

Week 10: Hash Table



Office: #N4-02c-86

College of Computing and Data Science

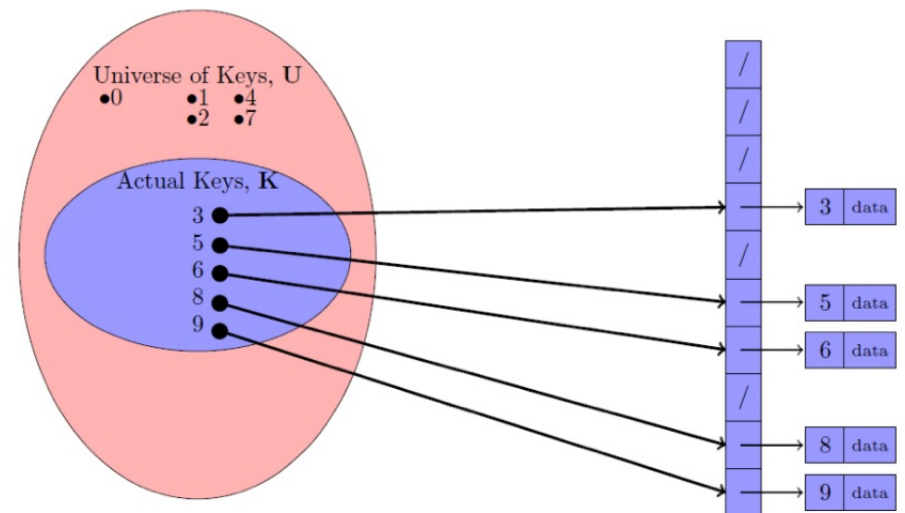
Problem

How to find and store data effectively?

Answer?

Direct-Address Table

- Assume that the keys of elements K drawn from the universe of possible keys U
- No two elements have the same key
- Search time is $O(1)$ but ...
 - The array size is enormous
 - $|U| \gg |K|$



Hashing

- Hashing: a typical space and time trade-off in algorithm
- To achieve search time in $O(1)$, memory usage will be increased

What is hashing?

- To reduce the key space to a reasonable size
- Each key is mapped to a unique index (hash value/address)
- Search time remains $O(1)$ on the average

hash function: $\{\text{all possible keys}\} \rightarrow \{0, 1, 2, \dots, h-1\}$

- The array is called a hash table
- Each entry in the hash table is called a hash slot
- When multiple keys are mapped to the same hash value, a collision occurs
- If there are n records stored in a hash table with h slots, its load factor is $\alpha = \frac{n}{h}$

Applications

- **Caching:** Hash tables are commonly used in caching applications to store frequently accessed data.
 - The hash table can be used to store key-value pairs, where the key is a unique identifier for the data and the value is the data itself.
- **Databases:** Hash tables are often used in databases to provide fast lookups of data.
 - For example, a hash table can be used to store a table of users, with the user ID as the key and the user's information as the value.
- **Counting:** Hash tables can be used to count occurrences of items in a dataset.
 - Each item can be stored as a key in the hash table, with the value representing the count of occurrences.
- **Cryptography:** Hash tables are used in cryptography to store password hashes.
 - When a user logs in, their password is hashed and compared to the hashed value stored in the hash table → secure authentication without storing the actual passwords in the hash table.

Hash Functions

- Must map all possible values within the range of the hash table uniquely
- Mapping should achieve an even distribution of the keys
- Easy and fast to compute
- Minimize collision

1. Modulo Arithmetic
2. Folding
3. Mid-square
4. Etc.

Hash Functions

1. Modulo Arithmetic: $H(k) = k \bmod h$

- E.g. $h = 13$ & $k = 37699 \rightarrow H(k) = 37699 \bmod 13 = 12$
- In practice, h should be a prime number, but not too close to any power of 2

2. Folding

- Partition the key into several parts and combine the parts in a convenient way
- Shift folding: Divide the key into a few parts and added up these parts
- $X = abc \rightarrow H(X) = (a + b + c) \bmod h$
- E.g. $H(123456789) = (123 + 456 + 789) \bmod 13 = 3$

