Design and Analysis of Algorithms

CS3230

Week 5

Hashing

Warut Suksompong

Assignments: Common mistakes

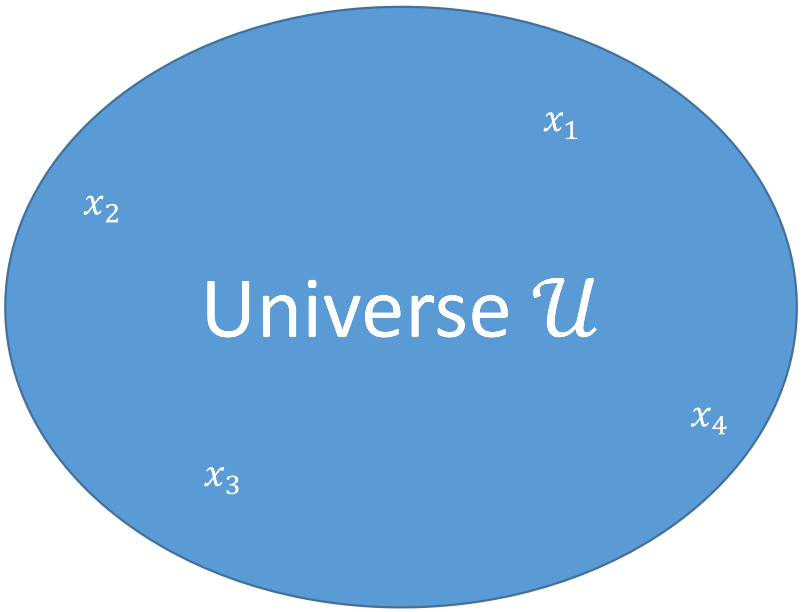
- Show that there exists an algorithm that makes at most k comparisons.
  - Upper bound: Giving one algorithm (that always makes at most k comparisons) suffices.
  - Lower bound: Need to rule out all possible algorithms that make at most k-1 comparisons. Usually done by an adversary argument.
- If $f(n) = O(g(n))$, is $2^{f(n)} = O\left(2^{g(n)}\right)$?
  - Here is a proof attempt.
  - There exist c, n_0 such that $f(n) \leq c \cdot g(n)$ for all $n \geq n_0$.
  - So $2^{f(n)} \leq 2^c \cdot 2^{g(n)}$ for all $n \geq n_0$.
  - Choosing $c' = 2^c$ and $n_0' = n_0$, we get $2^{f(n)} = O\left(2^{g(n)}\right)$.
  - What's wrong?

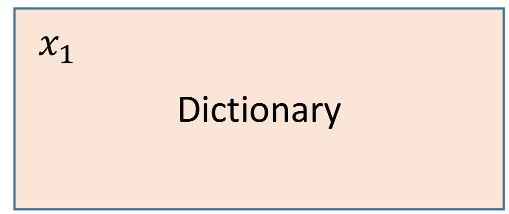
Asymptotic relations

- If $f(n) = \omega(g(n))$, then
  - $f(n) = \Omega(g(n))$
  - $f(n) \neq \Theta(g(n))$
  - $f(n) \neq O(g(n))$
  - $f(n) \neq o(g(n))$
- If $f(n) = o(g(n))$, then
  - $f(n) = O(g(n))$
  - $f(n) \neq \Theta(g(n))$
  - $f(n) \neq \Omega(g(n))$
  - $f(n) \neq \omega(g(n))$

