BẢNG BĂM

Bùi Tiến Lên

01/01/2017

KHOA CÔNG NGHỆ THÔNG TIN
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

LÝ THUYẾT ĐỒNG DƯ

Các khái niệm

Định nghĩa 1

Cho số nguyên dương $n \in \mathbb{N}$, hai số nguyên $a, b \in \mathbb{Z}$ được gọi là **đồng dư** theo mô-đun n nếu cả hai có cùng số dư khi chia cho n. Được ký hiệu

$$a \equiv b \,(\mathrm{mod}\, n)$$

Ví dụ 1

Ta có

- $11 \equiv 8 \,(\mathrm{mod} 3)$ (vì 11 và 8 chia cho 3 đều có số dư là 2)
- $-2 \equiv 5 \,(\mathrm{mod} 7)$ (vì -2 và 5 chia cho 7 đều có số dư là 5)

Một số tính chất

Tính chất

Quan hệ đồng dư là một quan hệ tương đương

- Tính phản xạ

$$a \equiv a \pmod{n}$$

- Tính đối xứng

$$a \equiv b \pmod{n} \Rightarrow b \equiv a \pmod{n}$$

- Tính bắc cầu

$$a \equiv b \pmod{n} \text{ và } b \equiv c \pmod{n} \Rightarrow a \equiv c \pmod{n}$$

Một số tính chất (cont.)

Tính chất

Nếu

$$a_1 \equiv b_1 \pmod{n}$$
$$a_2 \equiv b_2 \pmod{n}$$

Thì

- $a_1 + a_2 \equiv b_1 + b_2 \pmod{n}$
- $a_1 - a_2 \equiv b_1 - b_2 \pmod{n}$
- $a_1 a_2 \equiv b_1 b_2 \pmod{n}$
- $a_1^k \equiv b_1^k \pmod{n}$

Áp dụng

Ví dụ 2

Tìm phần dư của phép chia 3^{32} cho 17

Lời giải tính trực tiếp

- Ta có $3^{32} = 1853020188851841$
- Và $1853020188851841 \bmod 17 = 1$
- Vậy, phần dư của phép chia 3^{32} cho 17 là 1

■

Áp dụng (cont.)

Lời giải đồng dư

Ta có

$$
\begin{aligned}
3^{32} &\equiv 3^{2^{2^{2^2}}} \pmod{17} \\
&\equiv 9^{2^{2^2}} \pmod{17} \\
&\equiv 81^{2^2} \pmod{17} \equiv 15^{2^2} \pmod{17} \\
&\equiv 225^2 \pmod{17} \equiv 4^2 \pmod{17} \\
&\equiv 16^2 \pmod{17} \\
&\equiv 256 \pmod{17} \\
&\equiv 1 \pmod{17}
\end{aligned}
$$

■

Áp dụng (cont.)

Ví dụ 3

Tìm hai chữ số cuối cùng của số 7^{16}

Lời giải

Hai chữ số cuối cùng chính là phần dư của phép chia 7^{16} cho 100. Ta có

$$\begin{aligned} 7^{16} &\equiv 7^{2^{2^{2^2}}} \pmod{100} \\ &\equiv 49^{2^{2^2}} \pmod{100} \\ &\equiv 2401^{2^2} \pmod{100} \equiv 1^{2^2} \pmod{100} \\ &\equiv 1 \pmod{100} \end{aligned}$$

Vậy, hai chữ số cuối cùng là 01 ■

BẢNG BĂM

Giới thiệu

- Các thao tác tìm kiếm trên các cấu trúc dữ liệu như danh sách, cây nhị phân, ... thường dựa trên việc so sánh khóa của các phần tử của cấu trúc dữ liệu. Do đó, thời gian thao tác sẽ phụ thuộc vào kích thước của dữ liệu
- Bảng băm là một cấu trúc dữ liệu có thể giảm chi phí đến $O(1)$ (không phụ thuộc vào kích thước của dữ liệu)
- Nội dung được tham khảo từ `http://www.giaithuatlaptrinh.com/`

Các định nghĩa

Định nghĩa 2

Bảng băm (*hash table*) là một cấu trúc dữ liệu chứa các phần tử có "địa chỉ". Bảng băm thường là một mảng T có m phần tử. Tuy nhiên, nó có những biến thể khác nhau

Các định nghĩa (cont.)

