CMPT 307 (Fall 2013)

Practice 6(December 6, 2013)

- **stable sorting algorithms:** maintain the relative order of records with equal keys
- **in place algorithms:** need only $O(logn)$ extra memory beyond the items in an array of size n being sorted and they don't need to create auxiliary locations for data to be temporarily stored.

1. Given a set S of n elements and an index $k(1 \leq k \leq n)$, we define the k-smallest element to be the k-th element when the elements are sorted from the smallest to the largest.

   Suggest an $O(n)$ on average time algorithm for finding the k-smallest element.

2. Given an array A of n numbers, suggest an $O(n)$ expected time algorithm to determine whether there is a number in A that appears more than $n/2$ times.

3. n records are stored in an array A of size n. Suggest an algorithm to sort the records in $O(n)$ (time) and no additional space in each of the following cases:

   (a) All the keys are 0 or 1

   (b) All the keys are in the range $[0 \ldots k]$, k is constant

4. Given the following algorithm to sort an array A of size n:

   (a) Sort recursively the first 2/3 of $A : A[1 \ldots 2n/3]$

   (b) Sort recursively the last 2/3 of $A : A[n/3 + 1 \ldots n]$

   (c) Sort recursively the first 2/3 of $A : A[1 \ldots 2n/3]$

   Prove the above algorithm really sorts A and find a recurrence $T(n)$, expressing its running time.

5. How can we use an unstable sorting algorithm U (for example, quicksort or heapsort) to build a new stable sorting algorithm S with the same time complexity as the algorithm U?

6. Design an algorithm to report k smallest entries in an array in linear expected time where k is at most $n \log n$.