Bảng băm

Nội dung

- Giới thiệu về bảng băm
- Hàm băm.
- Các phương pháp giải quyết đụng độ
  1. Phương pháp kết nối trực tiếp
  2. Phương pháp kết nối hợp nhất
  3. Phương pháp dò tuyến tính
  4. Phương pháp dò bậc 2
  5. Phương pháp băm kép
- Phân tích Phép băm

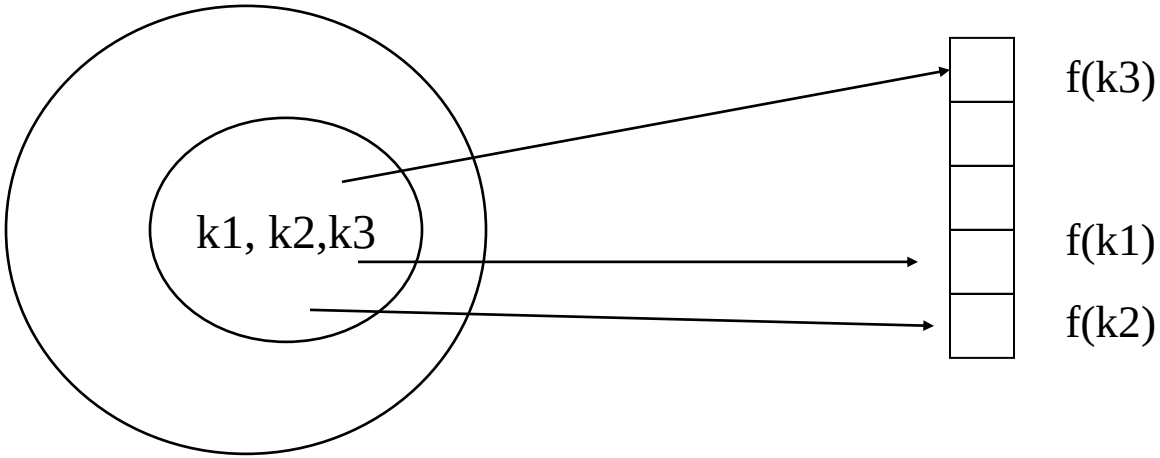
Giới thiệu Bảng băm

- **Bảng băm (hash table – HT):** là một cấu trúc dữ liệu cho phép tham chiếu trực tiếp các phần tử trong bảng thông qua việc biến đổi số học trên khoá của phần tử để có được địa chỉ tương ứng của phần tử trong bảng.

- HT có M vị trí được đánh chỉ mục từ 0 đến M-1, M là kích thước của bảng băm → Mảng?

HÀM BĂM (Hash functions)

- **Hàm băm (hash function):** Là hàm biến đổi giá trị khoá thành địa chỉ trong bảng băm.
- Khóa có thể là dạng số hay dạng chuỗi.
- Địa chỉ tính ra được là số nguyên trong khoảng 0 đến M-1 với M là số địa chỉ có trên bảng băm



HÀM BĂM (Hash functions)

Giả sử khoá k là số nguyên

Hàm lấy số dư:

$$f(k) = k \bmod M$$

Với M là số địa chỉ trên bảng băm

Hàm băm(Hash Functions)

Hàm băm tốt thỏa mãn các điều kiện sau:

- Tính toán nhanh.
- Các khoá được phân bố đều trong bảng.
- Ít xảy ra đụng độ.

Giải quyết vấn đề băm với các khoá không phải là số nguyên:

- Tìm cách biến đổi khoá thành số nguyên

  Ví dụ: Loại bỏ dấu '-' trong 9635-8904 → đưa về số nguyên 96358904

  Đối với chuỗi, sử dụng mã ASCCI của các ký tự
- Sau đó sử dụng các hàm băm chuẩn trên số nguyên.

Sự đụng độ (collision)

Sự đụng độ là hiện tượng các khóa khác nhau nhưng băm cùng địa chỉ như nhau.

Khi key1<>key2 mà f(key1)=f(key2) chúng ta nói nút có khóa key1 đụng độ với nút có khóa key2.

