

Dictionary Data Structure & Hashing

A69040872 Gaileng Chen

1 Basic Setup & Operations (Universe & Queries)

- **Universe:** Let there be a massive universe U with size $|U| = N$. We need to store a subset $S \subset U$ with a bounded size, meaning $|S| \leq n$.
- **Supported Queries:**
 - **Static:** Check if $x \in S$, and look up the data associated with x .
 - **Dynamic:** Insert or delete element x from S .

2 Hashing Idea & Pigeonhole Principle

- **Initial Idea:** Allocate an array A of size m . Design a mapping function $h : U \rightarrow \{1, \dots, m\}$, and store element x at $A[h(x)]$.
- **Core Problem (Collisions):** If there exist $x, y \in S$ such that $x \neq y$ but $h(x) = h(y)$, a collision occurs.
- **Pigeonhole Principle:** If $m < n$, collisions are inevitable. Thus, we cannot rely solely on a deterministic function h to perfectly solve the dictionary problem.

3 Collision Analysis & Hash Families

- **Expected number of collisions:** Assuming h is a completely random function, the expected number of collisions is:

$$\mathbb{E}[\# \text{ of collisions}] = \sum_{\substack{x, y \in S \\ x \neq y}} \Pr(h(x) = h(y)) \approx \frac{n^2}{2m}$$

- **Practical Problem:** We cannot store a completely random function in bounded memory.
- **Solution (Hash Family):** Introduce a k -wise independent hash family. In practice, we randomly choose a hash function from this family. For example, for a 2-wise independent family, the collision probability for any two distinct elements x, y is:

$$\Pr(h(x) = h(y)) = \frac{1}{m}$$

Note: We can construct this by using random polynomials of degree $< k$ over a finite field \mathbb{F} .

4 Strategies for Dealing with Collisions

Option A: Basic Chaining

Store all elements mapped to the same slot in a linked list.

- **Expected Time Complexity:**

$$\mathbb{E}[\text{time}] = O(1) + \mathbb{E}[\# \text{ of elements that collide with } x] = O\left(1 + \frac{n}{m}\right)$$

Option B: Perfect / Secondary Hashing

- **Goal:** $O(1)$ worst-case lookup time, $O(n)$ expected total memory.
- **Method:** In the primary hash table, suppose n_k elements map to slot k . To eliminate collisions in the secondary hash table, set its size to be quadratic, i.e., $m_k = n_k^2$. The probability of no collisions in this secondary table will be $> 50\%$.

- **Total Space Complexity Proof:**

Since $\sum_{k=1}^m n_k^2$ is equivalent to the number of pairs (x, y) such that $h(x) = h(y)$, we calculate its expectation:

$$\mathbb{E}\left[\sum_{k=1}^m n_k^2\right] = \sum_{x \in S} \Pr(h(x) = h(x)) + \sum_{\substack{x, y \in S \\ x \neq y}} \Pr(h(x) = h(y)) \leq n + \frac{n^2}{m}$$

Combining this with the size of the primary array m , the expected total space is:

$$\text{Total Space} = O(m) + O\left(n + \frac{n^2}{m}\right)$$

By setting $m = n$, the expected total space converges perfectly to $\mathbf{O(n)}$.