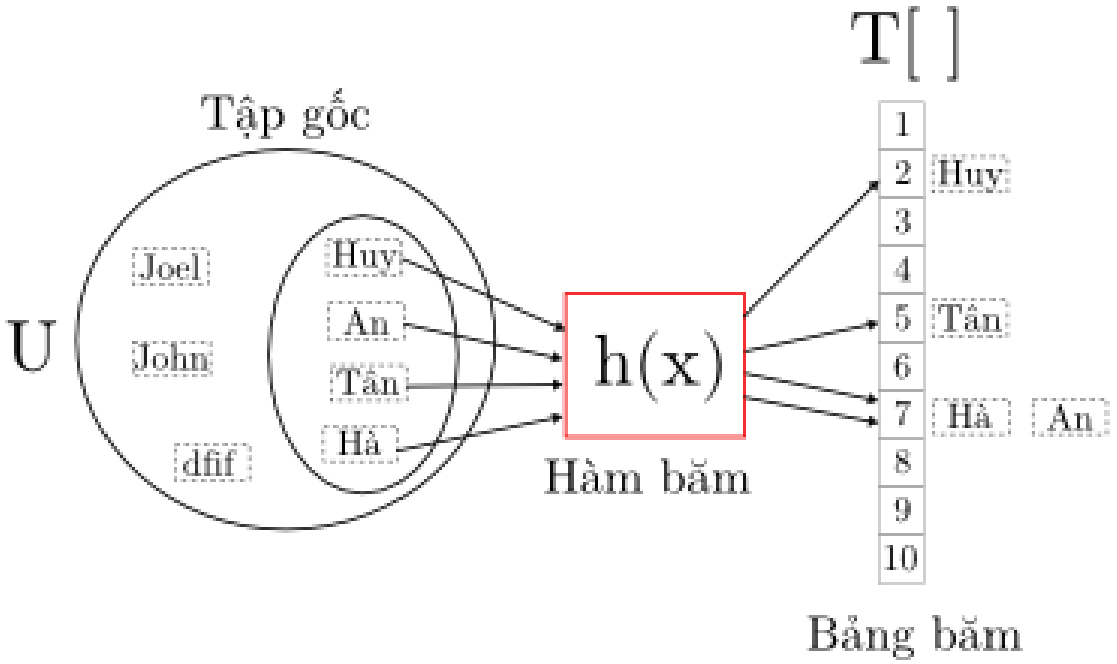
Định nghĩa 3

Hàm băm (*hash function*) là một ánh xạ **khoá** thành **địa chỉ**

$$\begin{array}{rccc} h: & \mathcal{U} & \rightarrow & \{0,1,...,m-1\} \\ & k & \mapsto & h(k) \end{array} \tag{1}$$

- $\mathcal{U}$ 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

Các định nghĩa (cont.)



Các định nghĩa (cont.)

Định nghĩa 4

Hệ số tải (load factor) α của một hàm băm h được định nghĩa như sau:

$$\alpha = \frac{|\mathcal{U}|}{m} = \frac{n}{m} \tag{2}$$

Định nghĩa 5

Cho trước hàm băm $h(x)$, hai khóa x, y mà $x \neq y$ được gọi là **đụng độ** (collision) nếu $h(x) = h(y)$. Xác suất đụng độ ký hiệu $Pr[h(x) = h(y)]$

Chuyển kiểu cho khóa

- Khóa phải ở dạng số nguyên không dấu. Ví dụ: 12345
- Mọi khóa bất kỳ đều có thể chuyển thành dạng chuỗi.
  - 12345.27 $\rightarrow$ “12345.27”
- Mọi khóa dạng chuỗi đều có thể chuyển thành dạng số nguyên
  - “AB” $\rightarrow$ 0100.0001.0100.0010 $\rightarrow$16706

Chuyển kiểu cho khóa (cont.)

Định nghĩa 6

Hàm băm chuyển kiểu $f(s)$ là ánh xạ một chuỗi s thành một số nguyên

$$\begin{array}{rccc} f: & \mathcal{S} & \rightarrow & \{0,1,...,2^n-1\} \\ & s & \mapsto & f(s) \end{array} \tag{3}$$

Ví dụ 4

Một số hàm băm

- Hàm CRC32 (check sum) sẽ trả về một số nguyên (32 bit)
- Hàm MD5 sẽ trả về một số nguyên (128 bit)
- Hàm SHA1 sẽ trả về một số nguyên (160 bit)
- Hàm SHA2 sẽ trả về một số nguyên (256 bit)

Chuyển kiểu cho khóa (cont.)