Thực tế người ta giải quyết sự đụng độ theo hai phương pháp:

- Phương pháp nối kết.
- Phương pháp băm lại.

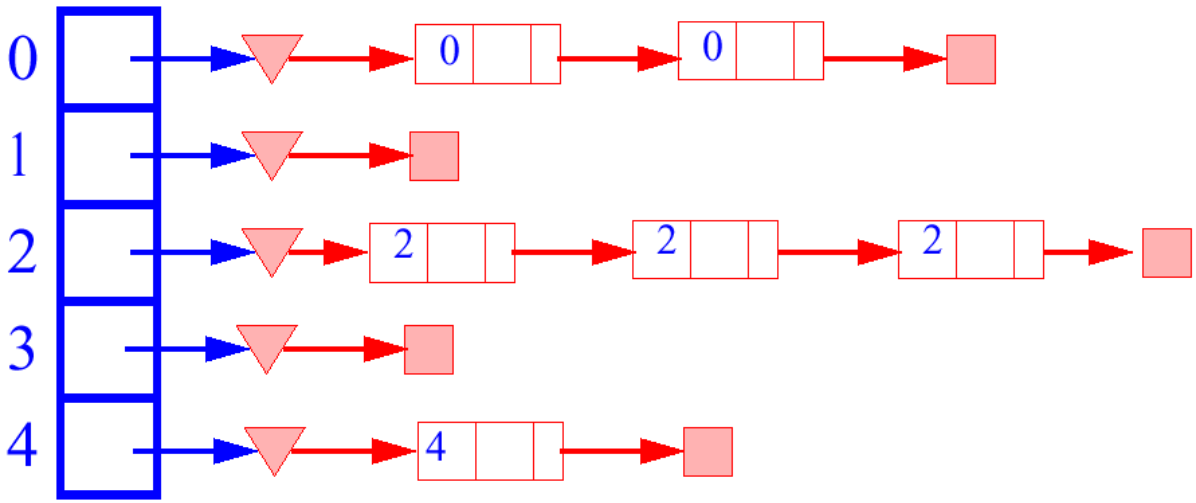
Các phương pháp xử lý đụng độ

- Phương pháp nối kết.
  - Phương pháp nối kết trực tiếp
  - Phương pháp nối kết hợp nhất
- Phương pháp băm lại.
  - Phương pháp dò tuyến tính
  - Phương pháp dò tuyến tính bậc hai
  - Phương pháp băm kép

I. Phương pháp nối kết trực tiếp:

Các nút bị băm cùng địa chỉ (các nút bị xung đột) được gom thành một danh sách liên kết.

Các nút bị xung đột tại địa chỉ i được nối kết trực tiếp với nhau qua danh sach liên kết i.



Phương pháp nối kết trực tiếp *(tt)*

```
#define M 100
typedef struct nodes
{
        int key;
        struct nodes *next
}NODE;

//khai báo kiểu con trỏ chỉ nút
typedef NODE *PHNODE;

// khai báo mảng HASHTABLE chứa M con trỏ
typedef PHNODE HASHTABLE[M];
```

Phương pháp nối kết trực tiếp*(tt)*

Nhận xét:

- ”băm” n nút vào M danh sách liên kết (M bucket).
- Tốc độ Truy xuất phụ thuộc vào việc lựa chọn hàm băm sao cho băm đều n nút của bảng băm cho M bucket.
- Nếu chọn M càng lớn thì tốc độ thực hiện các phép toán trên bảng băm càng nhanh tuy nhiên không hiệu quả về bộ nhớ. Chúng ta có thể điều chỉnh M để dung hòa giữa tốc độ truy xuất và dung lượng bộ nhớ;
- Nếu chọn M=n thời gian truy xuất tương đương với truy xất trên mảng, song tốn bộ nhớ.
- Nếu chọn M =n/k (k =2,3,4,…) thì ít tốn bộ nhớ hơn k lần, nhưng tốc độ chậm đi k lần.

