# Practice Sheet I: Hashing

COL106: Data Structures and Algorithms

Semester-I 2023–2024

## 1 Problem I: Basic Concepts

Given input $\{4371, 1323, 6173, 4199, 4344, 9679, 1989\}$ and a hash function $h(x) = x(mod\ 10)$, show the resulting:

1. Separate chaining hash table
2. Hash table using linear probing
3. Hash table using quadratic probing
4. Hash table with second hash function $h_2(x) = 7 - (x\ mod\ 7)$

## Problem II: Needle in a Haystack

Suppose you are given a pattern text called `needle` of length $k$ and you need to find the first occurrence of this string in a large text `haystack` of size $n$.

1. Think of a simple algorithm to solve this problem. What is its time complexity?
2. Suppose you have an efficient hash function for hashing strings. If you are allowed to have a *few* false positives of `needle` in the `haystack`, can you think of an approach to solve this problem in a faster way? Analyse the time complexity of your approach.
3. Prove that the expected number of false positives will be *few*.

*Hints:*

1. Brute force approach: Slide the string `needle` over the string `haystack`. Time taken $= O(nk)$
2. Still slide the `needle` over the `haystack`, but compare the hash values of the key $k$ with the hash values of the window you are checking against. For this to be efficient, you need your hash function to be fast, such that you can calculate the next hash from the previous hash in $O(1)$ time. This is called a *rolling hash.*
3. The probability of false positives is equal to the probability of collisions of the hash function you choose. So, if the hash function is good, the probability will be low.

## Problem III: Hashing and Probability

Let $U = [1, M]$ be a universe, and let $S = \{s_1, \cdots, s_n\}$ be a subset of $U$ of size n such that each $s_i$ is a uniformly random element of $U$ independent of other $s_j$'s. We are implementing a hash table $T$ with chaining. $T[i]$ represents the $i$th chain. Let $H$ be a hash function such that $H(x) = x\ mod\ n$.

1. Show that the expected size of $max_{i=0}^{n-1} T[i]$ is $O(logn)$.
   *Note*: The expected value of any random variable $X$ is defined as,

$$E(X) = \sum_x xP(X = x)$$

2. Argue that the expected value of maximum time taken to verify the membership of elements of $U$ in $S$ is $O(logn)$

*Hints:*
Let $X = max_i\ T[i]$. Note we just want an *upper bound* on $E(X)$. Here are some questions for you to think about:

- Consider the event $E_i$, which is that the size of the ith chain is greater than some constant $k$. Can you find an upper bound on $P(E_i)$?
- Consider the event $E = \cup_i E_i$. What does this event signify? Can you find (an upper bound on) $P(E)$?
- Since the maximum value of $X$ is $n$, convince yourself that you can bound the $E(X)$ term as the following:
$$E(X) \leq nP(E) + kP(E^c)$$
- Choose an appropriate value of $k$ and complete the proof.

## Problem IV: Amortized Analysis of Open Addressing

Consider a simple open addressing scheme, let's say linear probing with a hash code $f_0(x)$. We start with an array of size $n$, use hash function $f_n(x) = f_0(x)\ mod\ n$. When this array gets full we move to an array of size 2n with hash function $f_{2n}(x) = f_0(x)\ mod\ 2n$. When this gets full we again double the size and so on. Clearly a single insert could take a long time if rehashing is to be done. Show that the amortized insert time is $\theta(1)$.

*Hints:* Use an accounting method over a series of hashtable operations, similar to the analysis of stacks using dynamically sized arrays. You may want to refer to the analysis presented here.

## Problem V: Cubic Probing

Suppose instead of quadratic probing, we use "cubic probing"; here the $i$th probe is at $hash(x) + i^3$. Do you think cubic probing improves on quadratic probing?

*Hints:* The issues with both quadratic and cubic probing is that there is *secondary* clustering. A cubic will grow faster, but therefore will also get modded out faster (since the size of the hashtable is the same). There should not be any practical improvement in general.