Định nghĩa 7

Băm đa thức (polynomial hashing) là một hàm băm chuyển kiểu. Gọi $s[0, 1, \ldots, m-1]$ là chuỗi ký tự và tham số b. Giá trị $f(s)$ của s được tính bởi công thức sau

$$f(s) = \left(s_0 \cdot b^{m-1} + s_1 \cdot b^{m-1} + \ldots + s_{m-1}\right) \bmod 2^n \quad (4)$$

Có thể tính hiệu quả bằng phương pháp Horner

$$f(s) = ((\ldots((s_0 \cdot b + s_1) \cdot b + s_2) \cdot b + \ldots) \cdot b + s_{m-1}) \bmod 2^n \quad (5)$$

Chuyển kiểu cho khóa (cont.)

Có rất nhiều hàm băm chuyển kiểu và chúng có thể dùng vào các mục đích khác như

- Kiểm tra tính toàn vẹn của dữ liệu
- Mã hóa

Các loại hàm băm

Một hàm băm $h(x)$ tốt phải thỏa mãn các điều kiện sau:

- Tất định
- Ít xảy ra đụng độ
- Tính toán nhanh

Các loại hàm băm (cont.)

Định nghĩa 8

Cho một số nguyên tố $p \geq |\mathcal{U}|$, ta định nghĩa hàm băm

$$h_r(x) = (xr \bmod p) \bmod m \tag{6}$$

Công thức này sẽ cho một họ hàm băm là

$$\mathcal{H} = \{h_1(x), h_2(x), \ldots, h_{p-1}(x)\}$$

.

Các loại hàm băm (cont.)

Định nghĩa 9

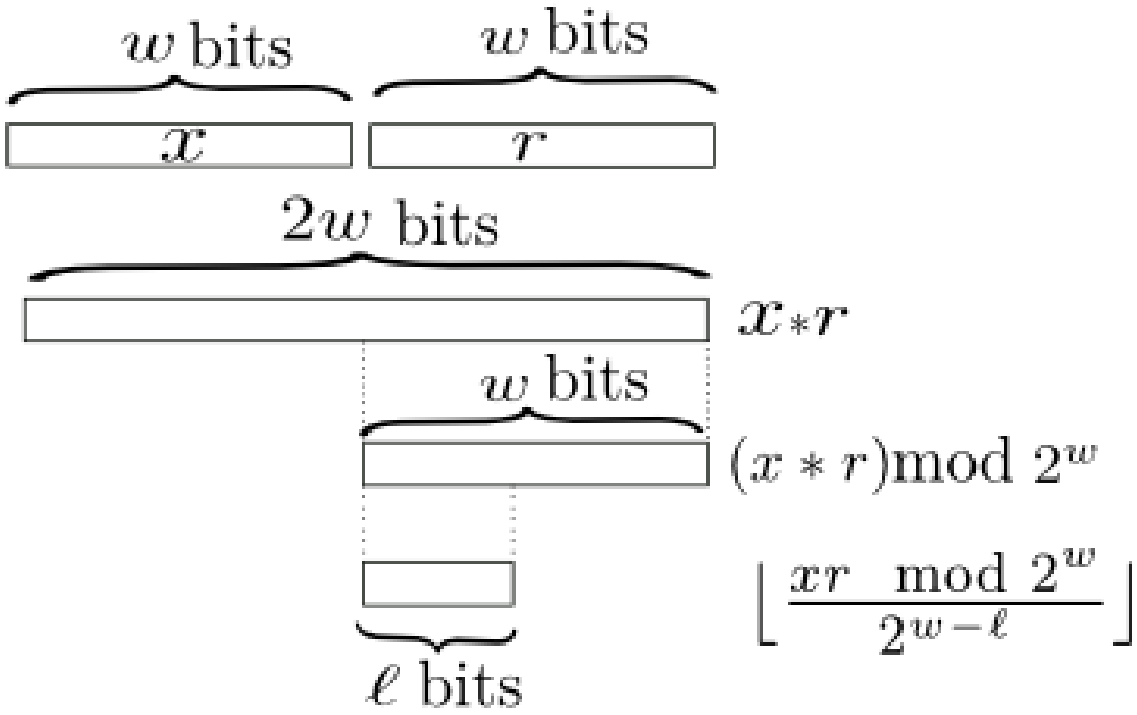
Gọi w là số nguyên dương nhỏ nhất sao cho $|\mathcal{U}| \leq 2^w$. Chọn $m = 2^\ell$. Với mỗi số nguyên dương **lẻ** r không lớn quá 2^w, ta định nghĩa hàm băm

$$g_r(x) = \left\lfloor \frac{(rx) \bmod 2^w}{2^{w-\ell}} \right\rfloor \tag{7}$$

