Omega(n^2)$ inversions. (Recall the definition of inversion from Exercise C-8.9.)

n, n-1, n-2, n-3, … , 2, 1

R-9.5

Show that the worst-case running time of quick-select on an n-element sequence is $\Omega(n^2)$.

If the sequence of elements that quick-select was given was something like (1,2,3,…,n) and had a key that was equal to n and if the first element was chosen as the pivot, in this case 1, then when the other numbers in the sequence were added to their respective sequences, L, E, or G, L would remain empty while G would contain n-1 elements. Meaning that the recursive call to quick-select for the elements greater than 1 would then pick the pivot 2 and the same process would continue giving us (n-1)(n-2)/2 comparisons or would have the complexity $\Omega(n^2)$.

C-9.1

Show that any comparison-based sorting algorithm can be made to be stable, without affecting the asymptotic running time of this algorithm.

Hint: Change the way elements are compared with each other.

This could be done by keeping track of the elements original position in the array. In doing so there will only be an increase of a constant factor, and won’t affect the asymptotic running time.