TRƯỜNG ĐẠI HỌC MỞ TP. HCM

Cơ hội học tập cho mọi người

CHƯƠNG 4: BẢNG BĂM

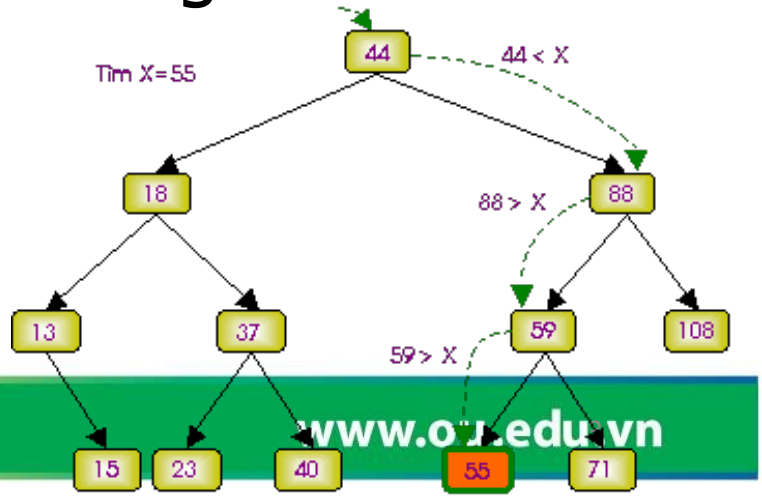
MỤC TIÊU

- Hàm băm
- Các phương pháp giải quyết đụng độ

Đặt vấn đề

- Phương pháp tìm kiếm trong các chương trước chủ yếu dựa vào khóa
  - Mảng
  - Danh sách liên kết
  - Cây nhị phân tìm kiếm

⇨ Tìm kiếm bằng cách **so sánh lần lượt các phần tử** ⇨Thời gian tìm kiếm không nhanh và **phụ thuộc N** (số phần tử)



Khái niệm bảng băm

- Bảng băm (hash table) là một cấu trúc dữ liệu, lưu trữ các khóa trong bảng T (danh sách đặc); sử dụng một hàm băm (hash function) để ánh xạ khoá (key) với một địa chỉ lưu trữ

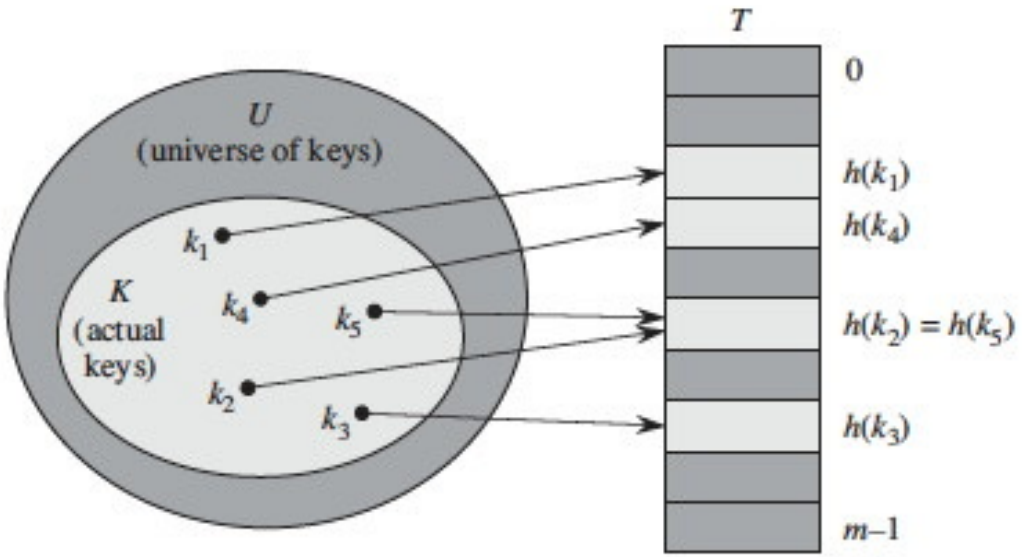
Hàm băm

- Hàm băm (hash function) là hàm biến đổi khóa k thành địa chỉ trong bảng băm
- Phép biến đổi khóa: là một ánh xạ khoá thành địa chỉ

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

- Trong đó
  - U là tập các khóa
  - {0, 1, …, m – 1} là tập các địa chỉ trên bảng băm
  - Giá trị h(k) gọi là hash code hoặc địa chỉ

Hàm băm


T
U
(universe of keys)
K
(actual
keys)
k_1
k_4
k_5
k_2
k_3
0
$h(k_1)$
$h(k_4)$
$h(k_2) = h(k_5)$
$h(k_3)$
$m-1$

Một vài loại hàm băm

- Chia modulo

$$h(k) = K \ mod \ Tsize$$

  - Với

$$Tsize = sizeof(table)$$

Tsize tốt nhất nên là số nguyên tố

- Sinh viên có thể tham khảo thêm các hàm băm
  - Folding (gấp số)
  - Mid-Square Function
  - …

