Lecture 6: Hashing and hash functions

Marina Meilă
mmp@stat.washington.edu

Department of Statistics
University of Washington

April 2019

Implementing a dictionary with an array

- **Assume** that we need to store pairs (key, data)
- and that the operations to support are Search, Insert, Delete. We call such a data structure a **dictionary**.
- Let U be the universe of all possible keys.

- **First idea** Create array T with size $|U|$.
  - At address k in T, store NIL if no element with key k in dictionary.
  - Otherwise store a pointer to (k, data).
  - If multiple elements with same key are allowed, store a pointer to a linked list of these elements.

  - Run time (with no duplicate keys) is $\mathcal{O}(1)$ for all operations.
- Inefficient if $|U|$ very large (which is typical).

Second idea: hashing

- **Assume** that n the number of items in the dictionary is much smaller than $|U|$.
- **Second idea:** associate each T entry with a set of keys. **Hope** that only 1 of them is in dictionary.
- a **hash function** is

$$h : U \rightarrow \{0, \ldots m-1\} \tag{1}$$

- **Essential assumption** $h(k)$ can be computed in constant time.
- The array T has now size m.
- Element (key,data) is stored at h(key) in T.
- Run time: all operations still run in constant time!

Collisions

- When $h(k) = h(k')$ and both k, k' are keys for dictionary elements, we have a **collision**.
- Collisions can be handled with a linked list, as above.

- A **good** h is a function that ensures that the p of a collision is $\approx \frac{1}{m}$ when the keys are sampled uniformly from U $\alpha = \frac{n}{m}$, known as the **load factor**
- Let n_j be the number of elements at location j in T.

- Run time with collisions: For INSERT, assuming insertion at the head of the list $\mathcal{O}(1)$ as before. For DELETE, SEARCH, in the worst case, one has to traverse a list of length n_j when $h(k) = j$. Hence, the run time is $\mathcal{O}(1 + n_j)$.
- Consider now average run times (assuming keys sampled uniformly as before). Obviously, $E[n_j] = \frac{n}{m} = \alpha$. Hence run time for DELETE, SEARCH is on average $\mathcal{O}(1 + \alpha)$.

Examples of hash functions

- $h(k) = k \bmod m$ with m a prime
- $h(k) = \lfloor m \,\mathrm{frac}(kA) \rfloor$ with m any array length (in particular, m can be 2^p), and $A < 1$ a chosen parameter; frac represents the fractional part of a real number, and $\lfloor , \rfloor$ denote the integer part of a number, i.e $z = \underbrace{\lfloor z \rfloor}_{\mathbb{Z}} + \underbrace{\mathrm{frac}(z)}_{[0,1)}$.

A family of hash functions

- A **universal family of hash functions** is a set $\mathcal{H} = \{h : U \rightarrow \{0, \ldots m-1\}\}$ with the following properties
  1. For each $h \in \mathcal{H}$, $Pr[h(k) = j] \approx \frac{1}{m}$
  2. For each pair of distinct keys k, k', $|\{h \in \mathcal{H} \mid h(k) = h(k')\}| \leq \frac{|\mathcal{H}|}{m}$

An example of universal family

- Let $p > |U|$ be a large prime number
- Let $a, b \in \mathbb{Z}_p$, $a \neq 0$ and

$$h_{ab} \;=\; [(ak+b) \bmod p] \bmod m \tag{2}$$

- Then, the family

$$\mathcal{H}_{m,p} = \{h_{ab}, \text{ with } a, b \in \mathbb{Z}_p, \ a \neq 0\} \tag{3}$$

  is universal.