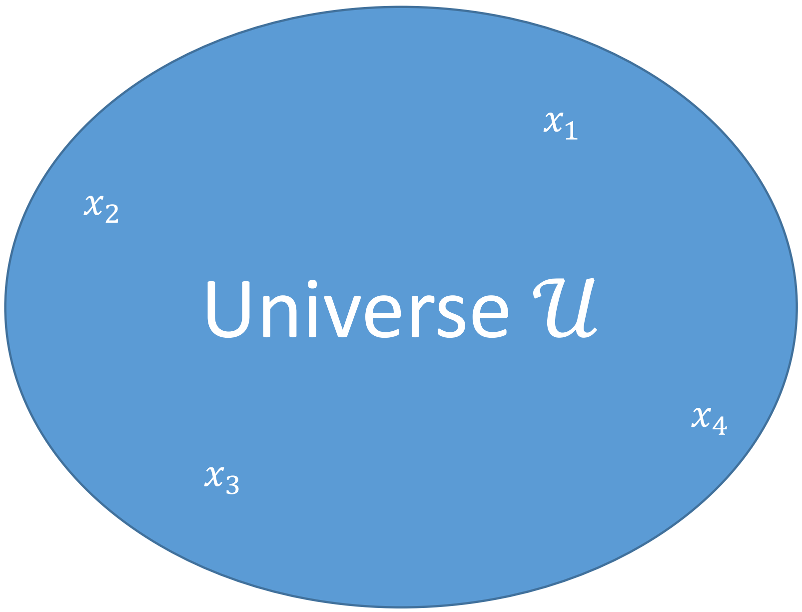
Dictionary Data Structure

$\texttt{add}(x_1)$





Dictionary Data Structure



`add`(x_1)

`add`(x_2)

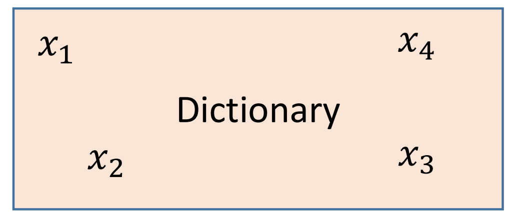
`add`(x_3)

`add`(x_4)

`delete`(x_2)

`query`(x_1) ✓

`query`(x_2) ✗



Dictionary Data Structure

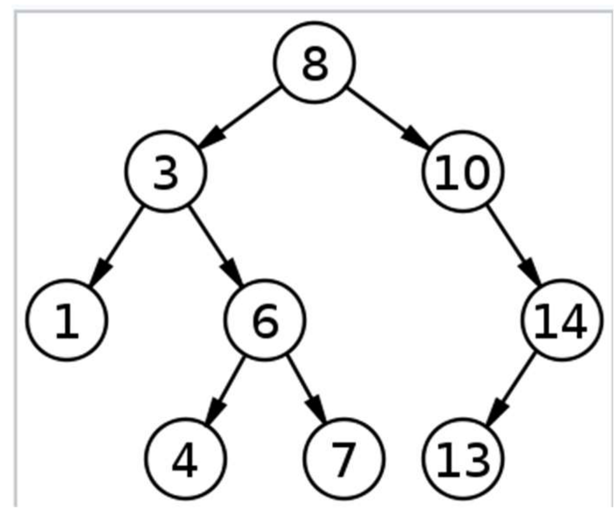
- The most popular data structure in computer science!

- Often, items inserted are (key, val) pairs. Inserting a key already in dictionary overwrites the val. Query returns val if key exists.
  - **Examples**: language dictionaries, compilers, virtual memory, network routers
  - Less obvious applications in searching and streaming covered next lecture

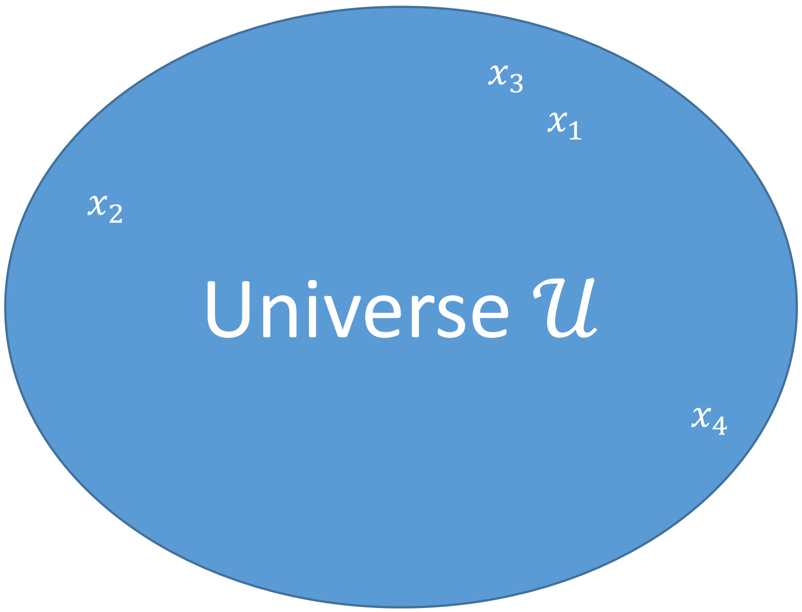
- **Static**: set of inserted items fixed; only care about queries.
  Insertion-only: Only insertions and queries.
  Dynamic: Insertions, deletions, and queries.