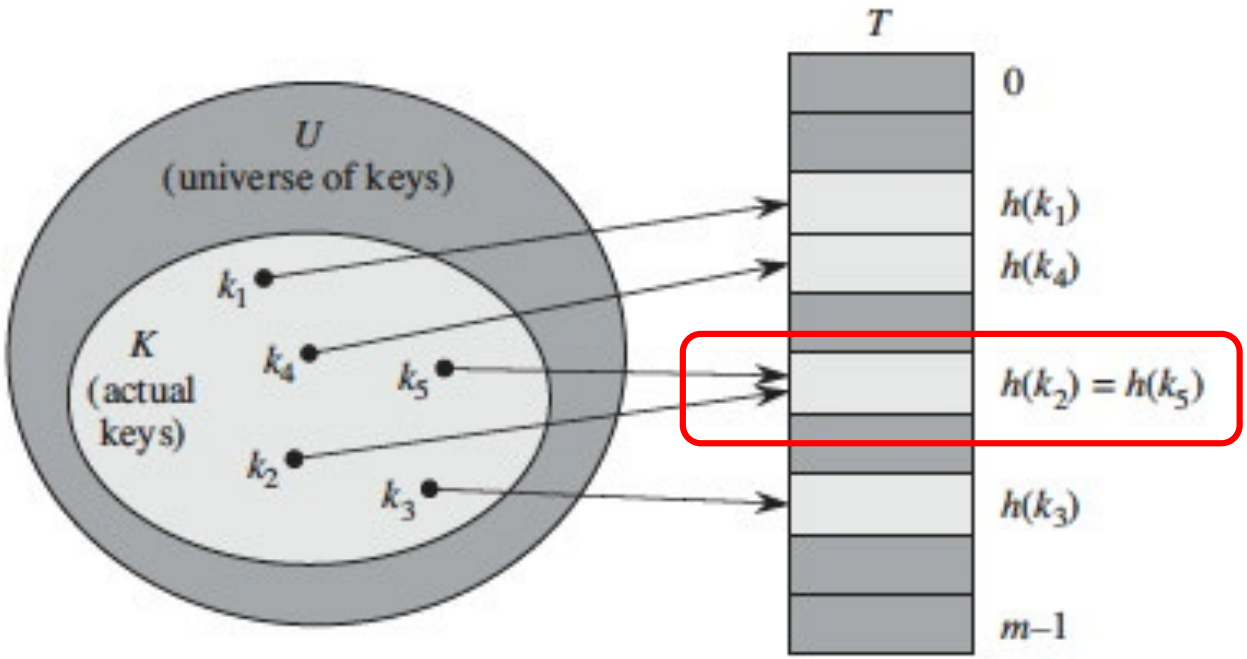
Sự đụng độ (Collision)

- Sự đụng độ hay xung đột địa chỉ
  - Một cách lý tưởng, hàm băm sẽ ánh xạ mỗi khoá vào một slot riêng biệt của bảng T
- Tuy nhiên, điều này trong thực tế khó đạt được, vì:
  - m << |U|
  - Các khoá là không biết trước

Sự đụng độ

❖Đụng độ là

$$\exists k_i, k_j \in K: k_i \neq k_j, h(k_i) = h(k_j)$$

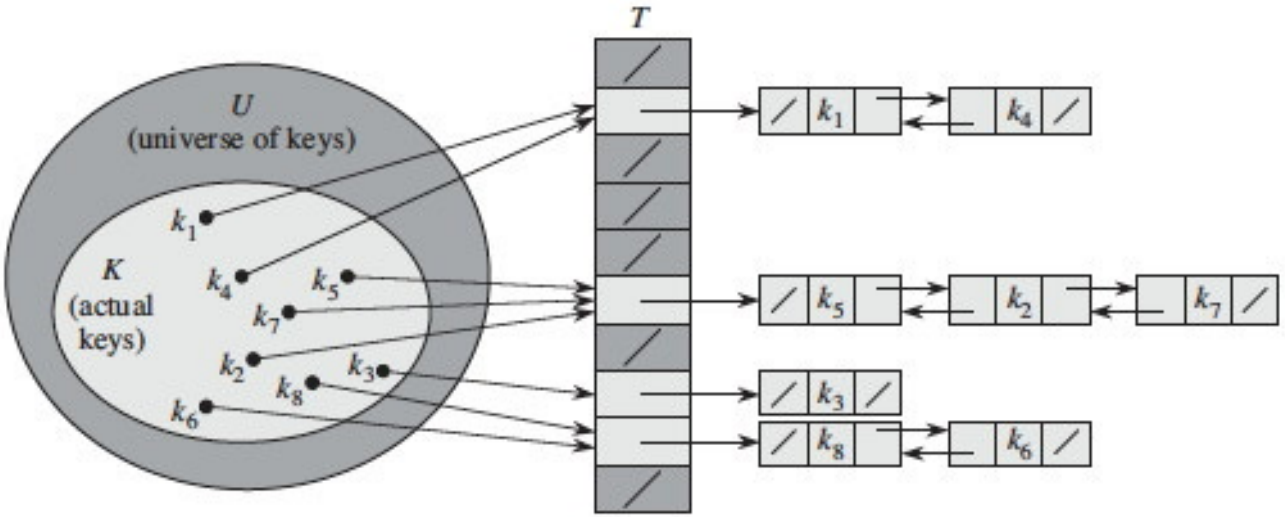


Các phương pháp xử lý đụng độ

- Phương pháp nối kết (separate chaining)
  - Phương pháp nối kết trực tiếp
  - Phương pháp nối kết hợp nhất
- Phương pháp địa chỉ mở (open addressing)
  - (SV tham khảo tài liệu) [2], [3]

Phương pháp nối kết

- Ứng với mỗi địa chỉ của bảng, ta có một danh sách liên kết chứa các phần tử có khóa khác nhau mà có cùng địa chỉ đó.
- Ta sẽ có danh sách (bảng băm) gồm M phần tử chứa địa chỉ đầu của các danh sách liên kết.