Công thức này sẽ cho một họ hàm băm là

$$\mathcal{G} = \{g_1(x), g_3(x), \ldots, g_{2^w-1}(x)\}$$

Các loại hàm băm (cont.)



Các loại hàm băm (cont.)

Định lý 1

Nếu chọn ngẫu nhiên một hàm băm $h_r(x)$ (hoặc $g_r(x)$) từ $\mathcal{H}$ (hoặc từ $\mathcal{G}$) để thực hiện băm thì xác suất đụng độ sẽ là cỡ $\frac{1}{m}$.

Chứng minh

Sinh viên tham khảo tài liệu ■

Các loại hàm băm (cont.)

Định nghĩa 10

Knuth đề xuất một hàm băm

$$h(k) = \lfloor m \cdot (k \cdot A \bmod 1) \rfloor \tag{8}$$

- Phép toán $x \bmod 1$ là phép lấy phần thập phân của số thực x
- Phép toán $\lfloor x \rfloor$ là phép toán lấy phần nguyên của số thực x
- k là khóa, m là kích thước bảng, A là hằng số và $0 < A < 1$
  - Người ta thường chọn $m = 2^p$
  - Giá trị A thường được chọn

$$A = \frac{\sqrt{5} - 1}{2} \approx 0.618033988749 89$$

Các loại hàm băm (cont.)

- Ví dụ $k = 12345$ và $m = 2^{10}$
- Từ công thức

$$h(k) = \lfloor 1024 \times (12345 \times 0.618033988874989 \bmod 1) \rfloor$$

- Cuối cùng

$$h(k) = h(12345) = 644$$

Các thao tác trên bảng băm

- Khởi tạo bảng băm
- Tìm kiếm một phần tử trên bảng
- Thêm một phần tử vào bảng
- Loại bỏ một phần tử khỏi bảng

Xử lý đụng độ

- Phương pháp nối kết (separate chaining)
- Phương pháp địa chỉ mở (open addressing)
- Phương pháp băm hoàn hảo (perfect hashing)

Phương pháp kết nối

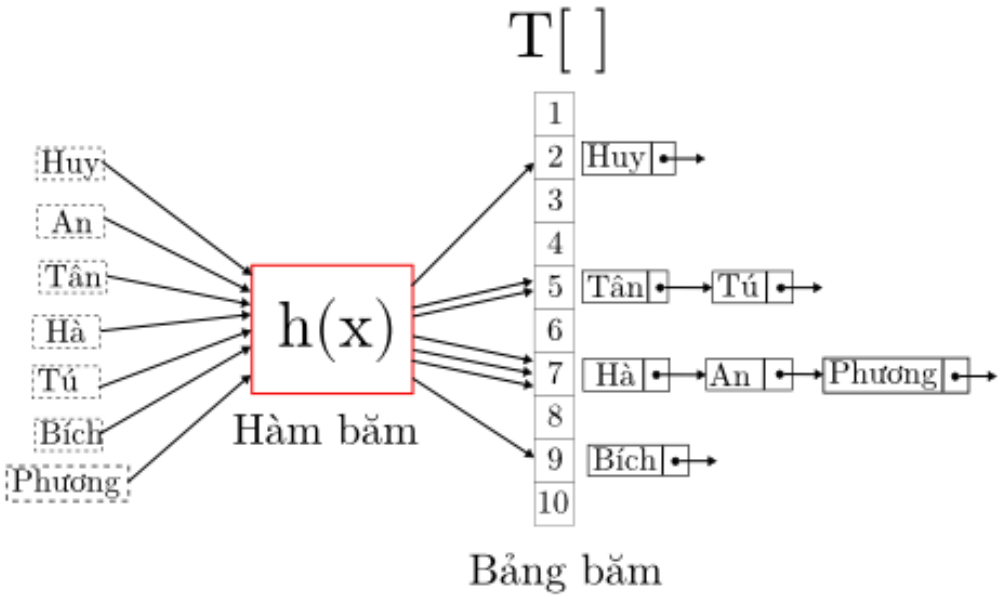
- Ý tưởng của phương pháp là đưa tất cả các khóa đụng độ vào chung một địa chỉ và tạo thành một danh sách liên kết
- Thêm khóa x

```
PutToChainingTable(x, h, T)
  add x to list T[h(x)]
```