First Try

- In static case, can store items in a sorted list.
  - Query: $O(\log N)$, where N is number of stored items
- In dynamic case, can use balanced search tree structures
  - Insertion, deletion, query: $O(\log N)$ worst-case



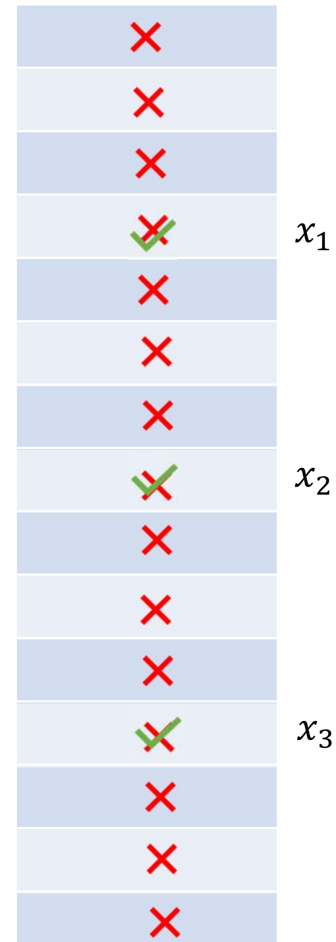
A different approach: Direct Access Tables



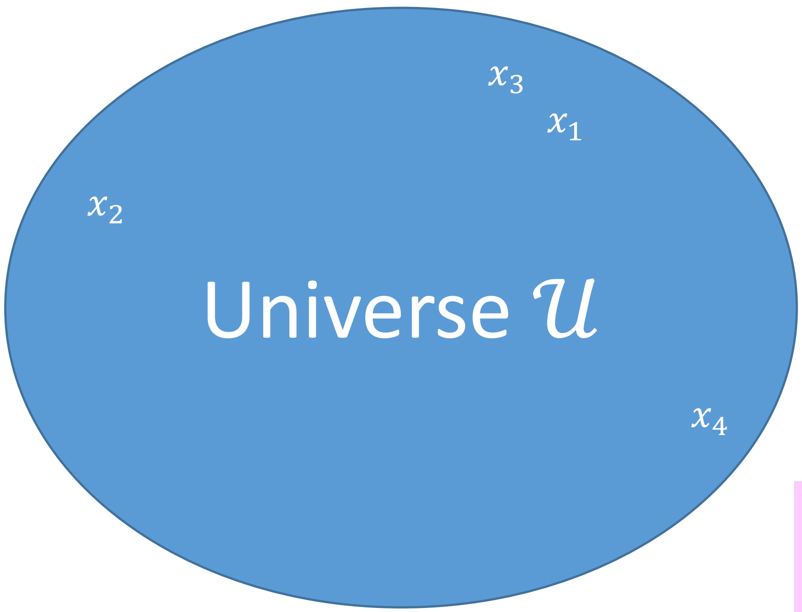
`add`(x_1)

`add`(x_2)

`add`(x_3)



