# CSCI2100: Regular Exercise Set 7

Prepared by Yufei Tao

Problems marked with an asterisk may be difficult.

**Problem 1.** Let $S_1$ and $S_2$ be two sets of integers (they are not necessarily disjoint). We know that $|S_1| = |S_2| = n$ (i.e., each set has $n$ integers). Design an algorithm to report the *distinct* integers in $S_1 \cup S_2$ using $O(n)$ expected time. For example, if $S_1 = \{1, 5, 6, 9, 10\}$ and $S_2 = \{5, 7, 10, 13, 15\}$, you should output: 1, 5, 6, 7, 9, 10, 13, 15.

**Problem 2 (No Single Hash Function Works for All Sets).** Let $U$ and $m$ be integers satisfying $U \geq m^2$. Fix a hash function $h$ from $[U]$ to $[m]$, where $[x]$ represents the set of integers $\{1, 2, ..., x\}$. Prove that there must be a set $S \subseteq [U]$ such that (i) $|S| = m$, and (ii) $h$ maps all the elements of $S$ to the same hash value.

**Problem 3*.** Let $S$ be a multi-set of $n$ integers. Define the *frequency* of an integer $x$ as the number of occurrences of $x$ in $S$. Design an algorithm to produce an array that sorts the *distinct* integers in $S$ by frequency. Your algorithm must terminate in $O(n)$ expected time. For example, suppose that $S = \{75, 123, 65, 75, 9, 9, 65, 9, 93\}$. Then you should output $(123, 93, 65, 75, 9)$. Note that if two integers have the same frequency, their relative ordering is unimportant. For example, $(93, 123, 75, 65, 9)$ is another legal output.

**Problem 4*.** Let $S$ be a set of $n$ key-value pairs of the form $(k, v)$, where $k$ is the key and $v$ is the value. Preprocess $S$ into a data structure so that the following queries can be answered efficiently. Given a pair $(q_k, q_v)$, a query

- Returns nothing if $S$ contains no pair with key $q_k$;
- Otherwise, it returns the number of pairs $(k, v) \in S$ such that $k = q_k$ and $v \leq q_v$.

Define the *frequency* of a key $k$ as the number of pairs in $S$ with key $k$. Define $f$ as the maximum frequency of all keys. Your structure must use $O(n)$ space, and answer a query in $O(\log f)$ expected time. Furthermore, it must be possible to construct the structure $O(n \log f)$ time.

For example, suppose that $S = \{(75, 35), (123, 6), (65, 32), (75, 22), (9, 1), (9, 10), (65, 74), (9, 8), (93, 23)\}$. Then, given $(63, 33)$, the query returns nothing. Given $(65, 33)$, the query returns 1. Given $(65, 2)$, the query returns 0. In this example, $f = 3$.

**Problem 5** (Dynamic Hashing).** Consider the following *dynamic dictionary search* problem. Let $S$ be a dynamic set of integers. At the beginning, $S$ is empty. We want to support the following operations:

- `Insert`$(e)$: Adds an integer $e$ to $S$.
- `Delete`$(e)$: Removes an integer $e$ from $S$.
- `Query`$(q)$: Determines whether $q$ belongs to the current set.

Design a data structure with the following guarantees:

- At all times, the space consumption is $O(|S|)$, i.e., linear to the number of elements currently in $S$.
- For any sequence of $n$ operations (each being an `insert`, `delete`, or `query`), your algorithm must use $O(n)$ expected time in total.