Phương pháp nối kết trực tiếp

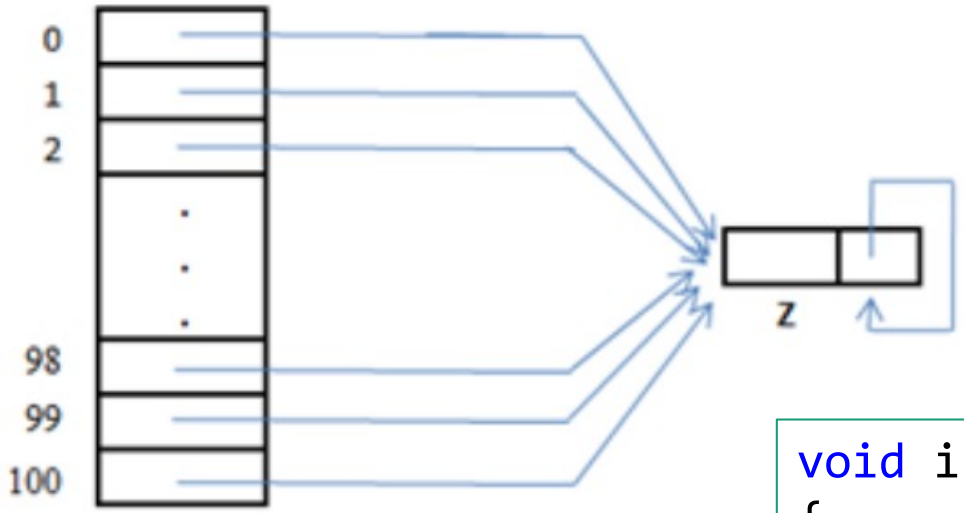
❖Khai báo

```
#define M 101

struct Node
{
    int key;
    Node* next;
};

Node* heads[M];
Node* z;
```

- Lưu ý chọn giá trị M phù hợp để quản lý một bảng băm có tổng số phần tử là n:
  - M phải là số nguyên tố
  - M tương đương (hoặc nhỏ hơn) giá trị n/10
  - Mảng danh sách đặc heads[M] khi khai báo có khả năng đủ vùng nhớ để cấp phát

Phương pháp nối kết trực tiếp

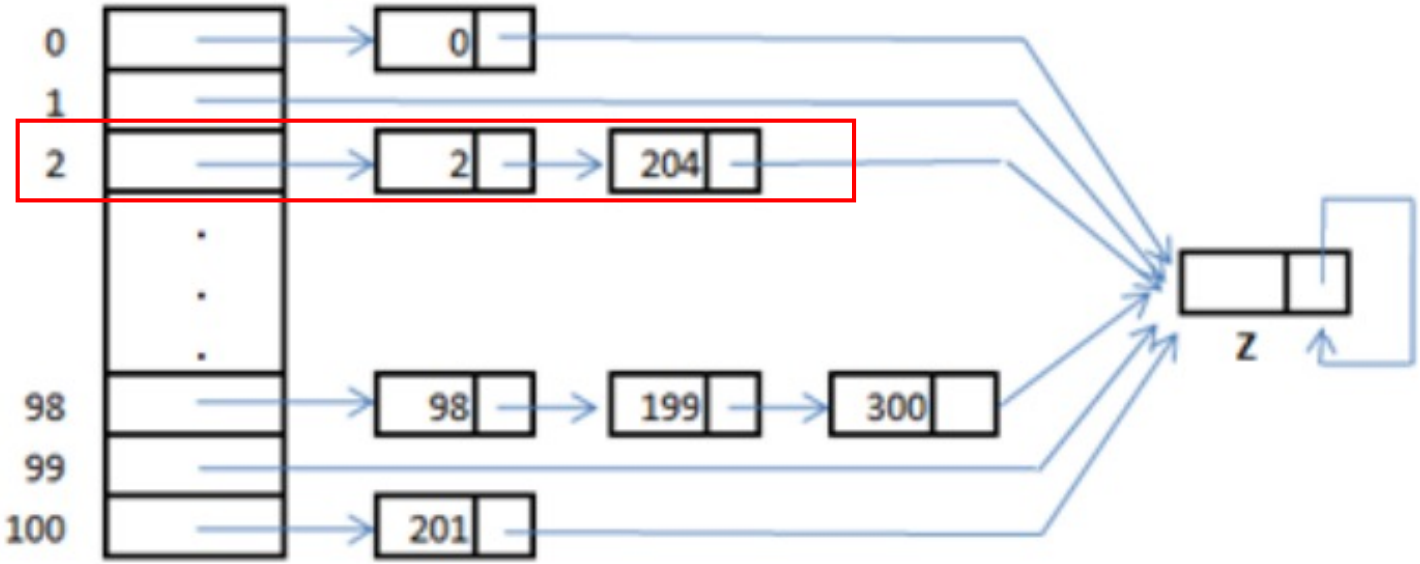
- Khởi tạo




```
void init()
{
    z = new Node;
    z->next = z;
    for (int i = 0; i < M; i++)
        heads[i] = z;
}
```