A different approach: Direct Access Tables

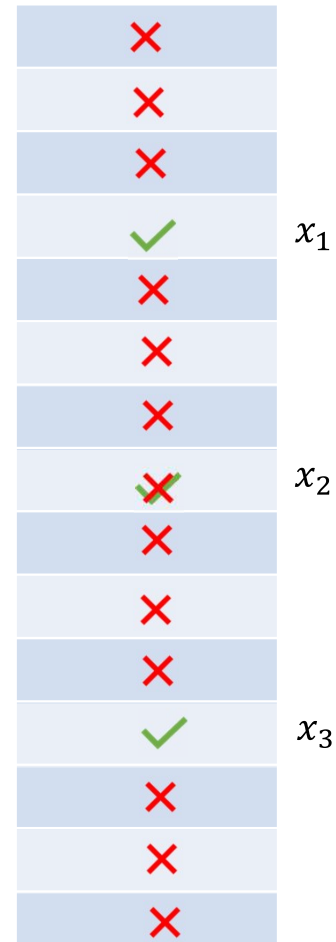


`add`(x_1)

`add`(x_2)

`add`(x_3)

`delete`(x_2)

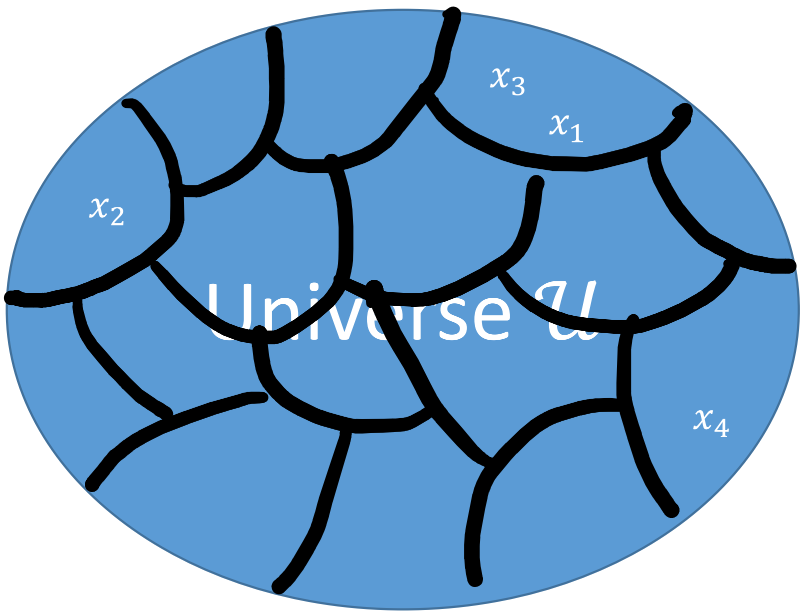


Note: Need items to be represented as non-negative integers (prehashing). Usually easy.

BIG PROBLEM: huge space requirement! Table size is that of the universe.

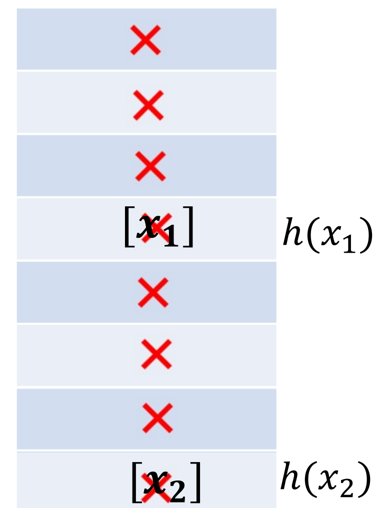
- Typical universe size: 2^{256} !!

