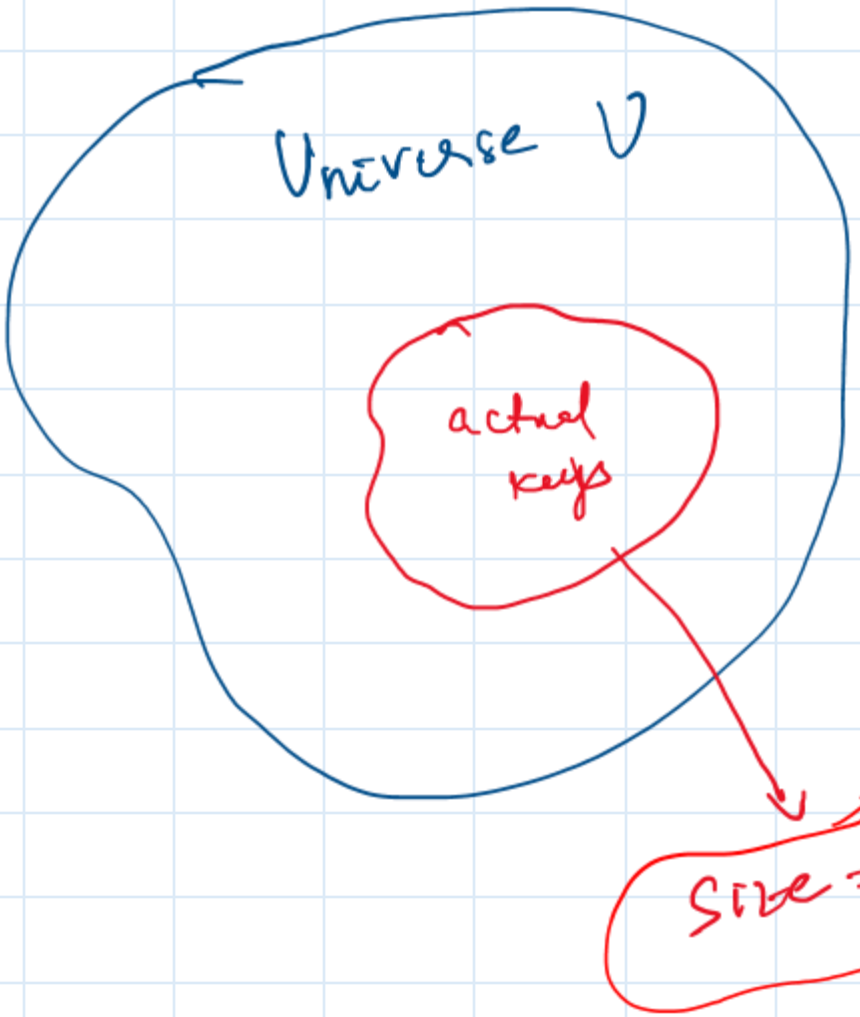
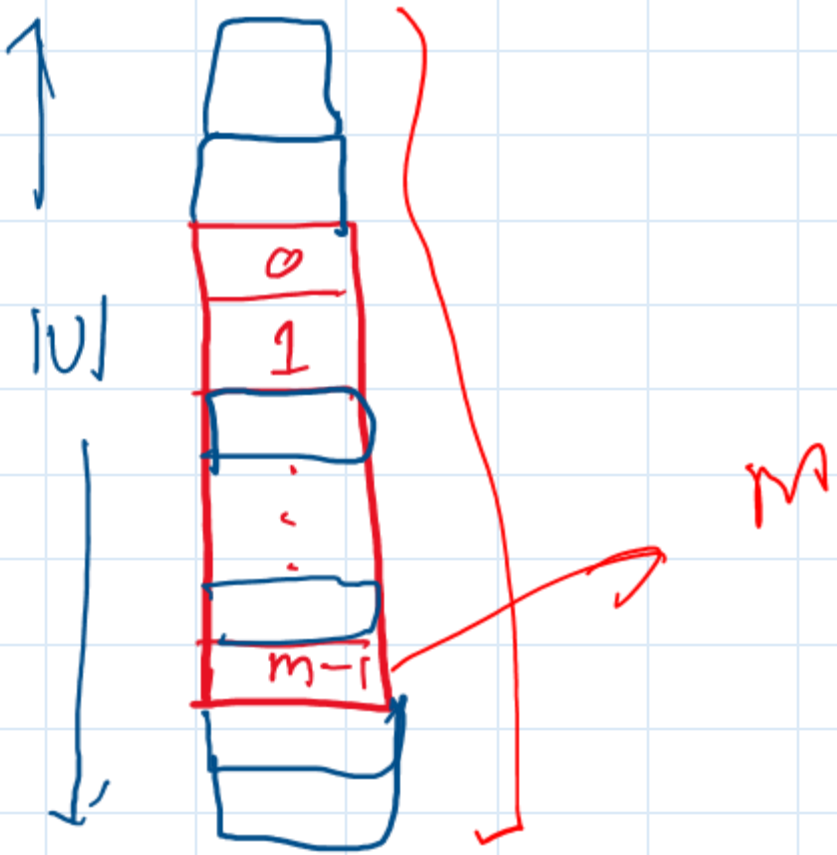
COL106 - Data Structures and Algorithms

