Lecture 15

Hashing

Heap Sort

- More swaps in general.
- Time complexity: O($n\log_2 n$)

Hashify

- Super fast data retrieval
- Hash table = Array (usually)
  - Unsorted = O(n) : Slow
  - Sorted = O($\log_2 n$) : Fast
  - Hash Table = O(1) : Superfast
- Retrieval using arrays is O(1)
- Arrays are Fixed/Rigid/Static data structure

Uses of Hashing

- World Wide Web [Google Search]
- Train/Flight booking systems, stock market websites [IP address]
- Search a word in a pdf file.
- Playing a live real-time multi-player game, possibly with high-end graphics
- Cryptographic password storage.
- Fingerprints
- Database indexing
- Caching systems

Methods of Hashing

- EVE = 5 + 22 + 5 = 32 = 3 + 2 = 5
- FEZ = 6 + 5 + 26 = 37 = 10 = 1

Create a function $f(x)$: Takes in a string and returns a number.

This function is called a *Hash function.*

Collisions

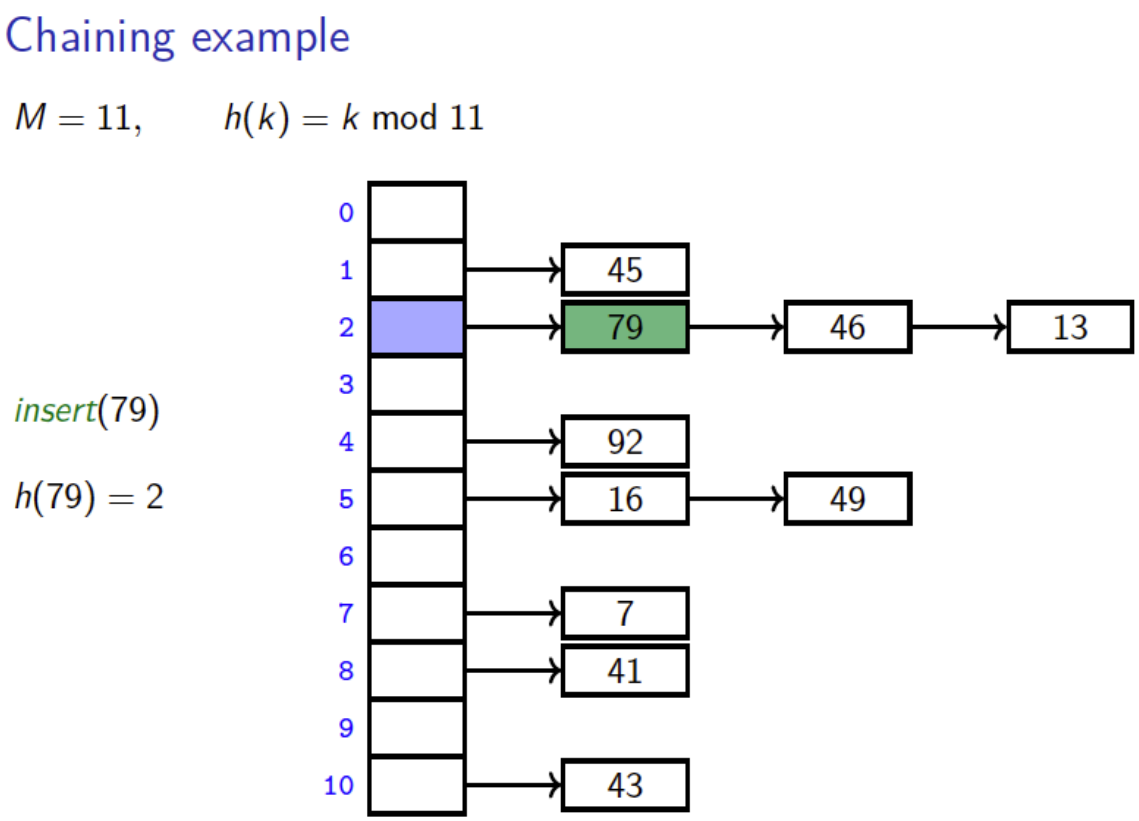
Situations where $X_1 \neq X_2$ but hashes are equivalent.

