Data Structures & Algorithms

Hashing

Hashing

Hash Tables : In hash table data structure, a hash function is used to compute an index, into an array from which the desired value can be found.

From Keys to Indices

A **hash function** h transforms a key into an index in a hash table T[0…k-1]:

$$h(x) : \rightarrow \{0, 1, \ldots, k - 1\}$$

We say that x **hashes** to slot h(x)

Ideally, the hash function will assign each key to a unique cell, but most hash table designs employ an imperfect hash function, which might cause hash collisions where the hash function generates the same index for more than one key.

Hashing

Properties/Requirements of Good Hash Functions

If the hash table has size k, hash function should return numbers in the range 0....k-1.

Hash function must be quick to calculate the value.

Hash function should minimize *collisions* (when several keys are hashed to the same number).

The last requirement is satisfied if each key is equally likely to map to any of the k indices (uniform distribution). Usually we cannot completely avoid collisions (as in the example with words collisions).

Hashing Methods

- Division method
- Folding method
- Mid-square method
- etc

The Division Method

Idea:

Map a key x into one of the k slots by taking the remainder of x divided by k

$$h(x) = x \bmod k$$

Example of the Division Method

Insert 36, 18, 72, 43, 6 into a hash table of size 8.

Hash key = key % table size

4 = 36 % 8

2 = 18 % 8

0 = 72 % 8

3 = 43 % 8

6 = 6 % 8

[0]	72
[1]	
[2]	18
[3]	43
[4]	36
[5]	
[6]	6
[7]	

The Folding Method

Idea:

- Break the key into pieces (sometimes reversing alternating chunks) and add them up.
- If the key is 123456789, then break it into 3 pieces
- 123+456+789 = 1368. To reduce the size to 3, either 1 or 8 is removed and accordingly the key would be 136 or 368 respectively.
- If the table size is 10, we get the index as 368%10 = 8

The Mid-square Method

Idea: square the key value, take the middle r digits

- If the key is 3121,
- 3121^2 = 9,7**40,6**41,
- and if the table size is 10,
- h(3121) = 406, which is the middle part of 3121^2
  - So, we get the index, 406%10 = 6

Collisions and their Resolution

A collision occurs when more than one keys hash to the same value.
E.g. For TableSize = 17, the keys 18 and 35 hash to the same value
18 mod 17 = 1 and 35 mod 17 = 1

Collision resolution may be done by searching for the next free slot at or after the position given by the hash function.

Two different methods for collision resolution:

- **Separate Chaining/Open hashing:** Use a dictionary data structure (such as a linked list) to store multiple items that hash to the same slot
- **Open addressing/Closed Hashing:** Open addressing**:** search for empty slots using a second function and store item in first empty slot that is found

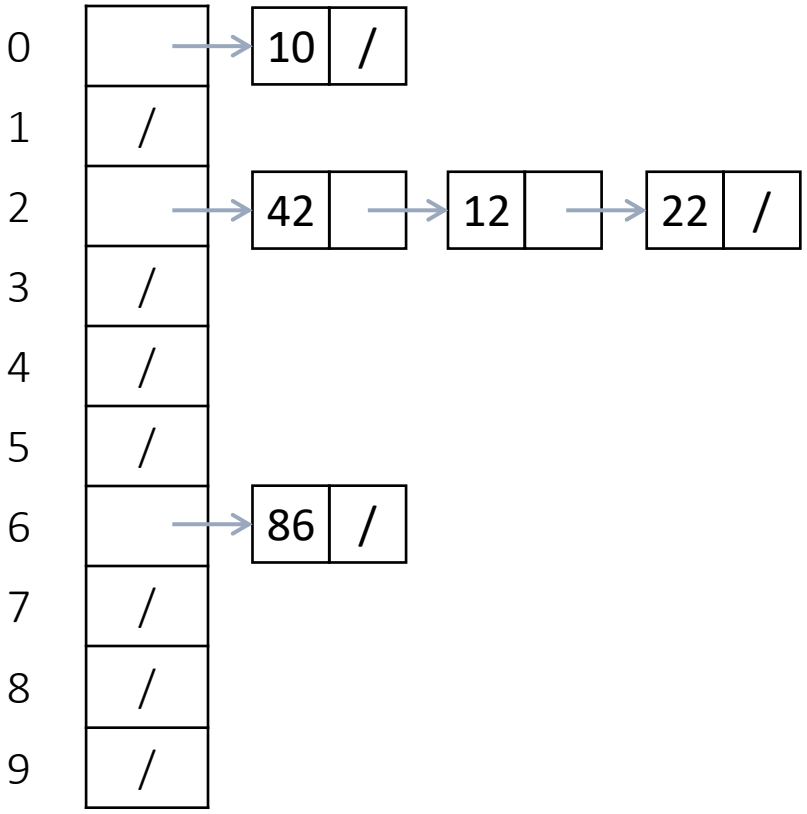
Collision Resolution by Separate Chaining

Each hash table entry requires a pointer to a linked list of records that can hold the element data.

This list is only used when a collision occurs.

All keys that map to the same table location are kept in a singly linked list.

Separate Chaining



Example: Insert 10, 42, 86, 12, 22 with h(x) = x % 10

Insertion, searching is O(1) in the average case