Hàm băm (Phép biến đổi khóa)

Phép biến đổi khóa là 1 phương pháp tham khảo trực tiếp các phần tử trong 1 bảng thông qua việc biến đổi trên những khóa (key) để có được địa chỉ tương ứng của những phần tử ở trong bảng.

Phép biến đổi khóa là 1 ánh xạ H từ tập các khóa K vào tập các địa chỉ A:

H : K → A

Tổ chức dữ liệu được dùng cho phép biến đổi khóa là dãy. Do đó, **H là 1 ánh xạ biến đổi khóa thành chỉ số của dãy**.

Trong phép biến đổi khóa, thông thường tập các khóa lớn hơn rất nhiều so với tập các đ/c bộ nhớ (chỉ số của mảng).

Ví dụ tập khóa mà mỗi khóa có 10 ký tự để chỉ định 1 tập 1000 người, bộ ký tự có 26 ký tự, do đó tập hợp khóa có 26^{10} khóa có thể được ánh xạ vào tập 1000 chỉ số. Như vậy H là 1 hàm nhiều – một (n – 1)

Để thực hiện phép biến đổi khóa, ta có 2 bước:

- Tính toán hàm H để biến đổi khóa cần tìm thành địa chỉ trong bảng.
- Giải quyết sự đụng độ cho những khóa khác nhau mà có cùng 1 địa chỉ trong dãy. Một trong những phương pháp giải quyết sự đụng độ là dùng danh sách liên kết. Một phương pháp khác để giải quyết sự đụng độ với thời gian nhanh là dùng danh sách tuyến tính có kích thước cố định.

I. **Hàm biến đổi khóa**: yêu cầu của 1 hàm biến đổi khóa là khả năng phân bố đều trên miền giá trị của địa chỉ. Gọi M là số các phần tử được chứa trong bộ nhớ (M là số nguyên tố), hàm băm sẽ biến đổi các khóa (thường là các số nguyên hoặc là các chuỗi ký tự ngắn) thành các số nguyên trong đoạn [0..M-1].

Giả sử các khóa là những số nguyên, ta có thể dùng hàm băm H(k) là:

H(k) = k % M, với k là khóa

Ví dụ: Bảng băm có 101 phần tử và ta phải tính địa chỉ của 1 khóa có 4 ký tự là AKEY. Nếu khóa được mã hóa theo bảng sau đây:

Ký tự	Nhị phân	Thập phân
A	00001	1
B	00010	2
C	00011	3
…….		

Nghĩa là ký tự thứ i được mã hóa thành số nhị phân của số i, lúc này khóa AKEY được mã hóa thành:

00001 01011 00101 11001 tương đương với số thập phân 44217. Vậy khóa AKEY có địa chỉ trong bảng là 44217 % 101 = 80.

Ví dụ mã của khoá VERYLONGKEY là:

10110	00101	10010	11001	01100	01111	01110	00111	01011	00101	11001

Giải thuật tính hàm băm :

```
h=key[0]-'A'+1;
for (i=1 ; i<keysize ; i++)
        h= h+ (key[i]-'A'+1);
h = h % M;
```

với key[i] là số thứ tự của ký tự thứ i của khóa (tính từ trái sang phải)
Như vậy, khóa VERYLONGKEY có địa chỉ là h = 97 (với M=101)

II. **GIẢI QUYẾT SỰ ĐỤNG ĐỘ**:

1. Phương pháp nối kết:
   a. Phương pháp nối kết trực tiếp: Ứng với mỗi địa chỉ của bảng, ta có 1 danh sách liên kết chứa các phần tử có khóa khác nhau mà có cùng 1 địa chỉ. Như vậy ta sẽ có 1 bảng băm gồm M phần tử chứa địa chỉ đầu của các danh sách liên kết.

```
const  int M = 101 ;
struct node {
    int key;     // khóa
    int info  ;  // nội dung
    struct node * next ;
};
typedef node * ref;
ref  heads [M]
```

↳ *Khởi tạo bảng băm*: cho các địa chỉ đầu của các dslk trong heads về NULL.
Bảng heads:

NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL
0	1	2	3	4	5	22			M-1

↳ *Thêm 1 khóa k mới vào bảng băm*:
Input : khóa k
- Dùng hàm băm để chuyển khóa k sang số h (địa chỉ của khóa k trong bảng băm)
- Đưa k vào dslk tại vị trí h.

Ví dụ: k= 1234 thì h = k % M = k % 101 = 22. Lúc này, ta sẽ đưa k vào dslk ở vị trí heads[22]

↳ *Tìm kiếm 1 khóa k có trong bảng băm hay không,* nếu tìm thấy thì hàm sẽ trả về địa chỉ của phần tử chứa khóa, nếu không tìm thấy thì trả về NULL
- Dùng hàm băm để chuyển khóa k sang số h (địa chỉ của khóa k trong bảng băm)
- Tìm khóa k tại vị trí heads[h]

b. Phương pháp nối kết hợp nhất: Tổ chức bảng băm của phương pháp này là 1 danh sách đặc mà mỗi phần tử có 1 vùng chứa chỉ số của phần tử kế tiếp khi có sự đụng độ xảy ra.