Phương pháp nối kết hợp nhất

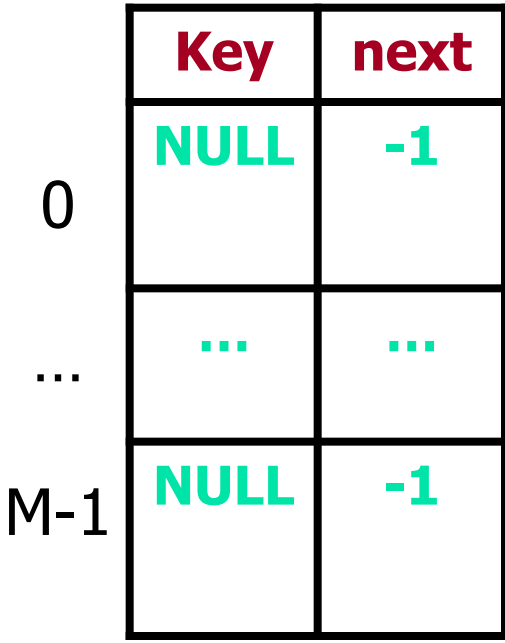
II. Bảng băm với phương pháp nối kết hợp nhất (*coalesced Chaining Method*):

Bảng băm trong trường hợp này được cài đặt bằng mảng có M nút.

Mỗi nút của bảng băm là một mẫu tin có 2 trường:

- Trường key: chứa các khóa node
- Trường next: con trỏ chỉ node kế tiếp nếu có xung đột.

Khởi động bảng băm: tất cả trường key được gán NULL, tất cả trường next được gán -1.



Phương pháp nối kết hợp nhất

- Khi thêm một nút có khóa key vào bảng băm,hàm băm f(key) sẽ xác định địa chỉ i trong khoảng từ 0 đến M-1.
- Nếu chưa bị xung đột thì thêm nút mới vào địa chỉ này.
- Nếu bị xung đột thì nút mới được cấp phát là nút trống phía cuối mảng. Cập nhật liên kết next sao cho các nút bị xung đột hình thành một danh sách liên kết.
- Khi tìm một nút có khóa key trong bảng băm,hàm băm f(key) sẽ xác định địa chỉ i trong khoảng từ 0 đến M-1, tìm nút khóa key trong danh sách liên kết xuất phát từ địa chỉ i.

Phương pháp nối kết hợp nhất

VD:

Cho bảng băm có:

- Tập khóa: tập các số tự nhiên
- Tập địa chỉ gồm 10 địa chỉ (M=10) (từ địa chỉ 0 đến 9).
- Hàm băm f(key)=M%10

Key: 11 32 21 41 53

Hash: 1 2 1 1 3

Đụng độ ➔ thêm vào nút trống cuối mảng

0	NULL	-1
1	11	9
2	21	-1
3	53	-1
...	NULL	-1
8	41	-1
9	32	8

Phương pháp nối kết hợp nhất

a. Khai báo cấu trúc bảng băm:

```
//Khai bao cau truc mot nut cua bang bam
typedef struct
{
        int key;  //khoa cua nut tren bang bam
        int next; //con tro chi nut ke tiep khi co xung dot
}NODE;

//Khai bao bang bam
typedef NODE HASHTABLE[M];
```

Phương pháp dò tuyến tính

(Linear Probing Method)

- Bảng băm trong trường hợp này được cài đặt bằng danh sách kề có M nút, mỗi nút của bảng băm là một mẫu tin có một trường key để chứa khoá của nút.
- Khi khởi động bảng băm thì tất cả trường key được gán NULLKEY.
- Khi thêm nút có khoá key vào bảng băm, hàm băm f(key) sẽ xác định địa chỉ i trong khoảng từ 0 đến M-1.

  f(key) = key % M

  - Nếu chưa bị xung đột thì thêm nút mới vào địa chỉ này.
  - Nếu bị xung đột thì băm lại lần 1- f1 sẽ xét địa chỉ kế tiếp,nếu lại bị xung đột thì băm lại lần 2 - f2 sẽ xét địa chỉ kế tiếp.
    - f(key) = (f(key) + i) % M
    - (i là lần băm lại thứ i, f(key) là hàm băm ban đầu)
  - Quá trình cứ thế cho đến khi nào tìm được địa chỉ trống và thêm nút vào địa chỉ này.

