Problem 1

Suppose you are given two sets $S$ and $T$ of $n$ positive integers each. You want to determine if $S = T$, i.e. if they contain exactly the same elements. Design an algorithm to solve this in $O(n)$ expected time.

Problem 2

You are given two sets $S_1$ and $S_2$ stored as 2-3 trees which are augmented with order information, as described in class. You are told that $S_1 \subseteq S_2$ and that $|S_2| = n$ and $|S_1| = n - 1$, i.e. the two sets are equal except that $S_2$ contains exactly one additional element $x$, so $S_1 = S_2 - \{x\}$. Design an algorithm which finds this missing element $x$, which occurs in set $S_2$ but not $S_1$. Your solution should run in time $O(\log^2(n))$. (Hint: Use the fact that you are given the value of $n$.)

Problem 3

CLRS Problem 11-2.

Problem 4: Another universal family

Let $m$ be a prime number and $t$ a positive integer. Let the universe of keys, $U$, be given by $(t+1)$ tuples $(u_0, u_2, \ldots, u_t)$ where $u_i \in \mathbb{Z}_m$. Define the set of hash keys, $\mathcal{K}$, as $t$ tuples $(k_1, k_2, \ldots, k_t)$ where each $k_i \in \mathbb{Z}_m$. For each $k = (k_1, k_2, \ldots, k_t) \in \mathcal{K}$ and $u = (u_0, u_2, \ldots, u_t) \in U$, define the hash function

$$h_k(u) = u_0 + \sum_{i=1}^{t} u_i k_i.$$

Define

$$\mathcal{H} = \{h_k\}_{k \in \mathcal{K}}.$$

Prove $\mathcal{H}$ is a universal family of hash functions. (Hint: use the property given in class that in a field, the equation $ax = b$ has a unique solution $x$ given that $a \neq 0$.)

Problem 5: More Dynamic Programming

Let $S = \{x_1, x_2, \ldots, x_n\}$ be a set of $n$ integers and let $M = \sum_{i=1}^{n} x_i$. Design a dynamic programming algorithm to partition $S$ into two subsets $S_1$ and $S_2$ such that the sum of the elements in each subset is exactly $M/2$ or determine that it is impossible to do so. You algorithm should run in time $O(nM)$. Note: For the problem, you should first give the relationship between a problem it subsequent subproblems. In addition, I expect an algorithm to be given which not only implements the above recurrence but actually computes the elements in each set $S_1$ and $S_2$. The trick here is to use an additional table to store information for each element. Without loss of generality, you may assume that $M$ is even, otherwise clearly there is no solution.

---

1 You may work with a partner, but you must submit your own solutions. You may NOT consult any textbook other than CLRS nor the internet to aid in solving these problems.