Hash Tables

Course: Design and Analysis of Algorithms

Laltu Sardar

Institute for Advancing Intelligence (IAI),
TCG Centres for Research and Education in Science and Technology (TCG Crest)

Jan 24, 2024

Hash Table

Previously Studied

- **Array** and **Linked List** to store a Set of elements.
- Insertion Time: $O(1)$ in Both
- Search Time: $O(N)$ in Both
- Deletion Time: $O(N)$ (As it requires to search before deletion)

Hash Table

Previously Studied

- **Array** and **Linked List** to store a Set of elements.
- Insertion Time: $O(1)$ in Both
- Search Time: $O(N)$ in Both
- Deletion Time: $O(N)$ (As it requires to search before deletion)

Question: **How to reduce search time?**

Hash Table

Previously Studied

- **Array** and **Linked List** to store a Set of elements.
- Insertion Time: $O(1)$ in Both
- Search Time: $O(N)$ in Both
- Deletion Time: $O(N)$ (As it requires to search before deletion)

Question: **How to reduce search time?**

- Solution: Hash Table
- An Effective data structure for dictionaries

Hash Table

Previously Studied

- **Array** and **Linked List** to store a Set of elements.
- Insertion Time: $O(1)$ in Both
- Search Time: $O(N)$ in Both
- Deletion Time: $O(N)$ (As it requires to search before deletion)

Question: **How to reduce search time?**

- Solution: Hash Table
- An Effective data structure for dictionaries
- It stores a value/element having some key: we say them key-value pair
- E.g.: To store a document: Content can be value while filePath is the key
- E.g.: In real dictionary: word => key and meaning => value
- In blockchain: block => value, blockhash => key

Direct Addressing

- A hash table generalizes the simpler notion of an ordinary array

When works?

Universe of keys is small.

How?

- Suppose key $k \in U = \{0, 1, \ldots, m-1\}$
- Take an array T of size m.
-

Direct Addressing

- A hash table generalizes the simpler notion of an ordinary array

When works?

Universe of keys is small.

How?

- Suppose key $k \in U = \{0, 1, \ldots, m-1\}$
- Take an array T of size m.
-

Then what?

- Insert: value x with k as $T[k] = x$. (O(1)) time.
- Search: just return T[k]. (O(1)) time.

Large Universe

- What if key universe is large?
- Suppose want to store Aadhaar info where key is 16-digit Aadhaar No.
- requires table of size $10^{16} \approxeq 2^{58}$
- Too large: Impractical
- Actual Number of Data: 2^{32}

Large Universe

- What if key universe is large?
- Suppose want to store Aadhaar info where key is 16-digit Aadhaar No.
- requires table of size $10^{16} \approxeq 2^{58}$
- Too large: Impractical
- Actual Number of Data: 2^{32}

For Large Universe

There may be collision

Hash Table

What to do for large universe:

- Consider Aadhaar Data:
- Take table of 2^{32} size.
- We require a **function** that maps keys to the indices of T.
- domain size (2^{58}) is larger than co-domain (2^{32})=> **Collision**

How to implement with collision

- Mixing Array and linked list => Chaining

Hash Table: Example

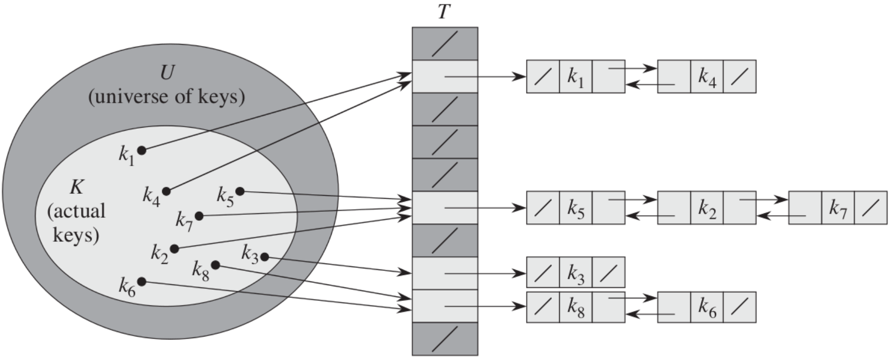

Figure 11.3 Collision resolution by chaining. Each hash-table slot $T[j]$ contains a linked list of all the keys whose hash value is j. For example, $h(k_1) = h(k_4)$ and $h(k_5) = h(k_7) = h(k_2)$.