3. Mid-square

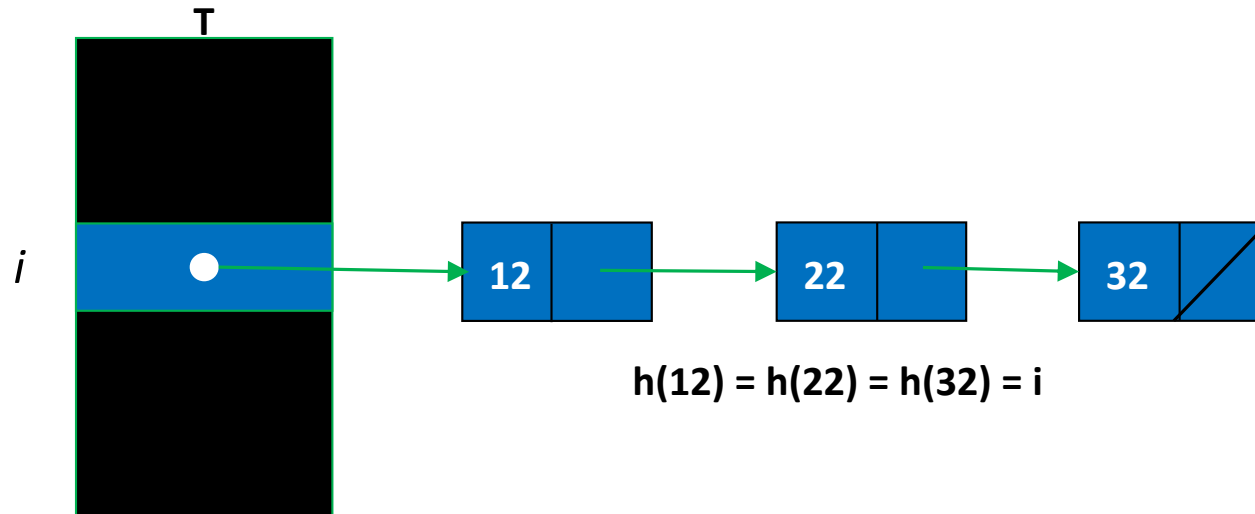
- The key is squared and the middle part of the result is used as the hash address
- E.g. $k=3121$, $k^2 = 3121^2 = 9740641 \rightarrow H(k) = 406$

Collision Resolutions

- Closed Addressing Hashing – a.k.a separate chaining
- Open Addressing Hashing
 - Linear Probing
 - Quadratic Probing
 - Double Probing

Closed Addressing: Separate Chaining

- Keys are not stored in the table itself
- All the keys with the same hash address are store in a separate list



- During searching, the searched element with hash address i is compared with elements in linked list $H[i]$ sequentially
- In closed address hashing, there will be α number of elements in each linked list on average $\alpha = \frac{n}{h}$

Closed Addressing: Separate Chaining

Time complexity in the **worst-case analysis**:

- When all elements are hashed to the same slot
- A linked list contains all n elements
- Its **unsuccessful search** takes n key comparisons, $\Theta(n)$
- Its **successful search**, assuming the probability of searching for each item is $\frac{1}{n}$
$$\frac{1}{n} \sum_{i=1}^n i = \frac{n+1}{2} = \Theta(n)$$
 - It is just like a sequential search

Closed Address Hashing: Separate Chaining

Time complexity in the **average-case analysis**:

- All elements are equally likely hashed into h slots.
- Its **unsuccessful search** takes $\frac{n}{h}$ key comparisons, $\Theta(\alpha)$
- Its **successful search**: $\frac{1}{n/h} \sum_{i=1}^{n/h} i = \Theta(\alpha)$
 - If α is constant (n is proportional to h), then $\Theta(1)$
 - Searching takes constant time averagely

Open Addressing

- Keys are stored in the table itself
- α cannot be greater than 1
- When collision occurs, probe is required for the alternate slot
 - Ideally, the probing approach can visit every possible slot

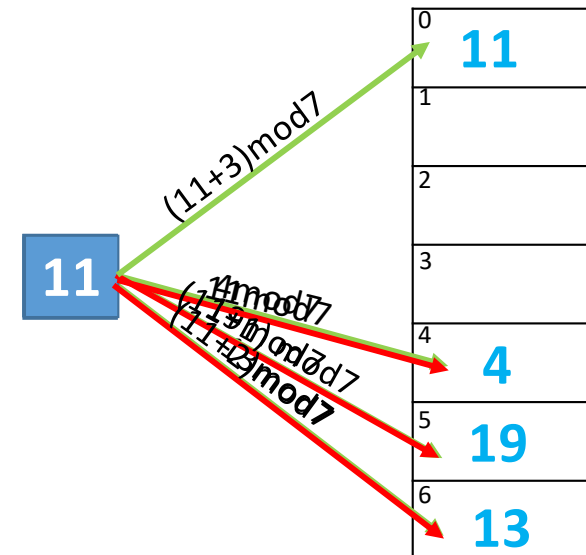
1. Linear Probing: probe the next slot

$$H(k, i) = (k + i) \bmod h \text{ where } i \in [0, h - 1]$$

$$\text{e.g. } H(k, i) = (k + i) \bmod 7$$
$$k \in \{4, 13, 19, 11\}$$

Primary clustering:

- A long runs of occupied slots
- Average search time is increased



Open Addressing

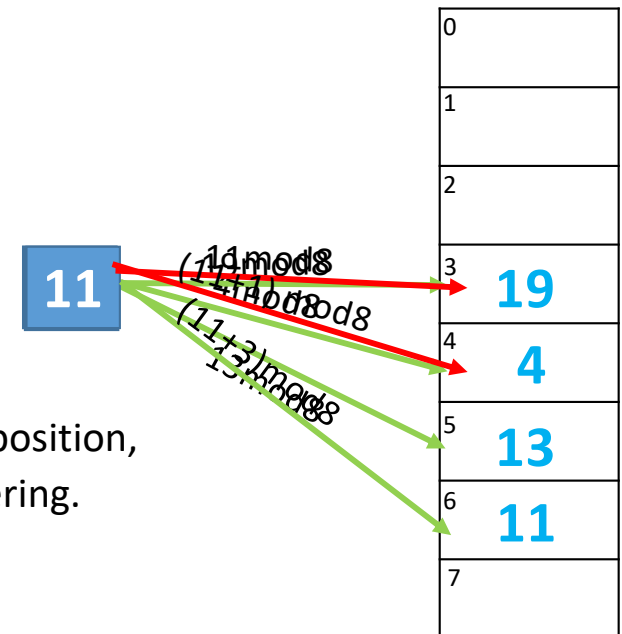
2. Quadratic Probing

$$H(k, i) = (k + c_1 i + c_2 i^2) \bmod h \quad \text{where } c_1 \text{ and } c_2 \text{ are constants, } c_2 \neq 0$$

- May not all hash table slots be on the probe sequence
- For $h = 2^n$, a good choice for the constants are $c_1 = c_2 = \frac{1}{2}$

e.g. $H(k, i) = \left(k + \frac{1}{2}i + \frac{1}{2}i^2\right) \bmod 8$
 $k \in \{4, 13, 19, 11\}$

$(\frac{1}{2}i + \frac{1}{2}i^2) \bmod 8$
1
3
6
2
7
5
4



- **Secondary Clustering**: if two keys have the same initial probe position, their probe sequences will be the same. This will form a clustering.
 - Inserting $k=3$ in the previous example.

Open Addressing

3. Double Hashing: a random probing method

$H(k, i) = (k + iD(k)) \bmod h$ where $i \in [0, h - 1]$ and $D(k)$ is another hash function

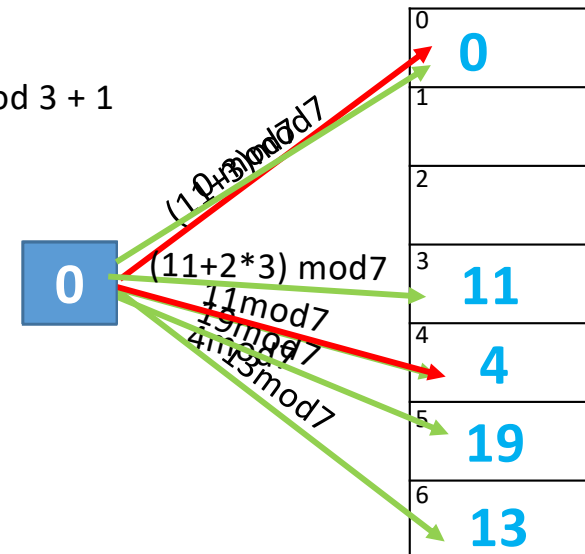
- The hash table size h should be a prime number

eg. $H(k, i) = (k + iD(k)) \bmod 7$

$$D(k) = (k) \bmod 3 + 1$$

$$k \in \{0, 4, 13, 19, 11\}$$

$$D(11) = 11 \bmod 3 + 1$$



Time Complexity

Linear Probing

- Successful Search: $\frac{1}{2} \left(1 + \frac{1}{1-\alpha}\right)$
- Unsuccessful Search: $\frac{1}{2} \left(1 + \left(\frac{1}{1-\alpha}\right)^2\right)$

Double Hashing

- Successful Search: $\frac{1}{\alpha} \ln \frac{1}{1-\alpha}$
- Unsuccessful Search: $\frac{1}{1-\alpha}$

*Proof can be found in The Art of Computer Programming by Knuth Donald (1973)

Delete A Key Under Open Addressing

- Leave the deleted key in the table
- Make a marker indicating that it is deleted
- Overwrite it when a new key is inserted to the slot
- May need to do a “garbage collection” when a large number of deletions are done
 - To improve the search time

Rehashing: Expanding the Hash Table

- As α increases, the time complexity also increases

Solution:

- Increase the size of hash table (doubled)
- Rehash all keys into new larger hash table

Summary

- Hash table: A typical space and time trade-off in algorithms
- Collision
 - Closed-address Hashing : Separate Chaining: $O(\alpha)$ on average
 - Open-address Hashing: $O(f(\frac{1}{1-\alpha}))$
 - Linear Probing
 - Quadratic Probing
 - Double Hashing
- Delete keys
- Rehashing