Hashtables (contd), other related ADTs

Hash Tables





Hash functions = Hash code + Compression map

key → integer

integer → [0, m-1]

Hash Code:

- Integer cast (interpret bits of key as integer)
- Component sum (partition bits of key into components & add up)
- Memory address (interpret memory address as integer)
- Polynomial accumulation (partition key to components $a_0, \dots a_{n-1}$ & evaluate polynomial)

$x \rightarrow a_0, a_1, \dots$ of fixed length

a_{n-1}

$$p(z) = a_0 + a_1 z + \dots + a_{n-1} z^{n-1}$$

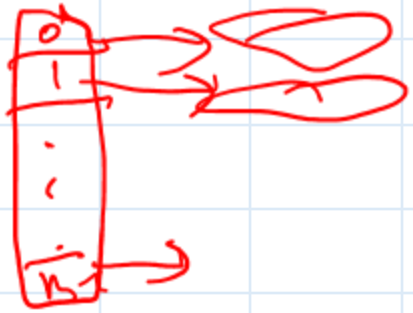
Horner's rule: $p(z) = a_0 + z(a_1 + z(a_2 + \dots a_n))$

$p_0(z) \leftarrow a_{n-1}$, $p_i(z) \leftarrow a_{n-i-1} + z\, p_{i-1}(z)$

$z = 33$ gives ≤ 6 collisions on a set of 50,000 English words

Compression Maps

(1) $h(k) = k \pmod m$ (simple division)



choosing $m = b^e$ (bad!)

Typically m is chosen to be prime - gives a uniform distribution.

not close to a power of 2.

Compression Maps

① $h(k) = k \pmod m$ (simple division)

choosing $m = b^e$ (bad!)

Typically m is chosen to be prime - gives a uniform distribution.

(not close to a power of 2.)

② $h(k) = \lfloor m (kA \bmod 1) \rfloor$ where $0 < A < 1$

Eg: $A = \frac{\sqrt{5} - 1}{2}$ (Fibonacci hashing)

(Knuth TAOCP vol.2)

Compression maps

3. "MAD" (Multiply, Add & Divide)

$h(k) = |ak+b| \bmod m$

a not a multiple of m

(often used in pseudo random number generators)

Universal Hashing:

$H = \{ h : U \to \{0, \ldots, m-1\} \}$

H is universal if for any randomly chosen h from H, & keys k_x, k_y

$$Pr[h(k_x) = h(k_y)] \leq 1/m$$

There are efficient constructions & UHF

Handling Collisions

$h: U \to \{0, 1, \ldots, m\}$

Open addressing:

In chaining all elements were stored outside the hash table.

- Put all the elements in the table $\Rightarrow n \leq m$
- Have a systematic way to probe elements of the table.
- Modify hash function: $h: U \times \{0, 1, \ldots, m-1\} \to \{0, 1, \ldots, m-1\}$
  - U: universe
  - $\{0, 1, \ldots, m-1\}$: specify the probe number
- Probe sequence: $\langle h(k,0), h(k,1), \ldots, h(k,m-1) \rangle$
- h gives a sequence of slots examined for a given key.

Linear Probing

Uses less memory than chaining, but slower

Example:

$h(x) = x \bmod 13$

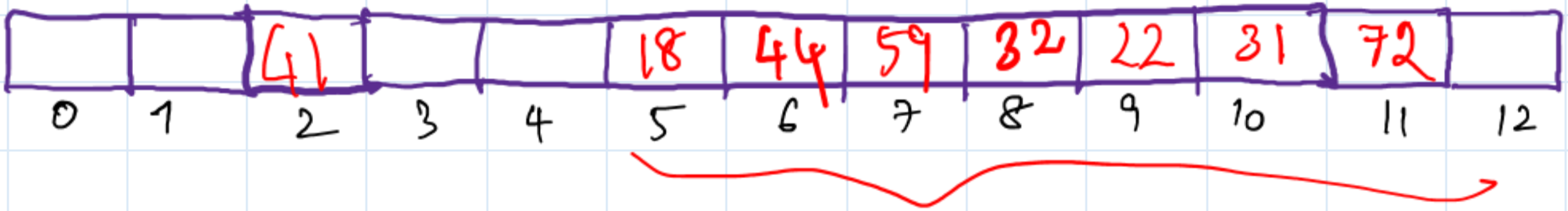
causes clustering

n →

n ≥ 12

n = 13

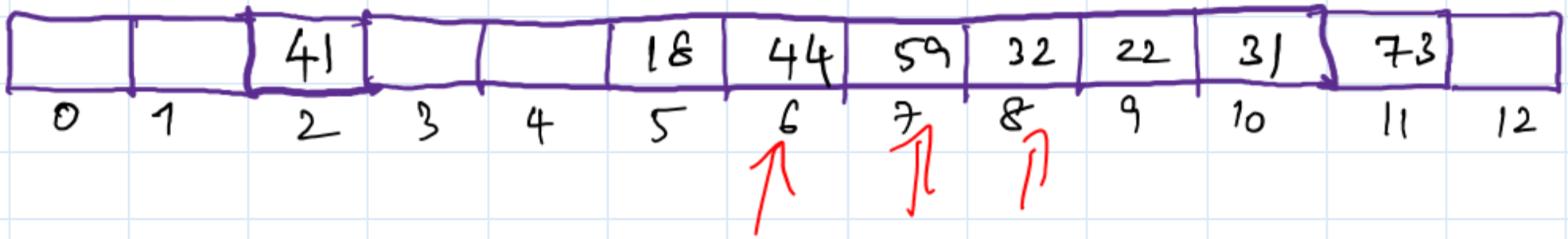
Insert: 18, 41, 22, 44, 59, 32, 31, 73



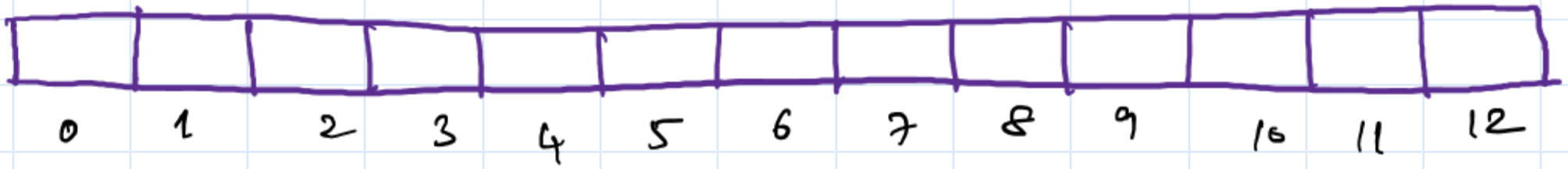
Linear Probing

Example: $h(x) = x \bmod 13$

Insert: 18, 41, 22, 44, 59, 32, 31, 73



Delete 32



Linear Probing

$ax+b \bmod 10$

Example: $h(x) = x \bmod 13$

$n \leq m$

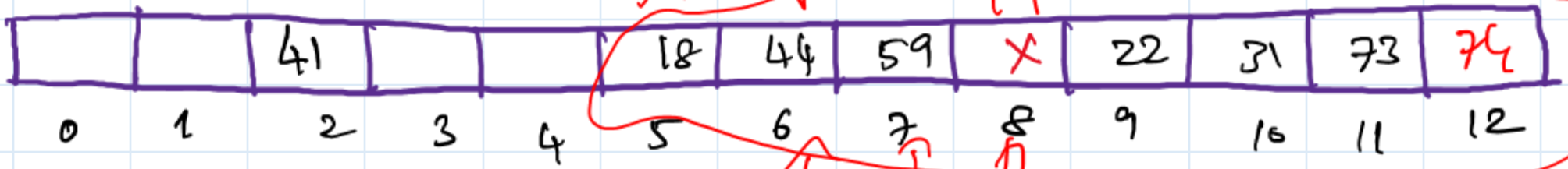
$h(x) = 12$

Insert: 18, 41, 22, 44, 59, 32, 31, 73

Delete 32

lookup -> ignore X

insert -> replace X



- Rehash if there are too many X's

Double Hashing

Quadratic probing:
$(h(k)+i^2) \bmod m$

Use two functions: primary hash $h(k)$
secondary hash $d(k)$

- table size (m) must be prime.

secondary hash → cannot take 0 values.

$d(k)$ → handles collision by placing item in first available cell in $(i+j\,d(k)) \bmod m$
$j \in \{0, 1, \ldots, m-1\}$

```
DoubleHashInsert(k)
    if (table is full) error
    probe = h(k);   offset = d(k)
    while (table[probe] occupied)
        probe = (probe + offset) mod m;

    table[probe] = k
```

Double Hashing Example

$h: V \times \{0, 1, \ldots, m-1\} \rightarrow \{0, 1, \ldots, m-1\}$

$m = 13$

$h(k) = k \bmod 13$

$d(k) = (7 - k \bmod 7)$

$h(k)$

5

Insert: 18, 41, 22, 44, 59, 32, 31, 73

offset : $d(44) = 5$

Analysis of Double Hashing

- Let α (load factor) be < 1
- Assume every probe looks at a random location in the table
- $1 - \alpha$ fraction of table is empty.
- Expected # probes to find an empty location $= \frac{1}{1-\alpha}$

required for unsuccessful search.

$\alpha = 1/10$

$\alpha \to 1/10$

Analysis of Double Hashing

- Average number of probes for a succesful search = Average number of probes required to insert all elements

- To insert an element we need to find an empty location.

	# probes	Total # probes
$\{1, \dots m/2\}$	≤ 2	m
$\{\frac{m}{2}+1, \dots \frac{m}{2}+\frac{m}{4}\}$	≤ 4	m
$\{\frac{m}{2}+\frac{m}{4}+1, \dots \frac{m}{2}+\frac{m}{4}+\frac{m}{8}\}$	≤ 8	m
next $m/2^i$	$\leq 2^i$	m