Collision

What about: **Insertion & Search complexities?**

Worst case

- Insertion: $O(1)$, • Search: Worst case $O(N)$, unsuccessful search

Average case

- choice of function matters
- Each element in co-domain should have equal number of pre-images.
- Simple Uniform hashing
- Complexity of Successful search
  - Insertion: $O(1)$; Search: Worst case $O(N/m)$

Collision

What about: **Insertion & Search complexities?**

Worst case

- Insertion: $O(1)$, • Search: Worst case $O(N)$, unsuccessful search

Average case

- choice of function matters
- Each element in co-domain should have equal number of pre-images.
- Simple Uniform hashing
- Complexity of Successful search
  - Insertion: $O(1)$; Search: Worst case $O(N/m)$

tcg crest
Inventing Harmonious Future

Next Class:

- Open addressing:
- Linear probing:
- Quadratic probing
- Double hashing
- Analysis of open-address hashing
- Universal Hashing
- Perfect hashing

tcg crest
Inventing Harmonious Future

Deletion in a Hash Table

Deletion in Chaining

- Process: Just delete the node from the linked list
- Complexity: worst case – O(N)
- For uniform distribution, average case– O(N)/O(M)

Deletion in Open Addressing

- Process: mark as deleted –No actual deletion
- Complexity– worst case – O(N)
- For uniform distribution ..average case– O(N)/O(M)

Chaining or Open Addressing: What to choose?

Implementation

- Chaining: Linked list and Array– Each node in a linked list stores (key,value, nextPtr)– Array stores heads
- Open addressing: Only Array. Array contains (key,value)

chaining is preferred when

- the expected number of collisions is high or
- when the data is unpredictable and unevenly distributed

open addressing is preferred when

- memory usage is a concern
- when the data is uniformly distributed

Probing in Open Addressing:

$H(key, 0), H(key, 1), \ldots, H(key, n-1)$.. Until we find an empty slot
Where $H(key, 0) = h(key)//$ h is the actual hash function.

Linear Probing

- $H(key, i) = h(key) + i$
- Cluster Problem:

Quadratic probing

- $H(key, i) = h(key) + c_1.i + c_2.i^2$, c_1, c_2- constants
- Cluster reduced: cache problem increased

Double hashing

- $H(key, i) = h_1(key) + i.h_2(key)$, h_1, h_2- on same co-domain
- Cluster reduced further: cache problem increased further

Selecting Good Hash Function

How to improve: **Collision reduction**

Good hash function selection: How?

- What makes a hash function good?
- Simple Uniform hashing:
- Example: $h(k) = \lfloor km \rfloor$: When $0 \leq k < 1$

Selecting Good Hash Function

How to improve: **Collision reduction**

Good hash function selection: How?

- What makes a hash function good?
- Simple Uniform hashing:
- Example: $h(k) = \lfloor km \rfloor$: When $0 \leq k < 1$

Most hash function assumes key k as a Natural Number

1. – If not we find a way.
2. E.g., string key "tcg" –represented as $116.128^2 + 99.128 + 103$

tcg crest
Inventing Harmonious Future

Division Method

$$\boxed{h(k) = k \mod m}$$

Discussion

- When $m = 2^p$, $h(k)$ is just the p lowest-order bits of k
- Good choice of m: A prime not too close to an exact power of 2

Multiplication Method

$$\boxed{h(k) = \lfloor m(kA - \lfloor kA \rfloor) \rfloor} \text{ where } 0 < A < 1$$

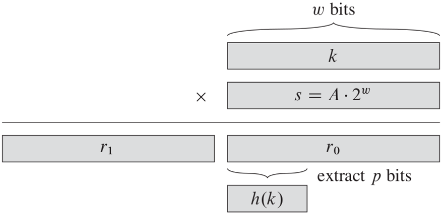

Figure 11.4 The multiplication method of hashing. The w-bit representation of the key k is multiplied by the w-bit value $s = A \cdot 2^w$. The p highest-order bits of the lower w-bit half of the product form the desired hash value $h(k)$.

- It is solving choice of m problem
- m can be chosen as 2^p form. It actually helps. How?

tcg crest
Inventing Harmonious Future

Universal class of hash functions

- What if some adversary chooses key such a way that all falls in a single bucket?
- Fixing some hash function is not enough.