Phương pháp nối kết trực tiếp

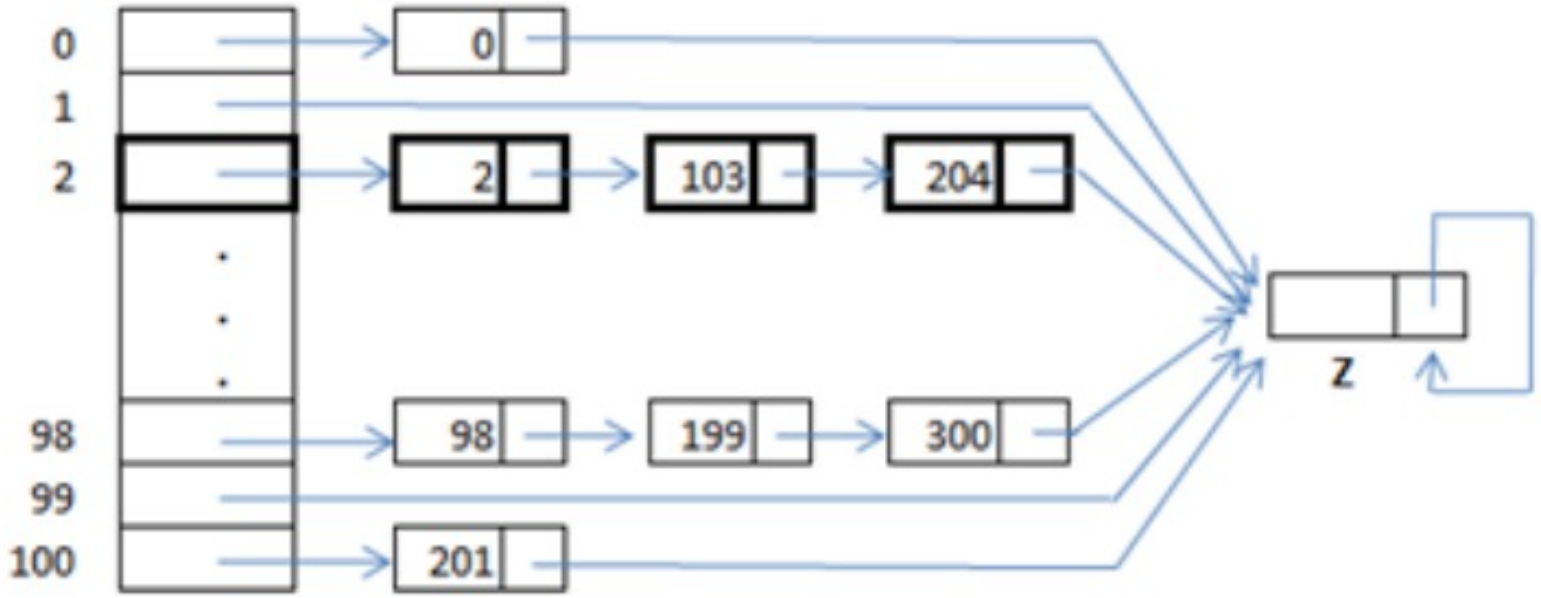
❖Thêm một phần tử vào bảng băm: k = 103



h(k) = 103 % 101 = 2

Phương pháp nối kết trực tiếp

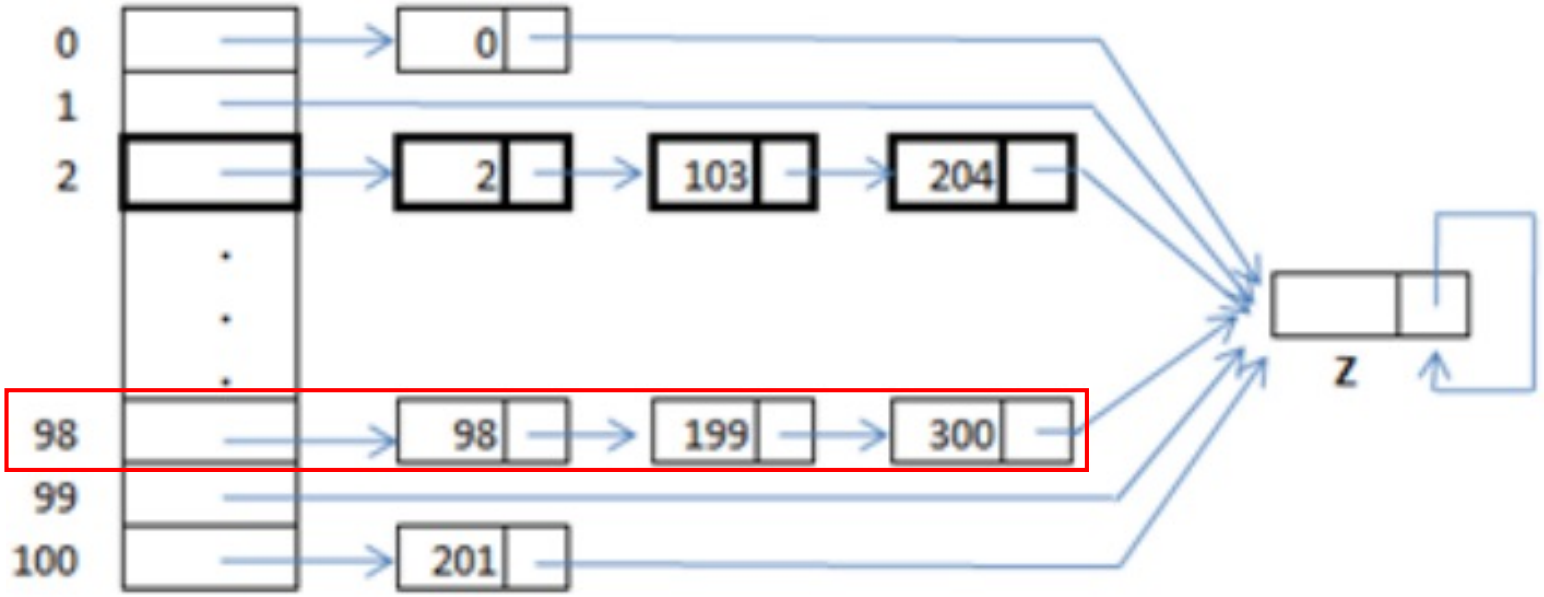
- Thêm một phần tử vào bảng băm: k = 103



