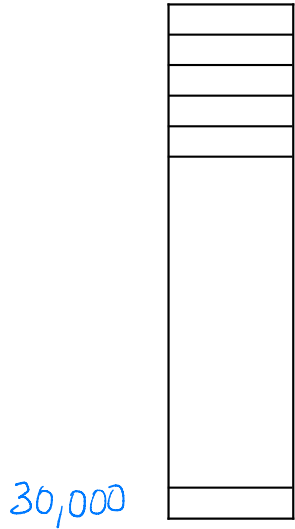


## Hash Code

$g("c_{k-1} c_{k-2} \dots c_2 c_1 c_0") \rightarrow \text{integer}$

$g("c_{k-1} c_{k-2} \dots c_2 c_1 c_0") =$



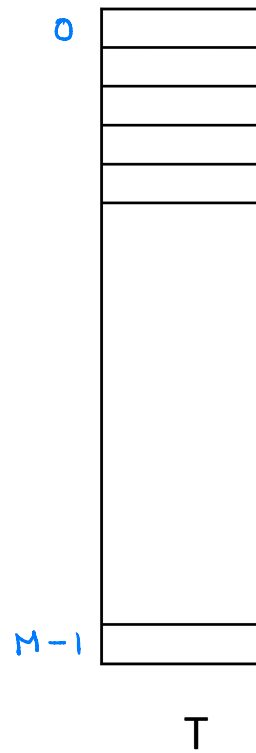
T

## Polynomial Hash Code

Polynomial:  $p(x) = a_{k-1}x^{k-1} + a_{k-2}x^{k-2} + \dots + a_1x^1 + a_0$

$g("c_{k-1} c_{k-2} \dots c_2 c_1 c_0") =$

# Compression Map



$$f(\text{integer}) \rightarrow \{0, 1, \dots, M-1\}$$

## Hash Function with Polynomial Hash Code

$$h("c_{k-1} c_{k-2} \dots c_0") = (\dots (((((\text{int})c_{k-1})x + (\text{int})c_{k-2})x + (\text{int})c_{k-3})x + \dots + (\text{int})c_1)x + (\text{int})c_0) \bmod M$$

## Hash Function with Polynomial Hash Code

$$h("c_{k-1} c_{k-2} \dots c_0") = (\dots (((((\text{int})c_{k-1})x + (\text{int})c_{k-2})x + (\text{int})c_{k-3})x + \dots + (\text{int})c_1)x + (\text{int})c_0) \bmod M$$

**Algorithm** polynomialHashFunction("c<sub>k-1</sub>,c<sub>k-2</sub>, ... c<sub>0</sub>",x,M)

**Input:** String "c<sub>k-1</sub>,c<sub>k-2</sub>, ... c<sub>0</sub>", value x, size M (**M is a prime number**) of the hash table

**Output:** value of the hash function for input string

# Collision Resolution: Separate Chaining

|   |  |
|---|--|
| 0 |  |
| 1 |  |
| 2 |  |
| 3 |  |
| 4 |  |
| 5 |  |
| 6 |  |

T

$$h(k) = k \bmod 7$$

Records to store  
in the table

(14,  $d_1$ )

(12,  $d_2$ )

(13,  $d_3$ )

(21,  $d_4$ )

**Algorithm** get(k)

**Input:** Key k

**Output:** Record with key k, or  
null if no record has key k

# Collision Resolution: Open Addressing

|   |  |
|---|--|
| 0 |  |
| 1 |  |
| 2 |  |
| 3 |  |
| 4 |  |
| 5 |  |
| 6 |  |

T

$$h(k) = k \bmod 7$$

Records to store  
in the table

(14,  $d_1$ )

(12,  $d_2$ )

(13,  $d_3$ )

(21,  $d_4$ )

(19,  $d_5$ )

(2,  $d_6$ )

(5,  $d_7$ )

remove (14)

**Algorithm** get(k)

**Input:** Key k

**Output:** Record with key k, or  
null if no record has key k

Initially every entry of T is null

Linear probing:

