ESO207A/211: Data Structures and Algorithms

Homework *3* **Due: September 10 2014**

Instructions.

1. Start each problem on a new sheet. Write your name, Roll No., the course number, the problem number, the date and the names of any students with whom you collaborated. Note that you may collaborate with other students but you have to write your own answers.
2. For questions in which algorithms are asked for, your answer should take the form of a short write-up. The first paragraph should summarize the problem you are solving and what your results are (time/space complexity as appropriate). The body of the essay should provide the following:
   (a) A description of the algorithm in English and/or pseudo-code, where, helpful.
   (b) At least one worked example or diagram to show more precisely how your algorithm works.
   (c) A proof/argument of the correctness of the algorithm.
   (d) An analysis of the running time of the algorithm.

   Remember, your goal is to communicate. *Full marks will be given only to correct solutions which are described clearly.* Convoluted and unclear descriptions will receive *low marks*.

Problem 1. Consider the following code for non-recursive quicksort. Here *Partition* is the standard procedure from quicksort and may or may not be randomized.

```
NR-Qsort(A, n) {
    Stack S;   // S is a stack of pairs of indices
    Push(S, (1, n))
    while S is not empty {
        (p, r) = Pop(S)
        q = Partition(A, p, r)
        if(q − 1 > p)
            Push (p, q − 1)
        if(r > q + 1)
            Push (q + 1, r)
    }
}
```

1. Give a (worst-case) scenario for the execution of **NR-Qsort** where the depth of the stack S is $\Omega(n)$.
2. Modify the loop so that the worst-case depth of the stack S is $O(\log n)$. Maintain the time complexity bound of standard quicksort.

Problem 2. Kendall tau distance between permutations. A permutation (or ranking) is an array of n integers where each of the integers between 1 and n appears exactly once. The Kendall tau distance between the two rankings is the number of pairs that are in different order in the two rankings. For example, the Kendall tau distance between 0 3 1 6 2 5 4 and 1 0 3 6 4 2 5 is 4 because the pairs $(0,1),(3,1),(2,4),(5,4)$ are in different relative order in the two rankings but all other pairs are in the same relative order.

1. Design an efficient $O(n \log n)$ time algorithm for computing the Kendall tau distance between a given permutation and the identity permutation (that is, 1 2 $\ldots n$). (***Hint:*** Modify Merge-sort.)

2. Extend the above algorithm to give an $O(n \log n)$ time algorithm for computing the Kendall tau distance between two permutations.

Problem 3. Lower bound for sorting with equal keys. Suppose that there are k distinct key values, with the ith key value occurring f_i times in the array A. There are a total of $n = f_1 + f_2 + \ldots + f_n$ keys. Show that any comparison sorting algorithm must make at least $nH - n$ comparisons, where, H is the Shannon entropy defined as

$$H = -\left(p_1 \log_2 p_1 + p_2 \log_2 p_2 + \ldots + p_k \log_2 p_k\right)$$

and $p_i = f_i/n$. (***Hint:*** You can use the fact that the number of arrangements with k distinct keys and with the ith key value occurring f_i times is $\frac{n!}{f_1! f_2! \ldots f_k!}$. Now revisit the comparison sort lower bound done in class.)