CS4311 Design and Analysis of Algorithms

Homework 2

Due: 11:10 am, March 20, 2008 (before class)

1. Our genius friend, John, has invented the following algorithm for sorting the array $A[i..j]$:

```
JohnSort(A, i, j)
 1. Set ℓ = j − i + 1,   k = ℓ % 3,   m = (ℓ − k)/3;
 2. if (k ≠ 0)
 3.   { Find the smallest k elements;
        Swap them with elements in A[i..i + k − 1];  }
 4. if (m == 0)   {   return;   }
 5. /* Else, sort the remaining 3m elements by recursion */
 6. JohnSort(A, i + k, j − m)      // Sort first 2m elements
 7. JohnSort(A, i + k + m, j)      // Sort last 2m elements
 8. JohnSort(A, i + k, j − m)      // Sort first 2m elements
```

   (a) (10%) Show that the above algorithm is correct.

   (b) (10%) Give a recurrence for the worst-case running time of `JohnSort`.

   (c) (10%) Obtain a tight asymptotic (Θ-notation) bound on the worst-case running time.[‡] How does it compare with the worst-case running time of insertion sort merge sort?

2. (15%) Illustrate the operation of `RadixSort` on the following list of English words:

   COW, DOG, SEA, RUG, ROW, MOB, BOX, TAB,
   BAR, EAR, TAR, DIG, BIG, TEA, NOW, FOX

3. (15%) Given an array $A[1..k]$ of k strings, with each string representing an integer. The characters in each string are chosen from $[0, 9]$. For instance, $A[i]$ may store "1203", which has four characters and represents the integer 1203.

   Suppose that the total length of *all* the strings is n. Also, assume that the length of each string can be found in $\Theta(1)$ time, and each character inside each string can be accessed in $\Theta(1)$ time. Show how to sort the strings in $O(n)$ time.

   Example: If the input array A is

   $$A = \{\text{"235"}, \text{"8"}, \text{"17"}, \text{"652"}, \text{"490"}, \text{"231562955"}, \text{"940"}, \text{"2"}\},$$

   then $n = 25$. After sorting, we should obtain an array

   $$\{\text{"2"}, \text{"8"}, \text{"17"}, \text{"235"}, \text{"490"}, \text{"652"}, \text{"940"}, \text{"231562955"}\}.$$

[‡] For simplicity, you may assume that $\lfloor 2n/3 \rfloor$ is equal to $2n/3$ when you solve the recurrence.

4. An $m \times n$ **Young tableau** is an $m \times n$ matrix such that (i) the entries of each row are in sorted order from left to right, and (ii) the entries of each column are in sorted order from top to bottom. Some of the entries of a Young tableau may be ∞, which we treat as nonexistent element. Thus, a Young tableau can be used to hold $r \leq mn$ finite numbers.

   (a) (10%) Draw three different 4×4 Young tableaux containing exactly the elements $\{9, 16, 3, 2, 4, 8, 5, 14, 12\}$.

   (b) (10%) Argue that an $m \times n$ Young tableau Y is empty if $Y[1, 1] = \infty$. Argue that Y is full (containing mn elements) if $Y[m, n] < \infty$.

   (c) (10%) Give an algorithm to implement `Extrac-Min` on a nonempty $m \times n$ Young tableau that runs in $O(m + n)$ time. (Think about `Extract-Min` in a heap.) Show that your algorithm is correct.

   (d) (10%) Suppose that inserting a new element into a nonfull $m \times n$ Young tableau can also be done in $O(m+n)$ time. Using no other sorting algorithm as subroutine, show how to use an $n \times n$ Young tableau to sort n^2 numbers in $O(n^3)$ time.

5. (Bonus: 10%)[§] Give an $O(m + n)$-time algorithm to determine whether a given number is stored in a given $m \times n$ Young tableau.

[§] Q5 is a bonus question. Total mark is calculated by: (Sum of Q1 to Q4) $\times$ (100%+ Q5).