$h(k), (h(k)+1) \bmod M, (h(k) + 2) \bmod M, ((h(k) + 3) \bmod M \dots$

# Computer Memory

|        |        |        |        |        |        |        |       |        |        |        |        |        |        |        |        |       |        |       |
|--------|--------|--------|--------|--------|--------|--------|-------|--------|--------|--------|--------|--------|--------|--------|--------|-------|--------|-------|
| 11010  | 11001  | 01100  | 10010  | 00011  | 10010  | 011010 | 11001 | 101100 | 010010 | 00011  | 110010 | 101100 | 010010 | 00011  | 110010 | 01110 | 11001  | 1101  |
| 11011  | 10001  | 00111  | 01101  | 10011  | 00101  | 101011 | 10001 | 100111 | 01101  | 10011  | 00101  | 100111 | 01101  | 10011  | 00101  | 11011 | 000111 | 11011 |
| 11101  | 11111  | 10000  | 001101 | 011101 | 001101 | 11101  | 11111 | 00000  | 001101 | 011101 | 10011  | 00000  | 001101 | 011101 | 10011  | 11001 | 10011  | 0011  |
| 101101 | 110010 | 101110 | 011111 | 11011  | 10001  | 101101 | 11001 | 11011  | 001111 | 11011  | 10001  | 11011  | 001111 | 11011  | 10001  | 10101 | 10101  | 01110 |
| 011010 | 11001  | 101100 | 010010 | 101101 | 011100 | 011010 | 11001 | 101100 | 010010 | 00011  | 110010 | 101100 | 010010 | 00011  | 110010 | 00011 | 010011 | 01110 |
| 101011 | 10001  | 100111 | 01101  | 10011  | 00101  | 101011 | 10001 | 100111 | 01101  | 10011  | 00101  | 100111 | 01101  | 10011  | 00101  | 11011 | 10011  | 10101 |
| 11101  | 11111  | 00000  | 001101 | 011101 | 10011  | 11101  | 11111 | 00000  | 001101 | 011101 | 10011  | 00000  | 001101 | 011101 | 10011  | 10110 | 011100 | 0111  |
| 101101 | 11001  | 11011  | 001111 | 11011  | 10001  | 101101 | 11001 | 11011  | 001111 | 11011  | 10001  | 11011  | 001111 | 11011  | 10001  | 10011 | 10111  | 0011  |
| 011010 | 11001  | 101100 | 010010 | 00011  | 110010 | 011010 | 11001 | 101100 | 010010 | 00011  | 110010 | 101100 | 010010 | 00011  | 110010 | 11100 | 001111 | 11101 |
| 101011 | 10001  | 100111 | 01101  | 10011  | 00101  | 101011 | 10001 | 100111 | 01101  | 10011  | 00101  | 100111 | 01101  | 10011  | 00101  | 10011 | 011101 | 10011 |
| 11101  | 11111  | 00000  | 001101 | 011101 | 10011  | 11101  | 11111 | 00000  | 001101 | 011101 | 10011  | 00000  | 001101 | 011101 | 10011  | 01110 | 11111  | 00111 |
| 101101 | 11001  | 11011  | 001111 | 11011  | 10001  | 101101 | 11001 | 11011  | 001111 | 11011  | 10001  | 11011  | 001111 | 11011  | 10001  | 11100 | 01010  | 01010 |
| 011010 | 11001  | 101100 | 010010 | 00011  | 110010 | 011010 | 11001 | 101100 | 010010 | 00011  | 110010 | 101100 | 010010 | 00011  | 110010 | 10111 | 0011   | 0111  |
| 101011 | 10001  | 100111 | 01101  | 10011  | 00101  | 101011 | 10001 | 100111 | 01101  | 10011  | 00101  | 100111 | 01101  | 10011  | 00101  | 10000 | 01110  | 10101 |
| 11101  | 11111  | 00000  | 001101 | 011101 | 10011  | 11101  | 11111 | 00000  | 001101 | 011101 | 10011  | 00000  | 001101 | 011101 | 10011  | 11100 | 0011   | 00110 |
| 101101 | 11001  | 11011  | 001111 | 11011  | 10001  | 101101 | 11001 | 11011  | 001111 | 11011  | 10001  | 11011  | 001111 | 11011  | 10001  | 10000 | 01101  | 10101 |
| 11101  | 11111  | 00000  | 001101 | 011101 | 10011  | 11101  | 11111 | 00000  | 001101 | 011101 | 10011  | 00000  | 001101 | 011101 | 10011  | 11101 | 11100  | 10101 |
| 101101 | 11001  | 11011  | 001111 | 11011  | 10001  | 101101 | 11001 | 11011  | 001111 | 11011  | 10001  | 11011  | 001111 | 11011  | 10001  | 10000 | 11111  | 0011  |