h(k) = 103 % 101 = 2

Phương pháp nối kết trực tiếp

❖Thêm một phần tử vào bảng băm: k = 401

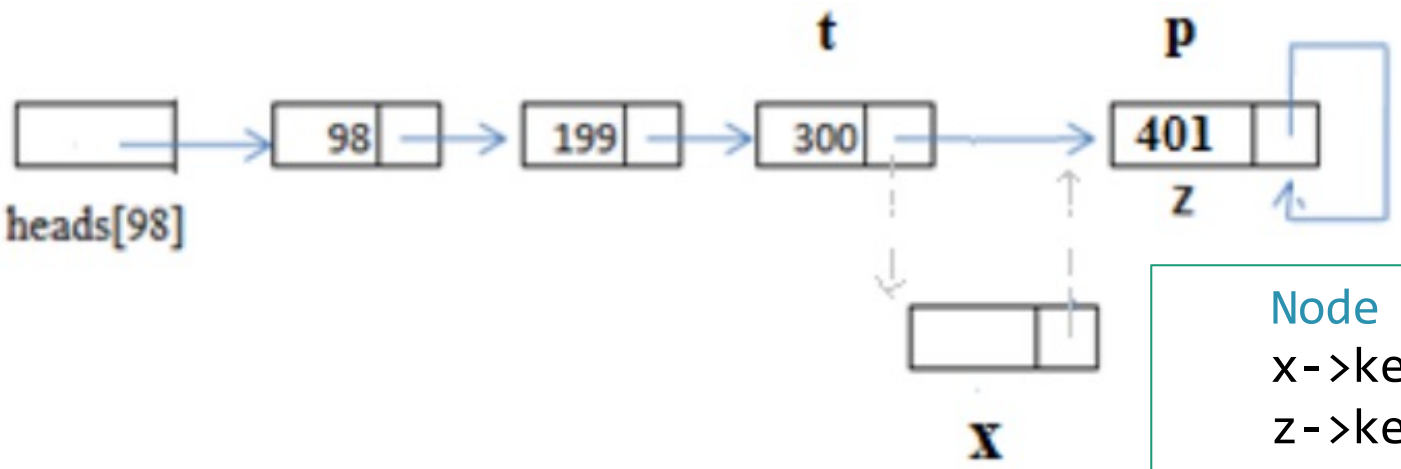


h(k) = 401 % 101 = 98

Phương pháp nối kết trực tiếp

- Thêm một phần tử vào bảng băm: k = 401

z -> key = k = 401




```
Node * x = new Node;
x->key = k;
z->key = k;

while (p->key < k)
{
    t = t->next;
    p = t->next;
}
t->next = x;
x->next = p;
```

