14.1 Overview

In this lecture, we covered the following topics:

1. Load balancing with power-of-2-choice
2. Basics of hashing
3. Universal hash functions
4. Application on hash tables

14.2 Load balancing with power-of-2-choice

Suppose we have N balls and N bins. In the method of power-of-2-choice for load balancing, we consider a ball and pick two random bins, then we put the ball in the less loaded bin.

Theorem 14.1. In power-of-2-choice process, the max loaded bin has $O(\log\log N)$ balls with probability at least $1 - O(1/N)$.

Proof (intuition) First, let get the probability of an incoming ball getting height 3. To get the height 3 of a bin, we must first have picked two bins both with height 2. We know that the fraction of bins with 2 balls is at most $1/2$ because there are at most N balls for N bins. So the probability of a height 3 bin is $1/4$. Likewise, to get a height 4 bin we need two height 3 bins and so the probability is $1/16$, and so on. Therefore, the probability of a bin getting height $h$ is $1/2(2^{2h-2})$. Choosing $h = O(\log\log N) + 2$ gives the probability $O(1/N)$.

14.3 Hashing basics

Let's introduce some basics of hashing. Suppose we have a larger set (or "universe") of keys denoted by $U$. The actual dictionary is some subset $S$ that has size much smaller than $U$, $|S| << |U|$. Basically, hashing is a way to map elements of $U$ onto smaller number of values such that with high probability there are not too many collisions among elements of $S$. Some of the operations we want to do with hashing is:
A natural way to implement hashing is to choose a hash function $h : U \rightarrow [M]$. In other words, we map every $x \in U$ to $[M]$ with $h(x)$. We want the hash function to have the following desired properties:

1. Small probability of distinct keys colliding, that is if $x \neq y \in S$, $P(h(x) = h(y))$ is small.
2. Small range, i.e., small $M$ so that the hash table is small.
3. Small number of bits to store $h$.
4. $h$ is easy to compute.

To reduce hash collision, we can either (1) pick random input or (2) choose random hash functions. The first approach is not very practical because the keys are usually not random. So we will focus on the second approach in the next sections.

### 14.4 Ideal hash functions

We define the perfectly random hash function as that for each $x \in S$, $h(x)$ uniformly random location in $[M]$. Apparently, such hash functions have the desired properties:

- Low collision probability: $P[h(x) = h(y)] = 1/M$ for any $x \neq y$
- Even conditioned on hashed values for any other subset $A$ of $S$, for any element $x \in S$, $h(x)$ is still uniformly random over $[M]$.

**Question** Problem with this ideal approach?

**Answer** For mapping each $x \in S$, we need to have space of $\log_2 M$ bits to store $M$. So such method requires large amount of storage space. On the other hand, computing $h$ is via a table lookup, which is clearly not efficient.

### 14.5 Universal hash functions

Let’s denote the size of $[M]$ as $M$. The definition of universal Hash functions is as follows:

**Definition** A family $H$ of hash functions mapping $U$ to $[M]$ is universal if for any $x \neq y \in U$, $P[h(x) = h(y)] \leq 1/M$.

We can see that universal Hash functions try to capture the basic property of non-collision.
14.5.1 A simple construction of universal hashing

Here, we show a simple example of universal hashing construction. Assume $|U| = 2^u$ and $|M| = 2^m$. One construction of universal hashing can be defined by an $m \times u$ matrix $A$ with random binary bits so that
\[ h(x) = Ax \]
where the arithmetic is modulo 2.

**Question** How many hash functions in this family?

**Answer** Because there are $m \times u$ elements in $A$, each element $\in \{0, 1\}$ and so there exist $2^{um}$ hash functions in total.

**Theorem 14.2.** The family of hash functions defined above is universal.

**Proof** Because $h$ is a linear map, if $h(x) = h(y)$ for any $x \neq y \in U$, then $h(x - y) = 0$. Let $z = x - y$ and so we want to show that the probability of $Az = 0$ is at most $1/M$. Denote the columns of $A$ to be $A_1, A_2, \ldots, A_u$. Then $Az = \sum_{i \in U} z_i A_i$. Let $z_{i^*} = 1$ be one of the non-zero element in $z$ and then we have $A_{i^*} = -\sum_{i \neq i^*} z_i A_i$. Because $A_i$ has $m$ random bits and the probability of each one matching the bit on the right is $1/2$, the probability of $Az = 0$ is thus $(1/2)^m = 1/M$.

14.6 Application: Hash table

One of the applications of hash functions is hash table, which is often used for dictionary data structures. One issue of Hash table, however, is that different keys can be mapped to the same location, known as hash collision. We need to have some ways to resolve the collision. In the following, we introduce a closed addressing method, called separate chaining, to deal with collision.

14.6.1 Separate chaining

In the approach of separate chaining, each location in the table stores a linked list with all the elements mapped to that location. Naturally, the look-up time at a location is proportional to the length of the linked list. Let’s dig deeper to understand the look-up time of separate chaining.

**First, we shall analyze the number of collisions.** Let $C(x)$ be the number of other elements mapped to the value where $x$ is mapped to and $|S| = N$.

**Question** What’s the expectation of $C(x)$?

**Answer** Because the probability of other key having the same hash value is $x$ is $1/M$, the expectation of $C(x)$ is $E(C(x)) = (N - 1)/M$. Hence, if $|M| = |S| = N$, we have $E(C(x)) = O(1)$ and thus constant look-up time.
Question Let $C$ be the total number of collisions. What’s the expectation of $C$?

Answer Because the probability of any two keys in $S$ having the same hash value is $x$ is $1/M$, the expectation of $C$ is $E(C) = \binom{n}{2} \frac{1}{M}$. Suppose $|M| = N^2$, then $E(C) \approx 1/2$. It means the expectation of one collision is less than 1! However, it requires a large space for $|M|$ which is undesirable.