COMP 5711: Advanced Algorithm
2019 Fall Semester
Assignment 3

MR 8.22 (30 pts) In this problem we consider a simpler family of hash functions $H = \{h_a : a \in [p], a \neq 0\}$, where $h_a(x) = (ax \bmod p) \bmod n$. Here p is still a prime between u and $2u$. Show that the collision probability of H is at most $2/n$, i.e., for any $x, y \in [u], x \neq y$, $\Pr_{h \in H}[h(x) = h(y)] \leq 2/n$. This means that H is also a universal family of hash functions, up to a factor of 2.

KT 13.14 (25 pts) Suppose we have a set of k *basic processes* and want to assign each process to run on one of two machines, M_1 and M_2. There are n *jobs*, and each job requires exactly $2n$ of these basic processes to be running (each on its assigned machine). We say that an assignment is *nearly balanced* if for each job, no more than $\frac{4}{3}n$ of the basic processes associated with that job have been assigned to the same machine. Design a randomized polynomial-time algorithm that finds a nearly balanced assignment. You may assume that n is sufficiently large. (Indeed, if n is less than any constant, you can solve the problem with brute force.)

KT 13.15 (40 pts) (Note that this problem is slight different from the one in the textbook.) Suppose you have an array S of n real numbers, and you'd like to approximate the median by sampling (with replacement). You may assume that all numbers in the array are distinct. We will say that a number x is an ε-*approximate median* of S if at least $(\frac{1}{2} - \varepsilon)n$ numbers in S are less than x, and at least $(\frac{1}{2} - \varepsilon)n$ numbers in S are greater than x. After obtaining a sample, you will simply return the the median of the sampled numbers. How large should the sample be if we want the output to be indeed an ε-approximate median with probability at least $1 - \delta$? Express your asymptotic bound on the sample size, but do not treat ε and δ as constants (i.e., they are asymptotically small). What if we use a pairwise independent hash function to sample the locations of the array?

RIC (25 pts) Consider the following sorting algorithm that is based on the *randomized incremental construction* framework. We first randomly permute all the n elements to be sorted, and then insert them into a sorted list (which is initially empty) one by one. Of course, if we build a binary search tree on this list, then we can insert each element in $O(\log n)$ time, and the total running time is $O(n \log n)$. But this is not a randomized algorithm and we do not actually need the random permutation at all. Here we consider a different way to perform each insertion. For each element x yet to be inserted, we maintain a pointer to the element y in the list such that x should be inserted after y. Then for each element y, we maintain a list of pointers pointing to all such x's. Give the details on how to maintain these pointers after an element has been inserted into the list, and analyze the expected running time of this algorithm.