Universal Hashing

A collection of hash functions that maps universe $\mathcal{U}$ of keys to $[n]$.

- For any pair $k_1.k_2 \in \mathcal{U}$, The number of hash function for which $h(k_1) = h(k_2)$ is at most $\mathcal{U}/m$.
- Hint: assume H_i is collection for which $h(k) = h(l) = i$.

Complexities while Chaining

- initially empty table with m slots
- any sequence of n INSERT, SEARCH, DELETE –>
- complexity-> $\Theta(n)$ where $n = O(m)$.

Example: Designing universal class of hashing

Example:

- $p(> m)$ is a large prime so that $0 \leq k < p$ every key
- $\mathbb{Z}_p = \{0, 1, \dots, p-1\}$; $\mathbb{Z}_p^* = \{1, 2, \dots, p-1\}$;
- choose $a \in \mathbb{Z}_p^*$ and $b \in \mathbb{Z}_p$
- Define $H_{ab}(k) = ((ak + b) \mod p) \mod m$
- $H_{ab} : \mathbb{Z}_p \to \mathbb{Z}_m$: count $= p(p-1)$

Other Example:

$h_k(x) = (k[0] + k[1]x + k[2]x^2 + ... + k[n-1]x^(n-1)) \mod p$ mod m; $m >> N$; $k[i]$ is an integer in some range $[0, N-1]$

tcg crest
Inventing Harmonious Future

Proof

- To prove that H is universal, we need to show that for any two distinct keys $k1$ and $k2$, the probability that $h(k1) = h(k2)$ for a randomly chosen hash function h in H is at most $1/m$.
- Let's assume that $k1$ and $k2$ are two distinct keys. We want to compute the probability that $h(k1) = h(k2)$ for a randomly chosen hash function h in H.
- We can express this probability as follows:
- $P(h(k1) = h(k2)) = P((ak1 + b) \mod p \mod m = (ak2 + b) \mod p \mod m)$
- Let's define $c = (ak1 + b) \mod p and d = (ak2 + b) \mod p$.

Proof: Part-II

- Note that c and d are both random integers chosen from the range $[0, p-1]$. Since p is prime and $k1$ is not equal to $k2$, we know that c and d are distinct with probability at least $(p-2)/p$.
- Now, let's consider two cases:
- Case 1: $c \mod m = d \mod m$
- If $c \mod m = d \mod m, thenwehaveh(k1) = h(k2)$. The probability of this event is:
- $P(c \mod m = d \mod m) = \sum_{i=0}^{m-1} P(c \mod p = i \text{ AND } d \mod p = i) = \sum_{i=0}^{m-1} P(c \mod p = i) * P(d \mod p = i) = \sum_{i=0}^{m-1} (1/p) * (1/p)$ (since c and d are chosen independently) $= m * (1/p^2)$

tcg crest
Inventing Harmonious Future

Proof: Part-III

- Since $p > m$, we know that $1/p^2 < 1/m$, so the probability of $h(k1) = h(k2)$ in this case is at most $1/m$.
- Case 2: $c \mod m \neq d \mod m$
- If $c \mod m \neq d \mod m$, then we have $h(k1) \neq h(k2)$. The probability of this event is:
- $P(c \mod m \neq d \mod m) = \sum_{i=0}^{m-1} P(c \mod p = i AND d \mod p \neq i) + \sum_{i=0}^{m-1} P(c \mod p \neq i AND d \mod p = i)$
  $= 2 * \sum_{i=0}^{m-1} P(c \mod p = i) * (1 - P(d \mod p = i))$
  $= 2 * \sum_{i=0}^{m-1} (1/p) * (1 - 1/p)$ (since c and d are chosen independently)
  $= 2 * m * ((p - 1)/p^2)$
- Since $p > m$, we know that $(p - 1)/p^2 < 1/p$, so the probability of h

Perfect hashing

Question

Can we achieve O(1) memory accesses are required to perform a search in the worst case.

- Yes: When the set of keys is static.
- Once the keys are stored in the table, the set of keys never changes.

Description

- Use two levels of hashing, with universal hashing at each level– primary & secondary
- Expected amount of memory used overall– $O(N)$ – How?

How to choose sizes?

tcg crest
Inventing Harmonious Future