- Tìm khóa x

```
LookupChainingTable(x, h, T)
  L ← T[h(x)]
  for each element e in L
    if e = x
      return Yes
  return No
```

Phương pháp kết nối (cont.)



Phương pháp kết nối (cont.)

Định lý 2

Gọi $\ell(x)$ là chiều dài của danh sách chứa khóa x

- *Chiều dài trung bình của ℓ là α*
- *Chiều dài dài nhất của ℓ là khoảng $O(\frac{\log n}{\log \log n})$*

Chứng minh

Sinh viên tự đọc tài liệu tham khảo ■

Phương pháp địa chỉ mở

- Trong phương pháp giải kết nối, có thể có khá nhiều ô của bảng rỗng trong khi một số ô khác lại chứa khá nhiều phần tử. Ngoài ra, ta cần duy trì một danh sách các con trỏ để liên kết các phần tử lại với nhau. Các liên kết này đương nhiên là sẽ tốn thêm bộ nhớ.
- Tên gọi "địa chỉ mở" mang ý nghĩa là "địa chỉ" của khóa không phải chỉ được xác định bằng "duy nhất" hash code của phần tử đó, mà còn có sự can thiệp của phép "dò tìm"
- Các phần tử chỉ lưu trong bảng băm, không dùng thêm bộ nhớ mở rộng như phương pháp kết nối

Phương pháp địa chỉ mở (cont.)

Ý tưởng chung

Sử dụng m hàm băm độc lập $h_0, h_1, \ldots, h_{m-1}$, sao cho, với bất kì phần tử x nào, m giá trị $h_0(x), h_1(x), \ldots, h_{m-1}(x)$ đôi một khác nhau.

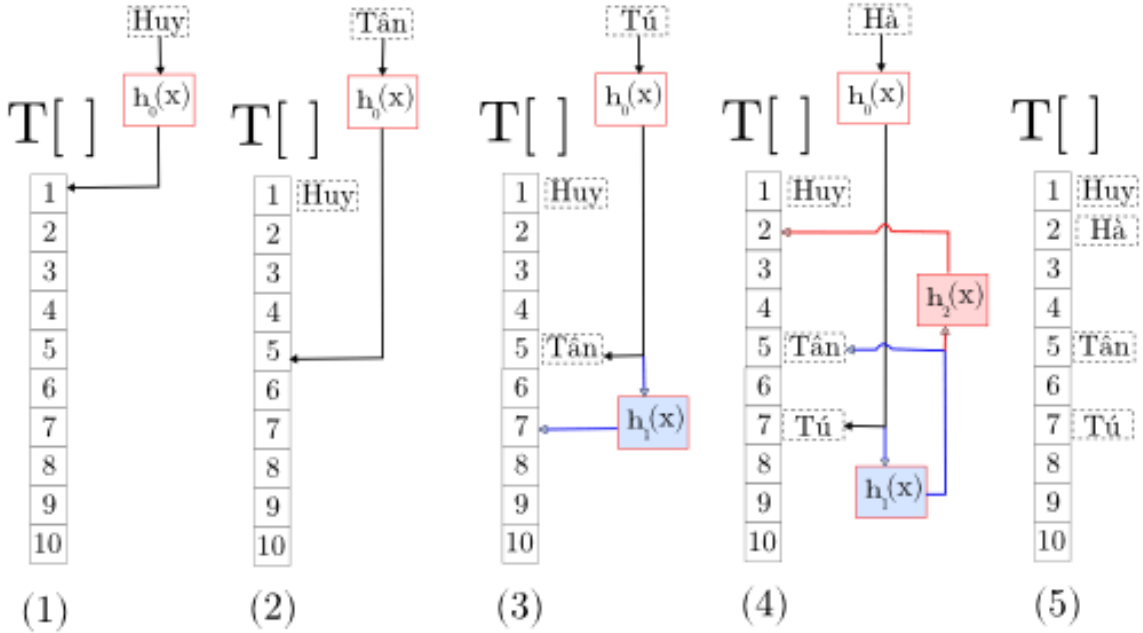
- Nếu tìm thấy một ô trống đầu tiên thì lưu x vào đó
- Nếu không thấy sử dụng hàm băm kế tiếp
- Do $h_0(x), h_1(x), \ldots, h_{m-1}(x)$ là một hoán vị của $\{0, 1, \ldots, m-1\}$, quá trình tìm kiếm ô trống luôn kết thúc sau tối đa m bước.