- **Lưu ý**:
  - địa chỉ dò tìm kế tiếp bắt đầu từ đầu bảng nếu đã dò đến cuối bảng.
  - Giải thuật dừng khi đã tìm được vị trí để thêm hoặc các địa chỉ băm bị lập lại

0	nullkey
1	nullkey
2	nullkey
3	nullkey
...	nullkey
M-1	nullkey

Thêm các nút 32, 53, 22, 92, 17, 34, 24, 37, 56 vào bảng băm.

Hàm băm: f(key) = key % 10

Hàm băm lại: f_i(key) = (f(key) + i) % 10

0	NULL	0	NULL	0	NULL	0	NULL	0	56
1	NULL	1	NULL	1	NULL	1	NULL	1	NULL
2	32	2	32	2	32	2	32	2	32
3	53	3	53	3	53	3	53	3	53
4	NULL	4	22	4	22	4	22	4	22
5	NULL	5	92	5	92	5	92	5	92
6	NULL	6	NULL	6	34	6	34	6	34
7	NULL	7	NULL	7	17	7	17	7	17
8	NULL	8	NULL	8	NULL	8	24	8	24
9	NULL	9	NULL	9	NULL	9	37	9	37

Phương pháp dò tuyến tính

- Khi tìm một nút có khoá key trong bảng băm, hàm băm f(key) sẽ xác định địa chỉ i trong khoảng từ 0 đến M-1, tìm nút có khoá key trong khối đặt chứa các nút xuất phát từ địa chỉ i.

Phương pháp dò tuyến tính

Nhận xét:

- Chúng ta thấy bảng băm này chỉ tối ưu khi băm đều, tốc độ truy xuất lúc này có bậc 0(1).
- Trường hợp xấu nhất là băm không đều hoặc bảng băm đầy, lúc này hình thành một khối đặc có n nút nên tốc độ truy xuất lúc này có bậc 0(n).

Phương pháp dò tuyến tính

```
//khai bao cau truc mot node cua bang bam
typedef struct
{
        int key;  //khoa cua nut tren bang bam
}NODE;

//Khai bao bang bam co M nut
NODE HASHTABLE[M];
int N; //bien toan cuc chi so nut hien co tren bang bam
```

Phương pháp dò tuyến tính bậc hai

- Bảng băm dùng phương pháp dò tuyến tính bị hạn chế do rải các nút không đều,bảng băm với phương pháp dò bậc hai rải các nút đều hơn.
- Bảng băm trong trường hợp này được cài đặt bằng danh sách kề có M nút, mỗi nút của bảng băm là một mẫu tin có một trường key để chứa khóa các nút
- Khi khởi động bảng băm thì tất cả trường key bị gán NULLKEY.
- Khi thêm nút có khóa key vào bảng băm, hàm băm f(key) sẽ xác định địa chỉ i trong khoảng từ 0 đến M-1.

Phương pháp dò tuyến tính bậc hai

- Nếu chưa bị xung đột thì thêm nút mới vào địa chỉ i này.
- Nếu bị xung đột thì hàm băm lại lần 1, f_1 sẽ xét địa chỉ cách i 1^2, nếu lại bị xung đột thì hàm băm lại lần 2, f_2 sẽ xét địa chỉ cách i 2^2 ,..., quá trình cứ thế cho đến khi nào tìm được trống và thêm nút vào địa chỉ này.
- Khi tìm môt nút có khóa key trong bảng băm thì xét nút tại địa chỉ i=f(key), nếu chưa tìm thấy thì xét nút cách i 1^2,2^2,...,quá trình cứ thế cho đến khi tìm được khóa (trường hợp tìm thấy) hoặc rơi vào địa chỉ trống(trường hợp không tìm thấy).

Phương pháp dò tuyến tính bậc hai

Hàm băm lại của phương pháp dò bậc hai là truy xuất các địa chỉ cách bậc 2.

Hàm băm lại được biểu diễn bằng công thức sau:

$f_i(key)=(f(key) + i^2) \% M$ với f(key) là hàm băm chính của bảng băm (i là lần băm lại thứ i).