Load Factor

$$(\alpha) \text{ Load Factor} = \frac{\text{Number of Items Stored}}{\text{Size of Array}}$$

Open addressing ($\alpha < 1$)	Closed addressing ($\alpha > 1$)
Items can go to any location	Items can go only to their prescribed location
Elements are stored within the array itself	Elements are stored within external data structures
Collision handling is done by linear probing, quadratic probing	Collision handling is done via linked lists and chaining
Quicker retrieval	Better at handling deletions (O(1))

Closed Addressing (Chaining)

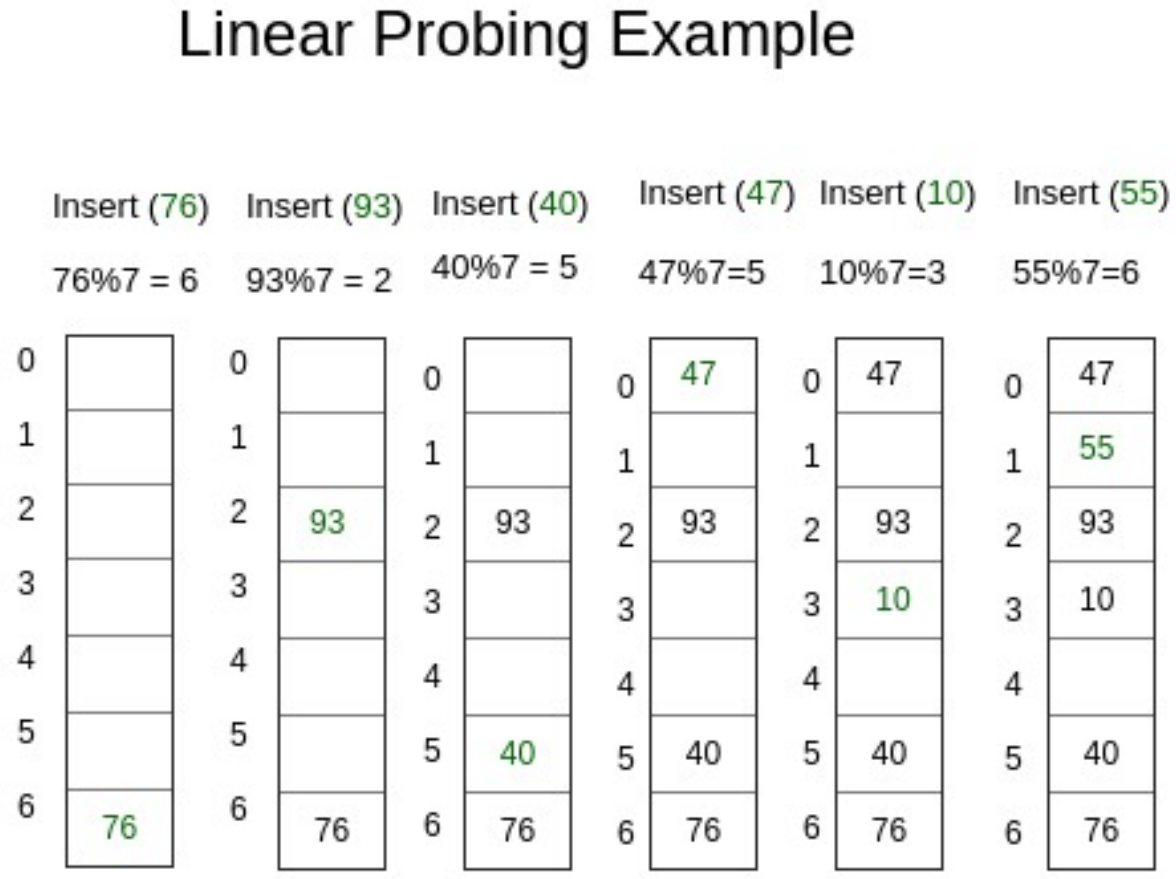


In the worst-case scenario we have α elements in each chain. If α gets too big we remake the hashmap and choose a new hashing function

Open Addressing

Linear Probing

If collision occurs we move on to the next empty element (wraparound)



Primary Clustering

> Items end up bunching together in linear probing

These clusters also tend to compound quite quickly.

This is bad because you now have increased search times for lookups and inserts, degrading complexity from O(1) to O(n).

Quadratic Probing

If a collision occurs, the next position is looked for as follows:

$h(k, i) = ai^2 + bi + h(k)$ where $h(k)$ is the auxiliary hash function.

Forms of Open Addressing

- Linear Probing
- Plus 3 Rehash
- Quadratic Probing
- Double Hashing: In case of collision a second hash function is applied to the first

For example, $H(k, i) = h_1(k) + ih_2(k)$

- If $\alpha = 1$, it is theoretically possible to design a perfect hash function (0 collisions)

Important

What is a good hash function?

- Minimizes collisions
- Uniformly distributes the hash values

- Easy to compute
- Resolve collisions efficiently

Universal Hashing

- Universal family of hashing functions
- Choose one hash function uniformly at random
- Family: $H,$ choose $h \epsilon H$

$$h_{a,b}(x) = [(ax + b) \mod p] \mod |A|$$

Note

When $x_1 \neq x_2$, then the number of different hash function $h_{a,b}(x_1) = h_{a,b}(x_2)$ is at most $\frac{|H|}{|A|} \Longrightarrow$ If A is large, the probability of a collision is very small