Phương pháp địa chỉ mở (cont.)

- Thêm một khóa x

```
PutToOpenAddressingTable(x, {h_0, ..., h_{m-1}}, T)
  i ← 0
  while T[h_i(x)] ≠ Null
    i ← i + 1
    T[h_i(x)] ← x
```

Phương pháp địa chỉ mở (cont.)



Phương pháp địa chỉ mở (cont.)

- Tìm một khóa x

```
LookupOpenAddressingTable(x, {h_0, ..., h_{m-1}}, T)
  i ← 0
  while T[h_i(x)] ≠ x
    if T[h_i(x)] = Null
      return No
    i ← i + 1
  if i ≤ m − 1
    return h_i(x)
  return No
```

Phương pháp địa chỉ mở (cont.)

Định lý 3

- *Số lần dò tìm trung bình để tìm kiếm một khóa x* ***không*** *có trong bảng là*

$$\frac{1}{1-\alpha} \tag{9}$$

- *Số lần dò trung bình để tìm kiếm một khóa x* ***có*** *trong bảng là*

$$\frac{1}{\alpha}\left(1+\ln\frac{1}{1-\alpha}\right) \tag{10}$$

- *Trong trường hợp xấu nhất số lần dò tìm là* $O(\log n)$

Chứng minh

Sinh viên tự chứng minh ■

Phương pháp địa chỉ mở (cont.)

Thực hiện

Trong thực tế, việc thiết kế m hàm băm ngẫu nhiên **độc lập** thỏa mãn mã băm đôi một khác nhau với một khóa cho trước là việc vô cùng khó. Cho dù ta có thực hiện được thì chi phí thời gian có lẽ cũng không nhỏ. Do đó, trong thực tế, ta chấp nhận các hàm băm "phụ thuộc" với nhau ở một mức độ nào đó, mỗi mức độ cho chúng ta một phép dò khác nhau: dò tuyến tính, dò nhị phân, dò bậc hai và băm kép.

Phương pháp địa chỉ mở (cont.)

Định nghĩa 11

Dò tuyến tính (linear probing), ta sẽ chỉ sử dụng **một hàm băm tốt** $h(x)$ để định nghĩa m hàm băm như sau

$$h_i(x) = (h(x) + i) \bmod m \qquad 0 \leq i \leq m-1 \tag{11}$$

- Điểm mạnh của phương pháp dò tuyến tính này là thực thi đơn giản.
- Tuy nhiên, các giá trị băm sẽ có xu hướng tụm lại với nhau thành một dãy con liên tục của T. Ngoài ra, khi hệ số tải gần bằng 1 thì tìm kiếm với dò tuyến tính cực kì kém hiệu quả.

Phương pháp địa chỉ mở (cont.)

Định nghĩa 12

Dò nhị phân, ta chọn $m = 2^\ell$ và sử dụng **một hàm băm tốt** $h(x)$ để định nghĩa m hàm băm như sau

$$h_i(x) = (h(x) \oplus i) \qquad 0 \leq i \leq m-1 \tag{12}$$

Phương pháp địa chỉ mở (cont.)

Định nghĩa 13

Dò bậc hai (quadratic probing), ta sẽ dùng **một hàm băm tốt** $h(x)$ và một hàm bậc 2 để thiết kế m hàm băm như sau

$$h_i(x) = (h(x) + i^2) \bmod m \qquad 0 \leq i \leq m-1 \tag{13}$$

- Phương pháp dò bậc hai về mặt lý thuyết và thực nghiệm tốt hơn dò tuyến tính

Phương pháp địa chỉ mở (cont.)