# Linear Probing and Double Hashing

|    |  |
|----|--|
| 0  |  |
| 1  |  |
| 2  |  |
| 3  |  |
| 4  |  |
| 5  |  |
| 6  |  |
| 7  |  |
| 8  |  |
| 9  |  |
| 10 |  |

$$h(k) = k \bmod 11$$

Records to store  
in the table

$(3, d_1)$   
 $(14, d_2)$   
 $(25, d_3)$   
 $(5, d_4)$   
 $(28, d_5)$   
 $(91, d_6)$

|    |  |
|----|--|
| 0  |  |
| 1  |  |
| 2  |  |
| 3  |  |
| 4  |  |
| 5  |  |
| 6  |  |
| 7  |  |
| 8  |  |
| 9  |  |
| 10 |  |

Secondary hash function:  
 $h'(k) = q - (k \bmod q)$   
for some prime value  $q$

$$h'(k) = 7 - (k \bmod 7)$$

Linear probing:

$h(k)$ ,  $(h(k) + 1) \bmod M$ ,  $(h(k) + 2) \bmod M$ ,  
 $((h(k) + 3) \bmod M) \dots$

Double hashing:

$h(k)$ ,  $(h(k) + h'(k)) \bmod M$ ,  $(h(k) + 2h'(k)) \bmod M$ ,  
 $((h(k) + 3h'(k)) \bmod M) \dots$

# Double Hashing and Size of the Table

|   |  |
|---|--|
| 0 |  |
| 1 |  |
| 2 |  |
| 3 |  |
| 4 |  |
| 5 |  |
| 6 |  |
| 7 |  |

$$h(k) = k \bmod 8$$

Records to store  
in the table

$(2, d_1)$

$(6, d_2)$

$(10, d_3)$

Secondary hash function:

$$h'(k) = q - (k \bmod q)$$

for some prime value  $q$

$$h'(k) = 7 - (k \bmod 7)$$

Double hashing:

$$h(k), (h(k) + h'(k)) \bmod M, (h(k) + 2h'(k)) \bmod M, ((h(k) + 3h'(k)) \bmod M \dots$$

# Open Addressing: put Method (linear probing)

**Algorithm** put (k,data, M)

**In:** record (k,data) to insert, size M of hash table

**Out:** {add record (k,data) to table, or ERROR if insertion not allowed}

pos  $\leftarrow$  h(k)

count  $\leftarrow$  0

**while** (T[pos]  $\neq$  NULL) **and** (T[pos]  $\neq$  DELETED) **do** {

**if** T[pos].getKey() = k **then** *ERROR*

    pos  $\leftarrow$  (pos + 1) **mod** M

    count  $\leftarrow$  count + 1

**if** count = M **then** *ERROR*

}

T[pos]  $\leftarrow$  (k,data)

# Open Addressing: put Method (double hashing)

**Algorithm** put (k,data, M)

**In:** record (k,data) to insert, size N of hash table

**Out:** {add record (k,data) to table, or ERROR if insertion not allowed}

pos  $\leftarrow$  h(k)