Nếu đã dò đến cuối bảng thì trở về dò lại từ đầu bảng.

Bảng băm với phương pháp dò bậc hai nên chọn số địa chỉ M là số nguyên tố.

Thêm vào các khóa 10, 15, 16, 20, 30, 25, 26, 36

Hàm băm: f(key) = key % 10

Hàm băm lại: $f_i(key) = (f(key) + i^2) \% 10$

0	10	0	10	0	10	0	10	0	10
1	NULL	1	20	1	20	1	20	1	20
2	NULL	2	NULL	2	NULL	2	NULL	2	36
3	NULL	3	NULL	3	NULL	3	NULL	3	NULL
4	NULL	4	NULL	4	30	4	30	4	30
5	15	5	15	5	15	5	15	5	15
6	16	6	16	6	16	6	16	6	16
7	NULL	7	NULL	7	NULL	7	26	7	26
8	NULL	8	NULL	8	NULL	8	NULL	8	NULL
9	NULL	9	NULL	9	25	9	25	9	25

Phương pháp dò tuyến tính bậc hai *(tt)*

```
//Khai bao nut cua bang bam
typedef struct
{
        int key;        //Khoa cua nut tren bang bam
}NODE;

//Khai bao bang bam co M nut
NODE HASHTABLE[M];
int  N; //Bien toan cuc chi so nut hien co tren bang bam
```

Phương pháp dò tuyến tính bậc hai *(tt)*

Nhận xét bảng băm dùng phương pháp dò bậc hai:

Nên chọn số địa chỉ M là số nguyên tố. Khi khởi động bảng băm thì tất cả M trường key được gán NULL, biến toàn cục N được gán 0.

Bảng băm đầy khi N = M-1, và nên dành ít nhất một phần tử trống trên bảng băm.

Bảng băm này tối ưu hơn bảng băm dùng phương pháp dò tuyến tính do rải rác phần tử đều hơn, nếu bảng băm chưa đầy thì tốc độ truy xuất có bậc 0(1). Trường hợp xấu nhất là bảng băm đầy vì lúc đó tốc độ truy xuất chậm do phải thực hiện nhiều lần so sánh.

Phương pháp băm kép

Mô tả:

- *Cấu trúc dữ liệu:* Bảng băm này dùng hai hàm băm khác nhau với mục đích để rải rác đều các phần tử trên bảng băm.

Chúng ta có thể dùng hai hàm băm bất kì, ví dụ chọn hai hàm băm như sau:

f1(key)= key %M.

f2(key) =(M-2)-key %(M-2).

bảng băm trong trường hợp này được cài đặt bằng danh sách kề có M phần tử, mỗi phần tử của bảng băm là một mẩu tin có một trường key để lưu khoá các phần tử.

Phương pháp băm kép *(tt)*

- *Khi khởi động* bảng băm,tất cả trường key được gán NULL.
- *Khi thêm phần tử* có khoá key vào bảng băm, thì a=f1(key) và b=f2(key) sẽ xác định địa chỉ a và b trong khoảng từ 0 đến M-1:

Nếu chưa bị xung đột thì thêm phần tử mới tại địa chỉ a này.

Nếu bị xung đột thì hàm băm lại lần i:

$$f_i(key) = (a + i*b)\%M$$

- **Lưu ý**:
  - địa chỉ dò tìm kế tiếp bắt đầu từ đầu bảng nếu đã dò đến cuối bảng.
  - Giải thuật dừng khi đã tìm được vị trí để thêm hoặc các địa chỉ băm bị lập lại

Thêm vào các khóa 10, 15, 16, 20, 30, 25, 26, 36 :

f1(key)= key % M

f2(key) =(M-2)-key %(M-2) với M=10

0	10	0	10	0	10	0	10	0	10
1	NULL	1	NULL	1	NULL	1	NULL	1	NULL
2	NULL	2	NULL	2	30	2	30	2	30
3	NULL	3	NULL	3	NULL	3	NULL	3	36
4	NULL	4	20	4	20	4	20	4	20
5	15	5	15	5	15	5	15	5	15
6	16	6	16	6	16	6	16	6	16
7	NULL	7	NULL	7	NULL	7	NULL	7	NULL
8	NULL	8	NULL	8	NULL	8	26	8	26
9	NULL	9	NULL	9	25	9	25	9	25