Định nghĩa 14

Dò băm kép (double hashing) sử dụng **hai hàm băm tốt độc lập** $h(x), g(x)$ để thiết kế m hàm băm như sau

$$h_i(x) = (h(x) + ig(x)) \bmod m \qquad 0 \leq i \leq m-1 \qquad (14)$$

- Phương pháp dò băm kép tốt hơn về mặt lý thuyết
- Tuy nhiên, trong thực tế, phương pháp này sẽ hơi chậm hơn.

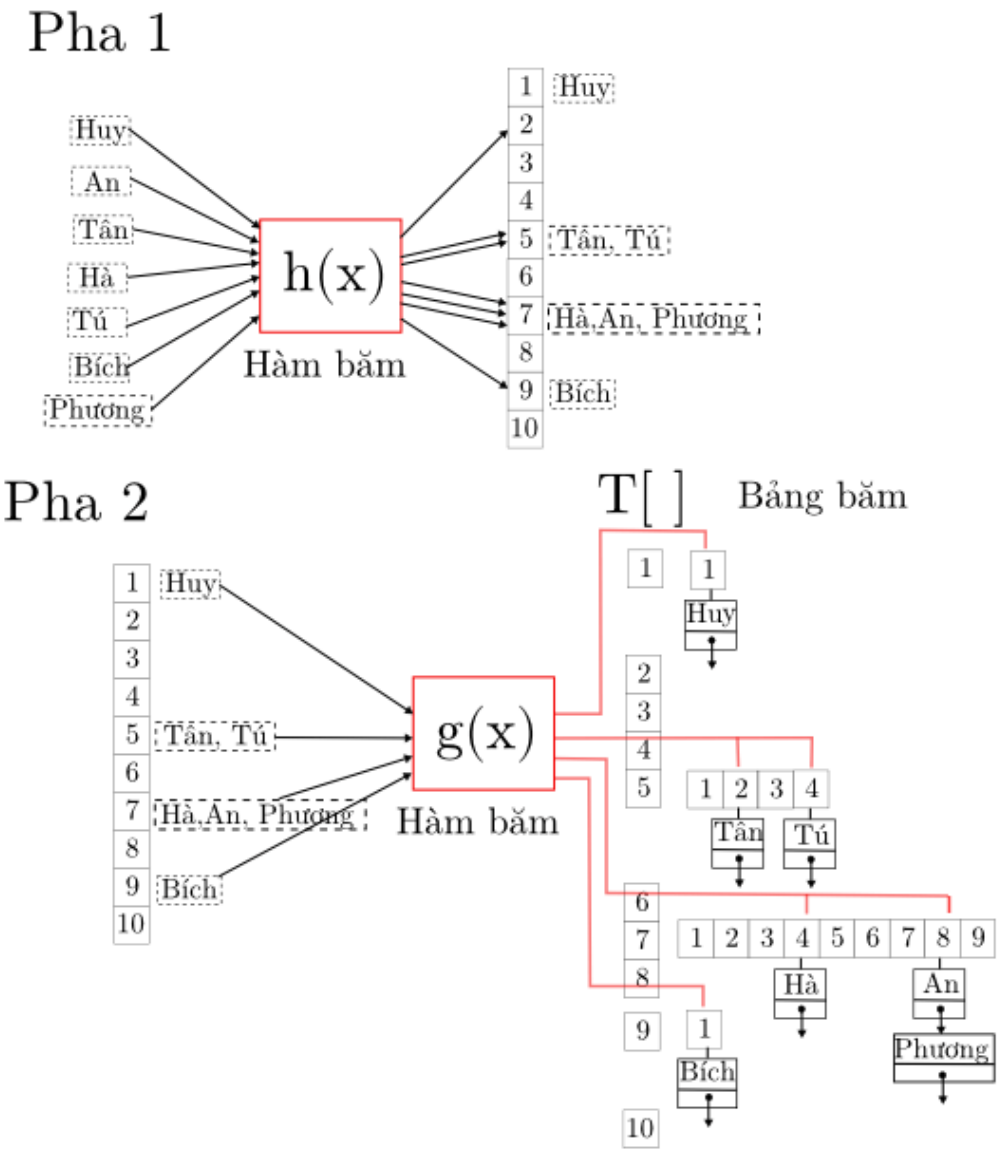
Phương pháp băm hoàn hảo

- Băm hoàn hảo sẽ sử dụng **hai hàm băm tốt** $\{h(x), g(x)\}$ và bảng băm hai chiều $T[1, 2, \ldots, m][\ldots]$.
- Mỗi hàng của bảng băm $T[i]$ sẽ được xem như một bảng băm phụ, có kích thước phụ thuộc vào đầu vào.
- Khi băm vào bảng, ta thực hiện băm theo 2 pha:
  - Trong pha đầu tiên, sử dụng h để băm x vào **hàng** $h(x)$ của bảng T.
  - Trong pha thứ 2, gọi $C[i]$ là số lượng phần tử được băm cùng vào hàng thứ i sau pha đầu tiên, với mỗi hàng i, ta cấp phát một bộ nhớ $C[i]^2$ cho hàng $T[i]$.
  - Sau đó, ta coi hàng này như một bảng băm và dùng g để băm các phần tử x có cùng mã băm i vào ô $g(x)$ của hàng này.

Phương pháp băm hoàn hảo (cont.)

- Đụng độ lần 2 này sẽ được giải quyết bằng phương pháp kết nối.

Phương pháp băm hoàn hảo (cont.)



Phương pháp băm hoàn hảo (cont.)

- Thêm mảng A vào bảng T

```
PutToPerfectTable(A[1, 2, ..., n], {h, g}, T)
  C[0, ..., m − 1] ← {0, ..., 0}
  for i ← 1 to n
    C[h(A[i])] ← C[h(A[i])] + 1
  for i ← 0 to m − 1
    allocate C[i]^2 memory slots to row T[i] of 2D table
  for i ← 1 to n
    j ← h(A[i])
    k ← g(A[i]) mod C^2[j]
    add A[i] to list T[j][k]
```

Phương pháp băm hoàn hảo (cont.)

```
LookupPerfectTable(x, {h, g}, C[0, ..., m − 1], T)
  i ← h(x)
  j ← g(x) mod C²[i]