Hashing



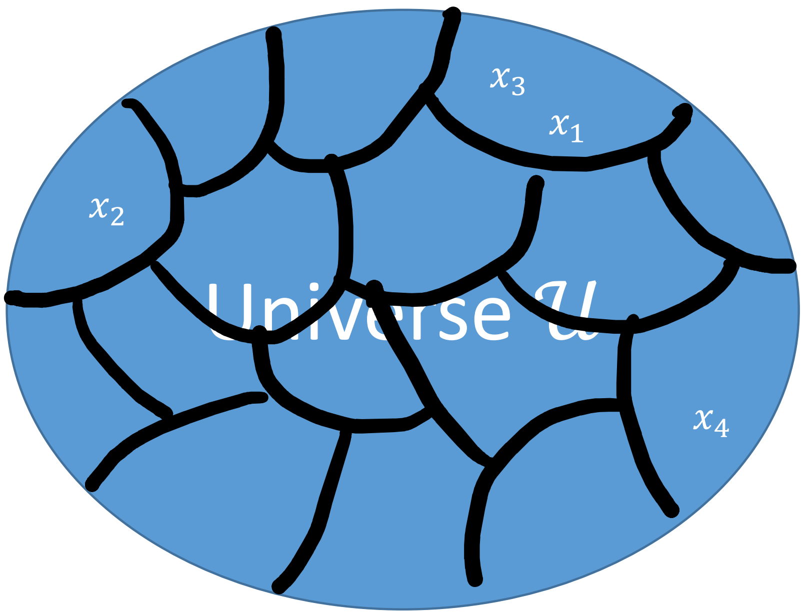
`add`(x_1)

`add`(x_2)



Hash Function: $h: U \to \{1, \ldots, M\}$ gives location of where to store in hash table.

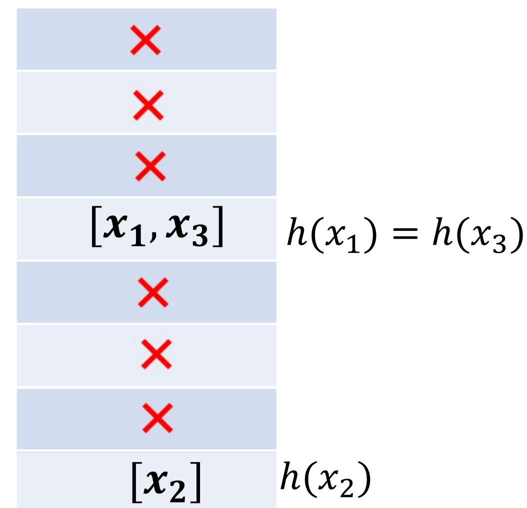
Hashing



`add`(x_1)

`add`(x_2)

`add`(x_3)



Hash Function: $h: U \to \{1, \ldots, M\}$ gives location of where to store in hash table.

A **collision** is when for two different keys x and y, $h(x) = h(y)$.
We resolve collisions by **chaining**. Other strategies possible, e.g., open addressing (see CLRS).

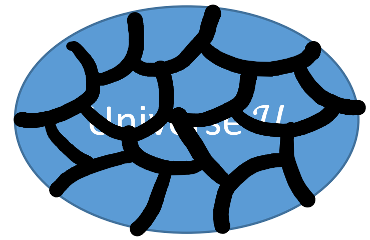
Desired properties

- Minimize collisions. `query`(x) and `delete`(x) take time $\Theta(|h(x)|)$. Worst case is when all inserted keys hash to the same location!

- Minimize storage space. Aim is to have $M = O(N)$. [Here and later, N is number of stored items.]

- The function h should be easy to compute. For this lecture, we will assume $h(x)$ computed in constant time, but in reality, this may be an issue.

Adversary strikes back!



- If U is large, then for any hash function with small M, there are many keys which all hash to the same location. 

- **Claim**: If $|U| \geq (N-1)M + 1$, for any $h: U \rightarrow [M]$, there is a set of N elements having the same hash value. Here and later, [M] denotes the set {1, 2, …, M}.

- **Proof**: Pigeonhole principle. If every slot in the hash table had $< N$ elements from U mapping to it, then $|U| \leq (N-1)M$. Contradiction!

Key Idea: Randomization

- Fool the adversary by not fixing the hash function!
- **Example**: Suppose $U = \{a, b, c\}$ and $M = 2$. Consider two hash functions h_1 and h_2.
  - $h_1(a) = 1, h_1(b) = 1, h_1(c) = 2$. **Note**: a and b collide.
  - $h_2(a) = 1, h_2(b) = 2, h_2(c) = 2$. **Note**: b and c collide.

If I randomly choose between h_1 and h_2, for any pair of keys, with probability $\geq \frac{1}{2}$, there will be no collision.