Giả sử các khóa có giá trị hàm băm và thứ tự thêm vào như sau:

Khóa: A C B D

Hash: 0 1 0 0

0	A	M-1	
1	C	-1	
2			
…			
M-2	D	-1	
M-1	B	M-2	

Ta khai báo:

```
const  int M = 101 ;
const int free = -1;
struct item {
    int key;     // khóa
    int info  ;  // nội dung
    int  next ;
};
item T [M+1];
int r;
```

Biến r là 1 thành phần của bảng băm T, nó dùng để trợ giúp việc tìm kiếm vị trí trống đầu tiên từ phía dưới bảng trở lên.

Khởi tạo : r = M;

Khi thực hiện phép thêm vào bảng băm, biến r đảm bảo rằng các phần tử T[j] đã được sử dụng với r <= j <= M

↳ *Khởi tạo bảng băm* :

```
for (int i =0 ; i < M ; i++ ) { T[i].key =T[i].next=free;}
r=M;
```

Bảng T:

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

↳ *Thêm 1 khóa k mới vào bảng băm T*: Giá trị trả về của hàm là vị trí của phần tử mới thêm vào hoặc số 0 nếu bảng băm bị đầy. Nếu đã có khóa k này trong T thì hàm này chỉ trả về vị trí của khóa này ở trong bảng chứ không thêm vào.

Thuật toán : Input : khóa k

```
i = H(k); // H là hàm băm trả về đ/c  0<=i < M
Nếu T[i].key <> k và T[i].next <> free thì :
{  Repeat
        j=i;
```

```
        i=T[i].next
    Until  (i=0 hoặc T[i].key = k)

    Nếu i =0  // không tìm thấy k
    {   Repeat
            r--;
        Until T[r].next = free;
        Nếu r <> 0 thì T[j].next=r
        i=r;
    }
}
Nếu (i <> 0 và T[i].key <> k ) thì
{       T[i].key = k ;  T[i].next = 0;
}
return i;
```

Lưu ý: Phần tử tại vị trí 0 luôn là phần tử trống (có khóa key = free), nó được dùng để đảm bảo việc tìm kiếm vị trí trống luôn luôn kết thúc thành công (nó đóng vai trò của phần tử cầm canh).

↳ *Tìm kiếm 1 khóa k trong bảng băm : sinh viên tự cài đặt*

2. Phương pháp thử tuyến tính: Khi có đụng độ xảy ra thì ta tìm đến vị trí kế tiếp nào đó ở trong bảng cho đến khi nào tìm thấy phần tử mong muốn hoặc vị trí kế tiếp là vị trí trống. Trong p.pháp này, ta gọi M (số nguyên tố) là số phần tử của bảng băm và N là số phần tử đã sử dụng. Bảng băm được gọi là đầy khi N = M-1.

Ta khai báo:

```
const  int M = 101 ;
const int free = maxint;
struct item {
    int key;     // khóa
    int info  ;  // nội dung
};
item T [M];
int N;
```

↳ *Khởi tạo bảng băm* :

```
for (int i =0 ; i < M ; i++ ) T[i].key= free;  // vị trí trống
N=0;
```

↳ *Thêm 1 khóa k mới vào bảng băm T*: Input : khóa k

Thực hiện thêm 1 phần tử có khóa là k vào trong bảng băm và trả về vị trí của phần tử này ở trong bảng hoặc trả về giá trị M nếu bảng băm bị đầy.

Thuật toán:

```
Nếu N = M-1 thì return M;
x= H(k) ;
while (T[x].key <> free)    // tìm vị trí trống
```

```
{ x++;
  if ( x>=M ) x -= M;
}
T[x].key=k;
N++;
return x;
```

Ví dụ: Giả sử ta muốn tạo bảng băm với dãy khóa sau (M=19):

Khóa :A S E A R C H I N G E X A M P L E

Hash : 1 0 5 1 18 3 8 9 14 7 5 5 1 13 16 12 5

Bảng :

0		S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S
1	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A
2				A	A	A											
3						C											
4													A				
5			E	E	E	E	E	E	E	E	E	E	E	E	E	E	E
6											E						
7										G							
8							H										
9								I									
10												X					
11																	E
12																L	
13														M			
14									N								
15																	
16															P		
17																	
18					R	R											

↳ *Tìm kiếm 1 khóa k trong bảng băm* : Sinh viên tự cài đặt

Bài tập:

Tra cứu từ điển: tổ chức từ điển theo cấu trúc danh sách TUYẾN TÍNH. Mỗi từ gồm có các thông tin: Từ, loại từ, các nghĩa việt (theo cấu trúc danh sách liên kết, các ví dụ (theo cấu trúc danh sách liên kết) .

Chương trình có các chức năng: nhập từ mới, hiệu chỉnh từ, xóa từ, TRA TỪ;

a. Viết khai báo để lưu trữ các từ theo cấu trúc mảng băm;

b. Giả sử ta có 1 từ tiếng Anh lưu trong chuỗi S. Viết chương trình con tìm S trong từ điển