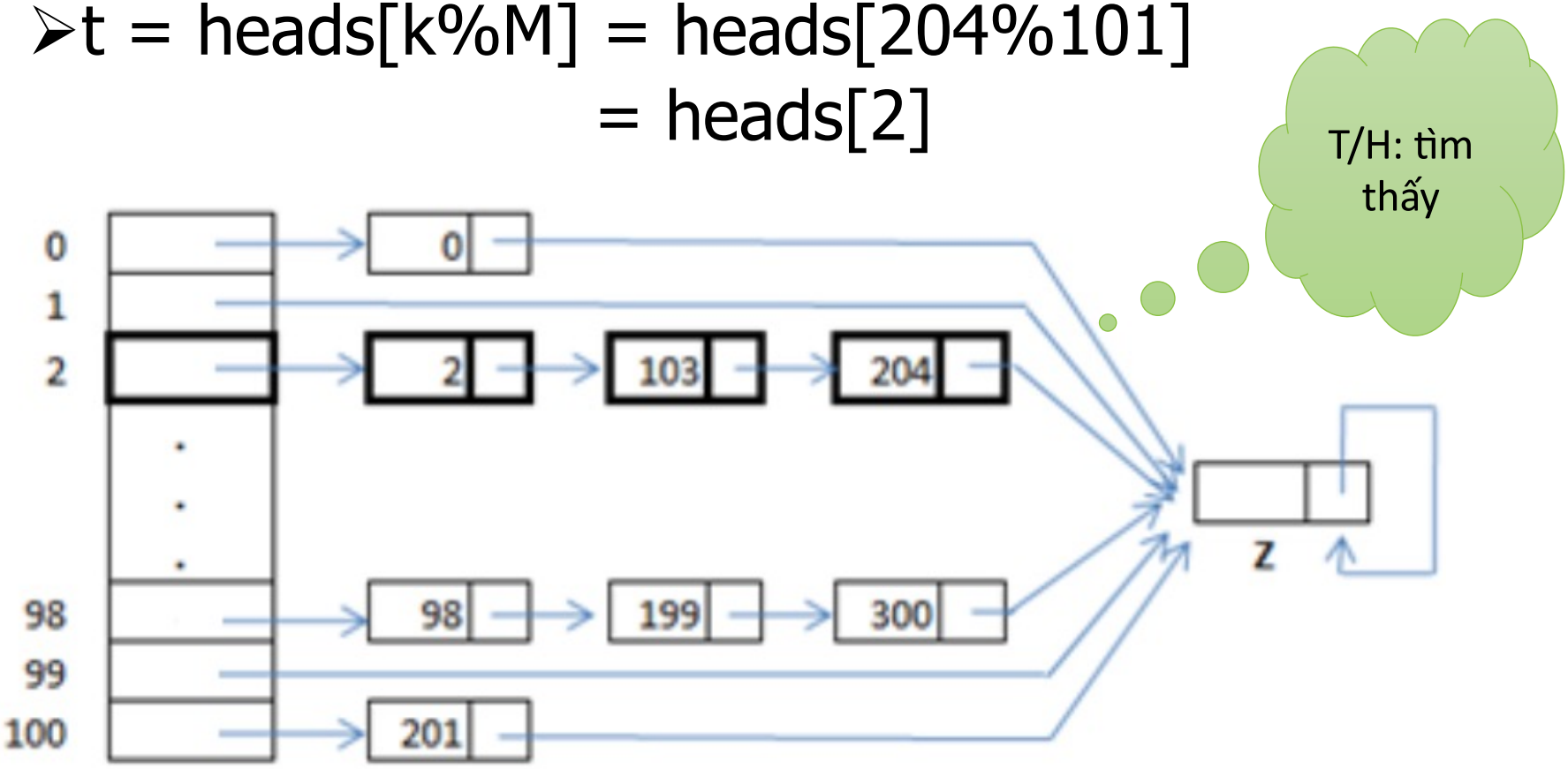
Phương pháp nối kết trực tiếp

- Thêm một phần tử vào bảng băm

```
Node* insert(int k)
{
    Node * t = heads[k % M];
    Node * x = new Node;
    x->key = k;
    z->key = k;

    if (t->key >= k)
    {
        x->next = t;
        heads[k % M] = x;
    }
    else
    {
        Node *p = t->next;
        while (p->key < k)
        {
            t = t->next;
            p = t->next;
        }
        t->next = x;
        x->next = p;
    }
    return x;
}
```

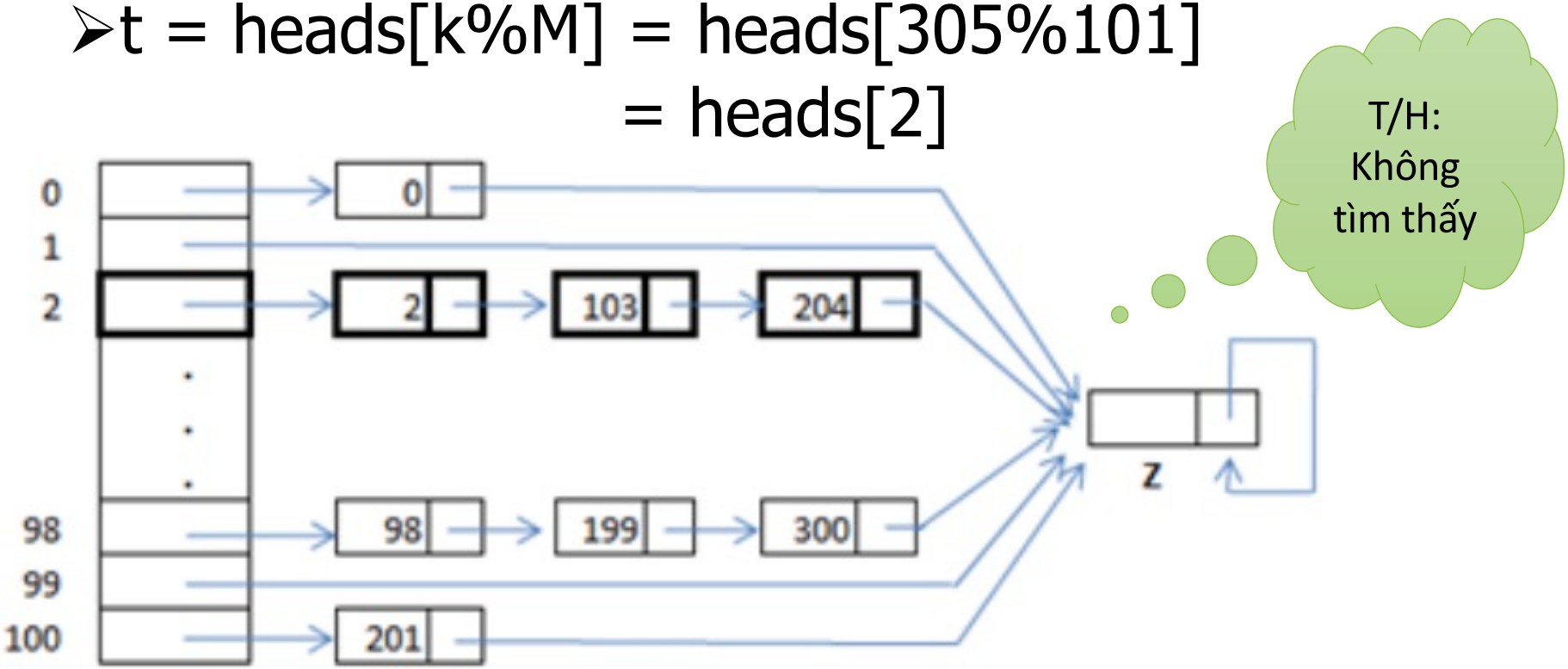
Phương pháp nối kết trực tiếp

- Tìm một phần tử trong bảng băm: k = 204
  - t = heads[k%M] = heads[204%101]
    = heads[2]



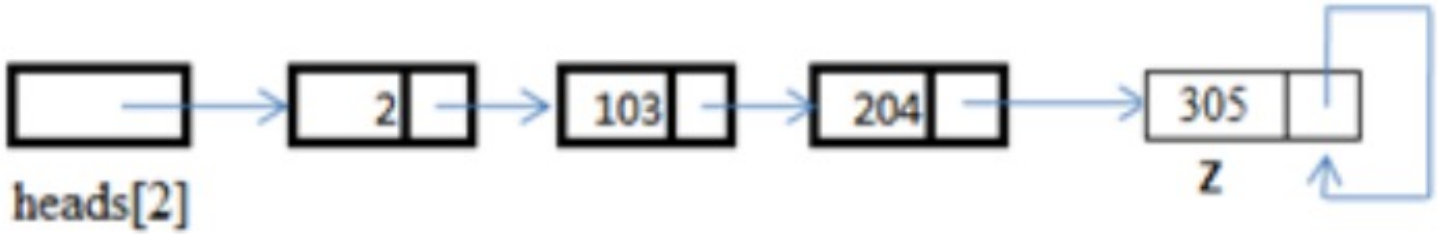
Phương pháp nối kết trực tiếp

- Tìm một phần tử trong bảng băm: k = 305
  - t = heads[k%M] = heads[305%101]
    = heads[2]