Each hash function by itself is not random!

Universal Hashing

Definition: Suppose $\mathcal{H}$ is a set of hash functions mapping U to $[M]$. We say $\mathcal{H}$ is ***universal*** if for all $x \neq y$:

$$\frac{|h \in \mathcal{H} : h(x) = h(y)|}{|\mathcal{H}|} \leq \frac{1}{M}.$$

# of hash functions for which x and y collide

For any $x \neq y$, if h is chosen uniformly at random from a universal $\mathcal{H}$, there's at most $\frac{1}{M}$ probability that $h(x) = h(y)$.

Universal Hashing Examples

	a	b
h_1	0	0
h_2	0	1

	a	b
h_1	0	1
h_2	1	0

	a	b
h_1	0	0
h_2	1	0
h_3	0	1

Universal

	a	b
h_1	0	0
h_3	1	1

	a	b	c
h_1	0	0	1
h_2	1	1	0
h_3	1	0	1

Not
Universal

Collision Analysis

Claim: Suppose $\mathcal{H}$ is a *universal* family of hash functions mapping U to $[M]$. For any N elements $x_1, \ldots, x_N$, the expected number of collisions between x_N and the other elements is $< \frac{N}{M}$.

PROOF

- Indicator random variable (see supplementary material)
- For $i < N$, let $A_i = 1$ if $h(x_i) = h(x_N)$ and 0 otherwise.
- $\mathbb{E}[A_i] = 1 \cdot \Pr[A_i = 1] + 0 \cdot \Pr[A_i = 0] = \Pr[A_i = 1] \leq \frac{1}{M}$.
- # of collisions with x_N is $\sum_{i<N} A_i$.
- $\mathbb{E}[\sum_{i<N} A_i] = \sum_{i<N} \mathbb{E}[A_i] \leq \frac{N-1}{M} < \frac{N}{M}$.

Expected Cost

Claim: Suppose $\mathcal{H}$ is a *universal* family of hash functions mapping U to $[M]$. For any sequence of N insertions, deletions and queries, if $M \geq N$, then the expected total cost for a random $h \in \mathcal{H}$ is $O(N)$.

PROOF

- Each operation costs $O(1)$ time in expectation by previous claim.
- By linearity of expectations, total cost is $O(N)$.

Construction of universal family

- But can we actually get a universal family of hash functions with $M = O(N)$?
  - **YES!**

- Suppose U is indexed by u-bit strings, and $M = 2^m$. For any binary matrix A with m rows and u columns:

$$h_A(x) = Ax \pmod 2$$

Claim: $\{h_A : A \in \{0,1\}^{m \times u}\}$ is universal.

Construction of universal family: Example

- Suppose $U = \{00, 01, 10, 11\}$, and $M = 2$.

	00	**01**	**10**	**11**
$\boldsymbol{h_{00}}$	0	0	0	0
$\boldsymbol{h_{01}}$	0	1	0	1
$\boldsymbol{h_{10}}$	0	0	1	1
$\boldsymbol{h_{11}}$	0	1	1	0

Proof of Correctness

- If $x \neq y$, what is $\Pr_A[Ax = Ay] = \Pr_A[A(x-y) = \mathbf{0}]$?

- Let $z = x - y$. We know $z \neq \mathbf{0}$. Need to show $\Pr_A[Az = \mathbf{0}] \leq \frac{1}{M}$.

- Special case: Suppose z is 1 at the i-th coordinate but 0 everywhere else. Then Az equals the i-th column of A. Since the i-th column is uniformly random, $\Pr[Az = \mathbf{0}] = \frac{1}{2^m} = \frac{1}{M}$.

Proof of Correctness

- Warm-up for general case: If you flip a fair coin independently k times, what is the probability that the number of times it comes up heads is even?

- General case: Suppose z is 1 at the i-th coordinate. See lecture notes or presentation.

Universal Hashing: Wrap-up

- Can use $\mathcal{H}$ for dictionaries. In addition to storing the hash table, dictionary also needs to store the matrix A.
  - Additional storage overhead $\Theta(\log N \cdot \log |U|)$ bits, if $M = \Theta(N)$.

