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*.

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 ith 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 $O(logn)$

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 $f_{2n}(x) = f_0(x)\ mod\ n$. 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 $\theta(1)$.

Problem V: Cubic Probing

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