**Data Structures and Algorithms : CS210A/CS210A**

# Practice-sheet 7

**Date:** *17 November, 2014.*

**Note:** While designing algorithms for these problems, you have to use **word RAM** model of computation.

## 1 Integer sorting

1. Describe an algorithm that, given an array $A$ storing $n$ integers in the range $[0, k-1]$, builds a data structure that can answer any query of the following form in $O(1)$ time:
   `Report_range_size`$(a, b)$: return the number of integers stored in $A$ that lie in the range $[a, b]$.
   The computation time to build the data structure and the space occupied by it should be $O(n+k)$.

2. Show how to sort an array $A$ storing $n$ integers in the range $[0, n^k]$ in $O(kn)$ time.

3. You are given an array $A$ storing $n$ records with keys which are integers in the range $[n, 2n-1]$. Moreover, there are no two records in $A$ with the same keys. Design an $O(n)$ time algorithm to sort the array $A$. Your algorithm may use only $O(1)$ extra space.

## 2 Hashing

1. Consider the hash function $h(i) = i \bmod n$. Assume the universe size, $m > n^2$.
   - Describe a set $S \subset U$ of size $n$ for which the function $h$ is too bad: all elements of $S$ are hashed into the same location in the hash table.
   - Describe a set $S \subset U$ of size $n$ for which the function $h$ is perfect: all elements of $S$ are hashed into the different locations in the hash table.

2. What problem will arise if the hash table $T$ is just a Boolean array such that $T[i] = TRUE$ if and only if there is some element in $S$ with hash value $i$ ?

### Miscellaneous Problems

1. Recall the second programming assignment ($n$ Queens problem). You were asked to study about the relationship between a stack and recursion. You would have realized that each recursive procedure uses a stack implicitly. In the class, we said that Quick sort is in-place sorting since it uses $O(1)$ extra space (a couple of variables). But it is not correct to say this if we consider the extra space used implicitly by Quick sort for recursion. In view of this fact, it is the Heap sort (and not Quick sort) which is in-place sorting. This makes us wonder how much extra space might Quick sort be

using due to recursion. Similarly, how much extra space might Merge sort be using due to recursion ? What about other recursive algorithms we designed in the course ?

2. Let $G = (V, E)$ be a given weighted undirected graph on $n$ vertices and $m$ edges. Design an efficient algorithm to compute a spanning tree $T$ of $G$ such that the weight of the maximum weight edge of any other spanning tree of $G$ is not less than the weight of the maximum weight edge of $T$.