- Other universal hashing constructions available, some with more efficient hash function evaluation.

Perfect Hashing

- Consider the static case: N fixed items in dictionary $x_1, x_2, \dots, x_N$.

- **QUESTION**: Can we do all queries in worst-case constant time?

Perfect Hashing: Quadratic Space

- Constant lookup time if no collisions.

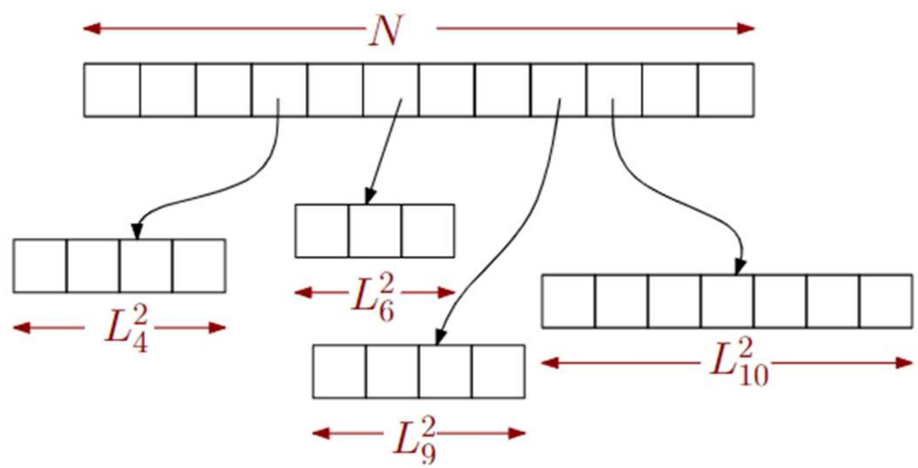
<u>Claim</u>: If $\mathcal{H}$ is *universal* and $M = N^2$, then if h is sampled uniformly from $\mathcal{H}$, the expected number of collisions is < 1.

PROOF

- For $i \neq j$, let A_{ij} equal 1 if $h(x_i) = h(x_j)$, and 0 otherwise.
- By universality, $\mathbb{E}[A_{ij}] = \Pr[A_{ij} = 1] \leq 1/N^2$.
- $\mathbb{E}[\#\text{collisions}] = \sum_{i \neq j} \mathbb{E}[A_{ij}] \leq \binom{N}{2} \frac{1}{N^2} < 1$.

There is a hash function $\boldsymbol{h}: \boldsymbol{U} \to [N^2]$ for which there are no collisions.

Perfect Hashing: 2-Level Scheme



- Choose $h: U \rightarrow [N]$ from a universal hash family.
- Let L_k be the number of x_i's for which

$$h(x_i) = k$$

- Choose $h_1, \ldots, h_N$ second-level hash functions $h_k: [N] \rightarrow [L_k^2]$ such that there are no collisions among the L_k elements mapped to k by h.
  - These exist because of the previous claim!
- **Question**: What is $\mathbb{E}\left[\sum_k L_k^2\right]$?

Perfect Hashing: 2-Level Scheme

Claim: If $\mathcal{H}$ is *universal,* then if h is sampled uniformly from $\mathcal{H}$:

$$\mathbb{E}\left[\sum_k L_k^2\right] < 2N.$$

- For $1 \le i, j \le N$, define $A_{ij} = 1$ if $h(x_i) = h(x_j)$ and $A_{ij} = 0$ otherwise
- **Crucial observation**:

$$\sum_k L_k^2 = \sum_{i,j} A_{ij}$$

- $\mathbb{E}\left[\sum_{i,j} A_{ij}\right] = \sum_i \mathbb{E}[A_{ii}] + \sum_{i \ne j} \mathbb{E}\left[A_{ij}\right] \le N \cdot 1 + N(N-1) \cdot \frac{1}{N} < 2N$

Acknowledgement

- The slides are modified from
  - the slides from Prof. David Woodruff
  - the slides from Prof. Arnab Bhattacharyya