Phương pháp băm kép *(tt)*

- *Khi tìm kiếm* một phần tử có khoá key trong bảng băm, hàm băm i=f1(key) và j=f2(key) sẽ xác định địa chỉ i và j trong khoảng từ 0 đến M-1. Xét phần tử tại địa chỉ i, nếu chưa tìm thấy thì xét tiếp phần tử i+j, i+2j, …, quá trình cứ thế cho đến khi nào tìm được khoá (trường hợp tìm thấy) hoặc bị rơi vào địa chỉ trống (trường hợp không tìm thấy).

Phương pháp băm kép *(tt)*

```
//Khai bao phan tu cua bang bam
typedef struct
{
        int key;      //khoa cua nut tren bang bam
}NODE;
//khai bao bang bam co M nut
struct node HASHTABLE[M];
int N;
//bien toan cuc chi so nut hien co tren bang bam
```

Tóm tắt về bảng băm

- Bảng băm đặt cơ sở trên mảng.
- Phạm vi các giá trị khóa thường lớn hơn kích thước của mảng.
- Một giá trị khóa được băm thành một chỉ mục của mảng bằng hàm băm.
- Việc băm một khóa vào vào một ô đã có dữ liệu trong mảng gọi là sự đụng độ.
- Sự đụng độ có thể được giải quyết bằng hai phương pháp chính: Phương pháp nối kết và phương pháp băm lại.

Tóm tắt về bảng băm (tt)

Trong phương pháp băm lại, các mục dữ liệu được băm vào các ô đã có dữ liệu sẽ được đưa vào ô khác trong mảng.

Trong phương pháp nối kết, mỗi phần tử trong mảng có một danh sách liên kết. Các mục dữ liệu được băm vào các ô sẽ được đưa vào danh sách ở ô đó.

Vấn đề Hàm băm

a. Hàm băm dùng phương pháp chia: $h(k) = k \bmod m$

m là kích thước bảng băm, k là khóa.

b. Hàm băm dùng phương pháp nhân: $h(k) = \lfloor m(k\,A \bmod 1) \rfloor$

Knuth đề nghị $A = 0.6180339887$

Tóm tắt về bảng băm (tt)

Kích thước bang băm	Số lần đụng độ	
	PP chia	PP Nhân
200	698	699
512	470	466
997	309	288
1024	301	292

- Theo bảng trên kết qủa cho thấy kích thước bảng băm tỷ lệ nghịch với số lần đụng độ. Số lần đụng độ còn phụ thuộc vào phương pháp sử dụng hàm băm.
- Hệ số tải: là tỉ số giữa các mục dữ liệu trong một bảng băm với kích thước bảng băm.
- Hệ số tải cực đại trong phương pháp băm lại khoảng 0,5. Đối với băm kép ở Hệ số tải này (0,5), các phép tìm kiếm sẽ có chiều dài thăm dò trung bình là 2.

Tóm tắt về bảng băm (tt)

- Trong phương pháp băm lại , thời gian tìm kiếm sẽ là vô cực khi hệ số tải đạt đến 1.
- Điều quan trọng trong phương pháp băm lại là bảng băm không bao giờ được quá đầy.
- Phương pháp nối kết thích hợp với hệ số tải là 1.
- Với hệ số tải này, chiều dài thăm dò trung bình là 1,5 khi phép tìm thành công, và là 2.0 khi phép tìm thất bại.

Tóm tắt về bảng băm (tt)

- Chiều dài thăm dò trong phương pháp nối kết tăng tuyến tính theo hệ số tải.
- Kích thước của bảng băm thường là số nguyên tố. Điều này đặc biệt quan trọng trong thăm dò bậc hai và trong phương pháp nối kết.
- Các bảng băm có thể dùng cách lưu trữ ngoại. Một cách để thực hiện việc này là cho các phần tử trong bảng băm chứa số lượng các khối của tập tin trên đĩa