  L ← T[i][j]
  if L = Null
    return No
  for each element e in L
    if x = e
      return Yes
  return No
```

Đánh giá

Load factor α	**0.1**	**0.5**	**0.8**	**0.9**	**0.99**	**2**
Kết nối	1.05	1.25	1.4	1.45	1.5	2
Mở, ngẫu nhiên	1.05	1.4	2	2.6	4.6	-
Mở, tuyến tính	1.06	1.5	3	5.5	50.5	-

Bảng 1: Số lần dò tìm trung bình **có** (lý thuyết)

Đánh giá (cont.)

Load factor α	**0.1**	**0.5**	**0.8**	**0.9**	**0.99**	**2**
Kết nối	0.1	0.5	0.8	0.9	0.99	2
Mở, ngẫu nhiên	1.1	2	5	10	100	-
Mở, tuyến tính	1.12	2.5	13	50	5000	-

Bảng 2: Số lần dò tìm trung bình **không có** (lý thuyết)

Đánh giá (cont.)

Load factor α	**0.1**	**0.5**	**0.8**	**0.9**	**0.99**	**2**
Kết nối	1.04	1.2	1.4	1.4	1.5	2
Mở, ngẫu nhiên	1.04	1.5	2.1	2.7	5.2	-
Mở, tuyến tính	1.05	1.6	3.4	6.2	21.3	-

Bảng 3: Số lần dò tìm trung bình **có** (thực nghiệm)

Đánh giá (cont.)

Load factor α	**0.1**	**0.5**	**0.8**	**0.9**	**0.99**	**2**
Kết nối	0.1	0.5	0.8	0.9	0.99	2
Mở, ngẫu nhiên	1.13	2.2	5.2	11.9	126	-
Mở, tuyến tính	1.13	2.7	15.4	59.8	430	-

Bảng 4: Số lần dò tìm trung bình **không có** (thực nghiệm)

Thiết kế hàm băm

- Trong các ứng dụng mà chúng ta phải thường xuyên thêm và xóa phần tử khỏi bảng, phương pháp chuỗi kết nối sẽ là một lựa chọn tốt.
- Ngược lại, trong các ứng dụng mà chúng ta chủ yếu thực hiện tìm kiếm, ít khi phải thêm hay xóa phần tử khỏi bảng (ví dụ ứng dụng từ điển chẳng hạn) thì băm hoàn hảo sẽ là một lựa chọn tốt.
- Tương tự như băm hoàn hảo, nếu ứng dụng của chúng ta chủ yếu thực hiện tìm kiếm nhưng chúng ta lại có thêm thông tin về tần suất truy nhập khóa, thì ta có thể sử dụng băm địa chỉ mở.