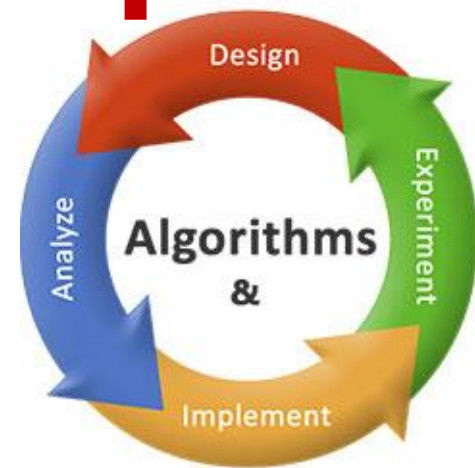
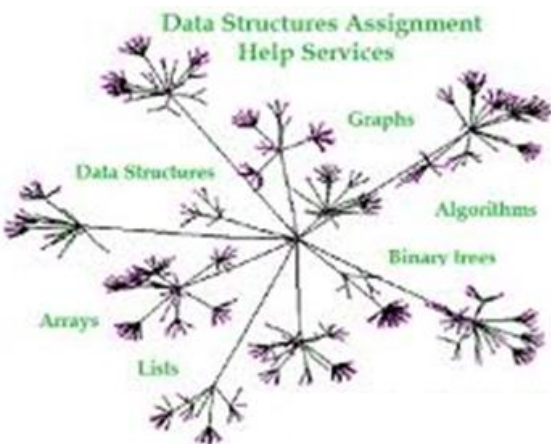


CẤU TRÚC DỮ LIỆU & GIẢI THUẬT



Datastructures



Lê Văn Hạnh

levanhanhvn@gmail.com

NỘI DUNG MÔN HỌC

- Chương 1: Ôn tập ngôn ngữ lập trình C
- Chương 2: Kiểu dữ liệu con trỏ
- Chương 3: Tổng quan về cấu trúc dữ liệu và giải thuật
- Chương 4: Danh sách kê (Danh sách tuyến tính)
- Chương 5: Các giải thuật tìm kiếm trên danh sách kê
- Chương 6: Các giải thuật sắp xếp trên danh sách kê
- Chương 7: Danh sách liên kết động (Linked List)
- Chương 8: Ngăn xếp (*Stack*)
- Chương 9: Hàng đợi (*Queue*)
- Chương 10: Cây nhị phân tìm kiếm (*Binary Search Tree*)
- Chương 11: Cây cân bằng (*Balanced binary search tree – AVL tree*)
- **Chương 12: Bảng băm (*Hash Table*)**

Chương 12

BẢNG BĂM (Hash Table)

MỤC TIÊU

Sau khi học xong bài này, học viên có thể:

- Hiểu được cấu trúc bảng băm (Hash table)
- Các tác vụ trên bảng băm.
- Vận dụng cấu trúc bảng băm để giải các bài toán cụ thể.

NỘI DUNG

1. Giới thiệu
2. Hàm băm (hash function)
3. Các tác vụ trên bảng băm
4. Phân loại bảng băm
5. Bảng băm với phương pháp kết nối trực tiếp (*Direct chaining Method*)
6. Bảng băm với phương pháp kết nối hợp nhất (*Coalesced chaining Method*)
7. Bảng băm với phương pháp dò tuyến tính (*Linear Probing Method*)
8. Bảng băm với phương pháp dò bậc hai
9. Bảng băm với phương pháp băm kép
10. Một số bảng băm theo phân loại cấu trúc

1. GIỚI THIỆU

- Trong thực tế, các phần tử cần lưu trữ thường được tổ chức dưới dạng kiểu cấu trúc, ví dụ:

```
typedef struct NODE
{
    int key;
    char value[30];
}
```

Trong đó,

- Kiểu dữ liệu của *key* có thể ở dạng số hay dạng chuỗi.
- Kiểu dữ liệu của *value* có thể ở là 1 trong các kiểu dữ liệu chuẩn của C hoặc có thể là kiểu struct khác (đã được khai báo trước đó).

1. GIỚI THIỆU

- Các phép toán dựa trên các cấu trúc như cây, danh sách, ... chủ yếu được thực hiện thông qua việc so sánh các phần tử có cấu trúc.
⇒ thời gian thực thi nhanh hay chậm phụ thuộc vào kích thước các phần tử này.
- Để hạn chế số lần so sánh, người ta đưa ra khái niệm bảng băm (Hash table).
- Nếu được tổ chức tốt, các phép toán trên bảng băm có độ phức tạp là $O(1)$ và không phụ thuộc vào kích thước bảng.