Phương pháp nối kết trực tiếp

- Tìm một phần tử trong bảng băm: k = 305
  - t = heads[k%M] = heads[305%101]
    = heads[2]
  - Gán z->key = 305 (cầm canh)




```
Node *t = heads[k % M];
z->key = k;
while (t->key < k)
{   t = t->next; }
if (t->key != k)
    return z;
return t;
```

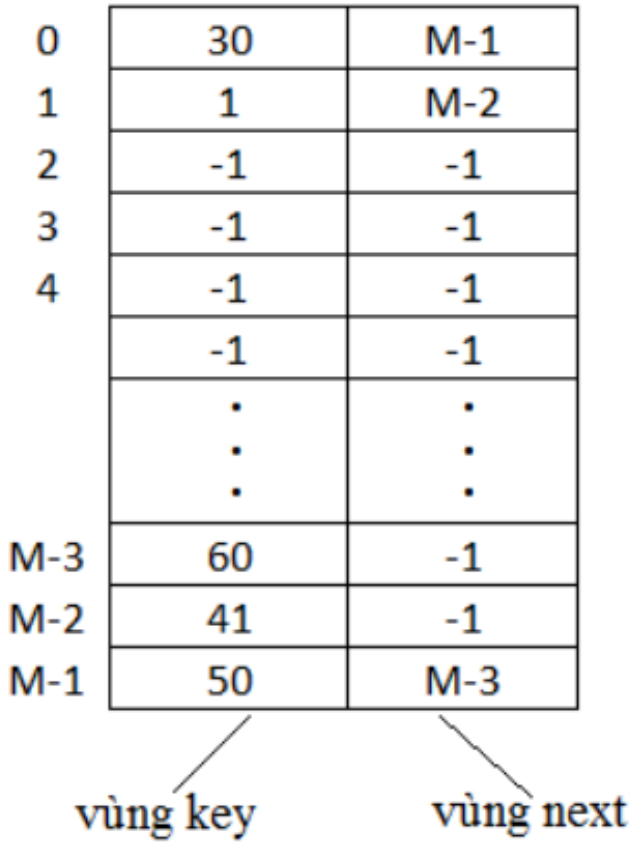
Phương pháp nối kết trực tiếp

- Tìm một phần tử trong bảng băm

```
Node * search(int k)
{
    Node *t = heads[k % M];

    z->key = k;
    while (t->key < k)
    {
        t = t->next;
    }

    if (t->key != k)
        return z;
    return t;
}
```

Phương pháp nối kết hợp nhất

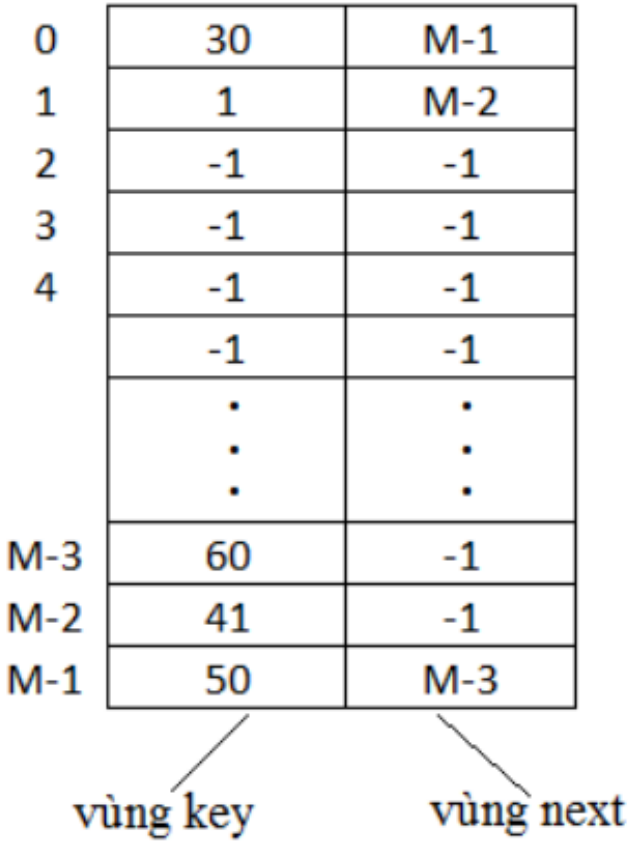
- Phương pháp nối kết hợp nhất (coalesced chaining)
  - Sử dụng mảng danh sách đặc chứa M phần tử (0 → M-1)
  - Mỗi phần tử trong bảng băm chứa 2 thành phần
    - key: chứa giá trị phần tử
    - Next: chứa vị trí của phần tử kế tiếp trong trường hợp xảy ra đụng độ



Phương pháp nối kết hợp nhất

❖Khai báo

```
#define M 100

struct Node
{
    int key;
    int next;
};

Node T[M];
int r = M - 1;
```



Phương pháp nối kết hợp nhất

❖Khởi tạo

```
void init()
{
    for (int i = 0; i < M; i++)
    {
        T[i].key = -1;
        T[i].next = -1;
    }
}
```