count  $\leftarrow$  0

**while** (T[pos]  $\neq$  NULL) **and** (T[pos]  $\neq$  DELETED) **do** {

**if** T[pos].getKey() = k **then** *ERROR*

    pos  $\leftarrow$  (pos +  $h'(k)$ ) **mod** M

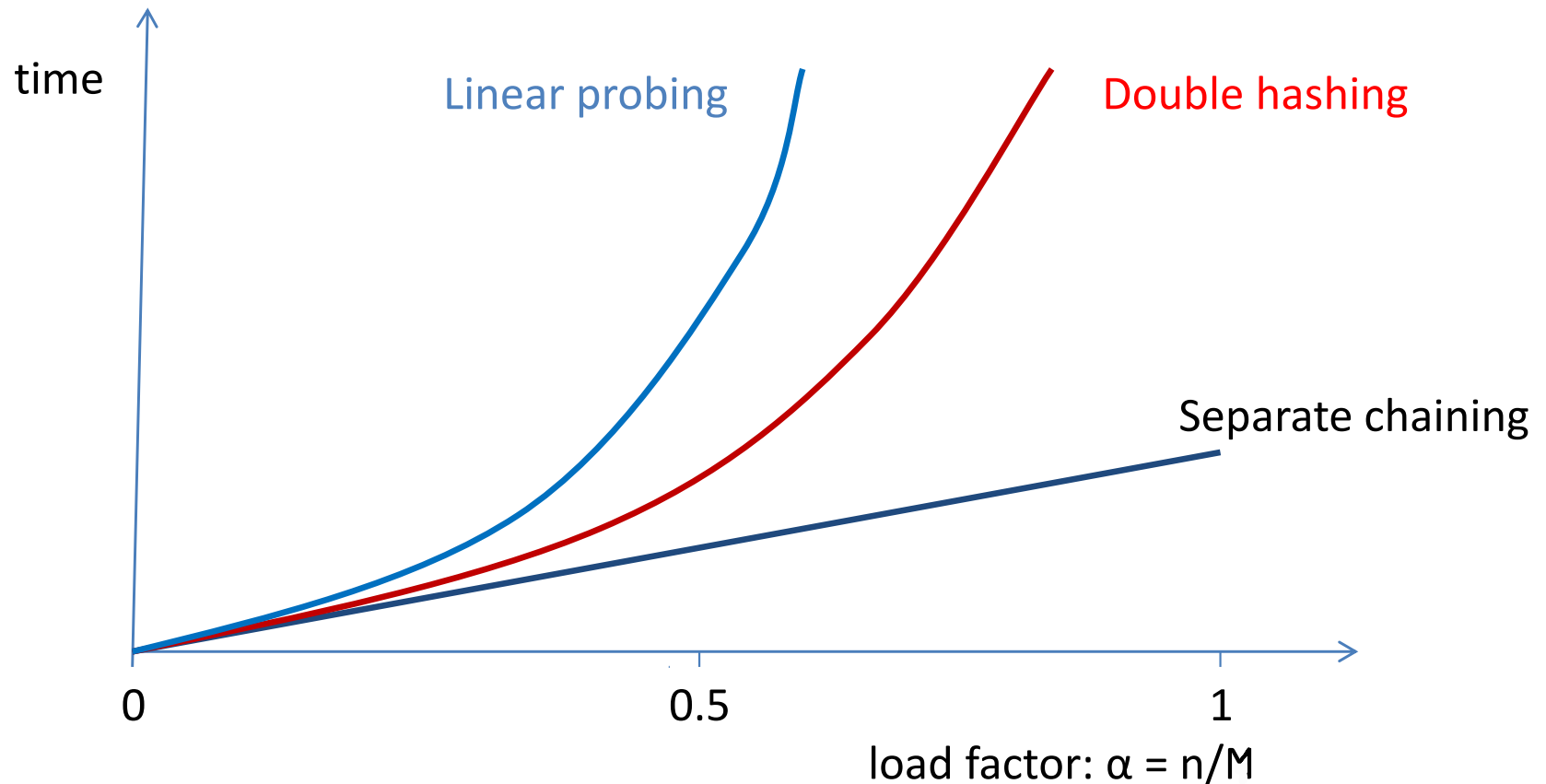
    count  $\leftarrow$  count + 1

**if** count = M **then** *ERROR*

}

T[pos]  $\leftarrow$  (k,data)

# Average Time Complexity of **get** Operation



Average number of key comparisons

Separate chaining

$$1 + \alpha$$

Linear Probing

$$\frac{1}{2} + \frac{1}{2(1 - \alpha)^2}$$

Double Hashing

$$\frac{1}{1 - \alpha}$$