CPS 130 Homework 6 - Solutions

1. (CLRS 7.4-5) The running time of QUICKSORT can be improved in practice by taking advantage of the fast running time of INSERTION-SORT when its input is "nearly" sorted. When QUICKSORT is called on a subarray with fewer than k elements, let it simply return without sorting the subarray. After the top-level call to QUICKSORT returns, run INSERTION-SORT on the entire array to finish the sorting process. Argue that this sorting algorithm runs in $O(nk + n\lg(n/k))$ expected time. How should k be picked, both in theory and in practice?

   Answer: A complete proof would be similar with the proof of the average case running time of QUICKSORT (CLR section 7.4.2).

   The main idea is to note that the recursion stops when $\frac{n}{2^i} = k$, that is $i = \log_2 \frac{n}{k}$. The recursion takes in total $O(n \cdot \lg \frac{n}{k})$. The resulting array is composed of k subarrays of size n/k, where the elements in each subarray are all less than all the subarrays following it. Running INSERTION-SORT on the entire array is thus equivalent to sorting each of the $\frac{n}{k}$ subarrays of size k, which takes on the average $\frac{n}{k} \cdot O(k^2) = O(nk)$ (the expected running time of INSERTION-SORT is $O(n^2)$).

   If k is chosen too big, then the $O(nk)$ cost of insertion becomes bigger than $\Theta(n \lg n)$. Therefore k must be $O(\lg n)$. Furthermore it must be that $O(nk + n \lg \frac{n}{k}) = O(n \lg n)$. If the constant factors in the big-oh notation are ignored, than it follows that k should be such that $k < \lg k$ which is impossible (unless $k = 1$) - the error comes from ignoring the constant factors. Let c_1 be the constant factor in quicksort, and c_2 be the constant factor in insertion sort. Than k must be chosen such that $c_2 k + c_1 \lg \frac{n}{k} < c_1 \lg n$ which requires $c_1 k < c_2 \lg k$. In practice these constants cannot be ignored (also there can be lower order terms in $O(n \lg n)$) and k should be chosen experimentally.

2. (CLRS 7-3) Professors Dewey, Cheatham, and Howe have proposed the following "elegant" sorting algorithm:

   STOOGE-SORT(A, i, j)
   if $A[i] > A[j]$
       then exchange $A[i] \leftrightarrow A[j]$
   if $i + 1 \geq j$
       then return
   $k \leftarrow \lfloor (j - i + 1)/3 \rfloor$
   STOOGE-SORT$(A, i, j - k)$
   STOOGE-SORT$(A, i + k, j)$
   STOOGE-SORT$(A, i, j - k)$

   a. Argue that STOOGE-SORT$(A, 1, \mathit{length}[A])$ correctly sorts the input array $A[1..n]$, where $n = \mathit{length}[A]$.

   Solution: By induction:

For the base case let $n = 2$. The first two lines of the algorithm will check if the two elements are sorted; if not, it exchanges them (and now they are sorted). The algorithm returns after the following `if` statement. Thus STOOGE-SORT sorts correctly for $n = 2$.

Assume STOOGE-SORT correctly sorts an input array $A[1..k]$, where $k = length[A]$ and $1 \leq k < n$. In particular, STOOGE-SORT correctly sorts an input array of size $k = 2n/3$ (you may also assume STOOGE-SORT sorts correctly for $1 < k = 2n/3$). Let $A[1..n]$ be an input array of size $n = length[A]$. By the induction hypothesis the first call to STOOGE-SORT$(A, i, j - k)$ correctly sorts the first $2n/3$ elements, so that the elements $1 \ldots n/3$ are less than elements $(n + 1)/3 \ldots 2n/3$. The call to STOOGE-SORT$(A, i, j - k)$ correctly sorts the last $2n/3$ elements, so that the elements $(n + 1)/3 \ldots 2n/3$ are less than elements $2(n + 1)/3 \ldots n$, which are the largest $n/3$ elements in A. The last call to STOOGE-SORT$(A, i, j - k)$ sorts correctly (by induction hypothesis) the sorted elements are less than elements $2(n+1)/3 \ldots n$. Thus the array A of size n is sorted.

b. Give a recurrence for the worst-case running time of STOOGE-SORT and a tight asymptotic (Θ-notation) bound on the worst-case running time.

Solution:

$$\begin{aligned} T(n) &= 3T(\frac{2n}{3}) + \Theta(1) \\ &= \Theta(n^{\log_{3/2} 3}) \\ &= \Theta(n^{2.7\cdots}). \end{aligned}$$

c. Compare the worst-case running time of STOOGE-SORT with that of INSERTION-SORT, MERGE-SORT, HEAPSORT, and QUICKSORT. Do the professors deserve tenure?

Solution: STOOGE-SORT is the worst of all the algorithms – the professors do not deserve tenure.

INSERTION-SORT:	$\Theta(n^2)$
MERGE-SORT:	$\Theta(n \lg n)$
HEAPSORT:	$\Theta(n \lg n)$
QUICKSORT:	$\Theta(n^2)$
STOOGE-SORT:	$\Theta(n^{2.7\cdots})$