0	-1	-1
1	-1	-1
2	-1	-1
3	-1	-1
4	-1	-1
	-1	-1
	. . .	. . .
M-3	-1	-1
M-2	-1	-1
M-1	-1	-1

Phương pháp nối kết hợp nhất

- Thêm: 30, 1, 50, 41, 60;

h(k) = x % 10

0	30	-1
1	-1	-1
2	-1	-1
3	-1	-1
4	-1	-1
	-1	-1
	. . .	. . .
M-3	-1	-1
M-2	-1	-1
M-1	-1	-1

Phương pháp nối kết hợp nhất

❖Thêm: 30, 1, 50, 41, 60; h(k) = x % 10

0	30	-1
1	1	-1
2	-1	-1
3	-1	-1
4	-1	-1
	-1	-1
	. . .	. . .
M-3	-1	-1
M-2	-1	-1
M-1	-1	-1

❖Thêm: 30, 1, 50, 41, 60; h(k) = x % 10

0	30	M-1
1	1	-1
2	-1	-1
3	-1	-1
4	-1	-1
	-1	-1
	. . .	. . .
M-3	-1	-1
M-2	-1	-1
M-1	50	-1

Phương pháp nối kết hợp nhất

❖Thêm: 30, 1, 50, 41, 60; h(k) = x % 10

0	30	M-1
1	1	M-2
2	-1	-1
3	-1	-1
4	-1	-1
	-1	-1
	. . .	. . .
M-3	-1	-1
M-2	41	-1
M-1	50	-1

❖Thêm: 30, 1, 50, 41, 60; h(k) = x % 10

0	30	M-1
1	1	M-2
2	-1	-1
3	-1	-1
4	-1	-1
	-1	-1
	. . .	. . .
M-3	60	-1
M-2	41	-1
M-1	50	M-3

Phương pháp nối kết hợp nhất

- Hàm băm

```
int h(int x)
{
      return (x % 10);
}
```

Phương pháp nối kết hợp nhất

- Tìm một phần tử trong bảng băm

```
int search(int x)
{
    int i = h(x);
    while (x != T[i].key && i != -1)
    {
        i = T[i].next;
    }
    if (x == T[i].key)
        return i;
    else
        return -1;
}
```

PP nối kết hợp nhất

- Thêm một phần tử vào bảng băm

```
int insert(int x)
{
     int i, j;
     i = h(x);
     if ((T[i].key != x) && (T[i].key != -1))
     {
          do {
               j = i;
               i = T[i].next;
          } while ((T[i].key != x) && (i != -1));

          if (i == -1)
          {
               while (r != -1 && T[r].key != -1)    r--;
               if (r < 10) return -1;
               if (r != -1) T[j].next = r;
               i = r;
          }
     }
     if (i != -1 && T[i].key != x)
          T[i].key = x;
     return i;
}
```

Phương pháp địa chỉ mở

- Phương pháp dò tuyến tính (Linear probing)

$$H(k,i) = (h(k) + i) mod\ M$$

- Phương pháp dò bậc 2 (Quadratic probing)

$$H(k,i) = (h(k) + i^2) mod\ M$$

- Phương pháp băm kép (Double hashing)

$$H(k,i) = \big(h_1(k) + i * h_2(k)\big) mod\ M$$

- ⇒Sinh viên tham khảo tài liệu [2], [3].

Bài tập

Bài 1: Dùng cấu trúc bảng băm – phương pháp kết nối trực tiếp, quản lý 5000 phần tử kiểu số int.

a. Khai báo cấu trúc bảng băm
b. Viết thủ tục khởi bảng băm rỗng
c. Viết thủ tục thêm một phần tử vào bảng băm
d. Viết thủ tục tìm kiếm một phần tử trong bảng băm

Bài tập

Bài 2: Dùng cấu trúc bảng băm – phương pháp kết nối hợp nhất, quản lý 200 phần từ kiểu số int.

a. Khai báo cấu trúc bảng băm
b. Viết thủ tục khởi bảng băm rỗng
c. Viết thủ tục thêm một phần tử vào bảng băm
d. Viết thủ tục tìm kiếm một phần tử trong bảng băm

Bài tập làm thêm

Bài 3: hãy mô tả (vẽ hình minh họa) việc thêm tuần tự các phần tử sau vào bảng băm có cấu trúc nối kết trực tiếp (quản lý khoảng 2000 phần tử kiểu int):

10, 20, 30, 45, 60, 70, 210, 220, 221, 440, 360, 470

Bài tập làm thêm

Bài 4: hãy mô tả (vẽ hình minh họa) việc thêm tuần tự các phần tử sau vào bảng băm có cấu trúc kết nối hợp nhất (quản lý khoảng 2000 phần tử kiểu int):

3, 7, 10, 20, 30, 45, 60, 70, 213, 222, 228, 443, 367, 470, 503, 507.

Bài tập nâng cao

Bài 5: Quản lý một từ điển Anh – Việt gồm khoảng 50.000 từ

a. Khai báo cấu trúc từ điển
b. Viết thủ tục tìm kiếm một từ
c. Viết thủ tục load các từ từ file lên bộ nhớ
d. Viết thủ thêm một từ mới

Tài liệu tham khảo

[1] Truong, Le Xuan. Cấu trúc dữ liệu. Trường ĐH Mở Tp.HCM, 2015 (chapter 4).

[2] Cormen, Thomas H., et al. *Introduction to algorithms*. MIT press, 2009 (III.11).

[3] Drozdek, Adam. *Data Structures and algorithms in C++*. Cengage Learning, 2012 (chapter 10).