1. Giới thiệu

1.1. Cấu trúc bảng băm

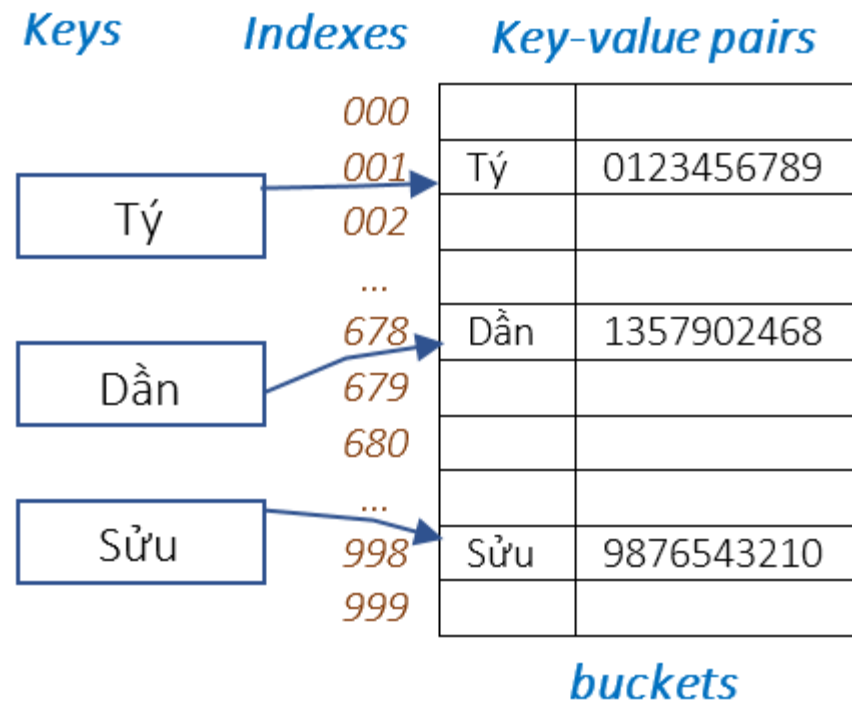
- Băm (*hashing*) là một kỹ thuật được sử dụng để xác định duy nhất một đối tượng cụ thể từ một nhóm các đối tượng tương tự.
- Một số ví dụ về băm:
 - Trong các trường đại học, mỗi sinh viên được chỉ định một mã số duy nhất có thể được sử dụng để lấy thông tin khác về họ.
 - Trong các thư viện, mỗi cuốn sách được gán một mã sách duy nhất có thể được sử dụng để xác định thông tin về cuốn sách, chẳng hạn như vị trí chính xác của cuốn sách trong thư viện hoặc thông tin về người đang mượn cuốn sách đó, ...

1. Giới thiệu

1.1. Cấu trúc bảng băm

- Bảng băm:

- Là một kiểu dữ liệu trừu tượng, lưu trữ một cặp dữ liệu *Key/Value* và cho phép tìm Key một cách nhanh chóng.



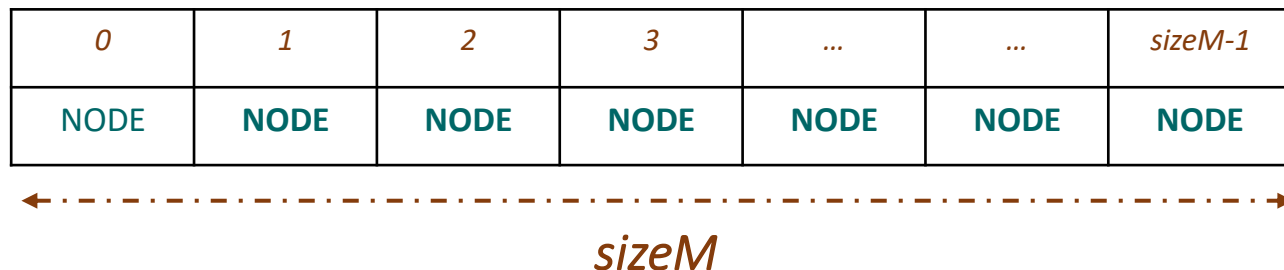
Minh họa cấu trúc dữ liệu của bảng băm

1. Giới thiệu

1.1. Cấu trúc dữ liệu dùng cho bảng băm

- Hầu hết được cài đặt bằng danh sách kê có M phần tử, lưu trữ các cấu trúc *NODE* và cho phép tìm *key* một cách nhanh chóng.
- Trong đó, M (hoặc được đặt tên khác là *sizeM*) thường được chọn là 1 số nguyên tố hoặc là số lũy thừa của 2 (2^p) vì giúp giảm các đụng độ về vị trí lưu trữ giữa các phần tử.
- Khi khởi động bảng băm, tất cả trường *key* được gán trị *NULLKEY* (hoặc -1).

```
#define sizeM 11
#define NULLKEY -1
typedef struct NODE
{
    int key;
    char value[30];
};
NODE HT[sizeM];
```



1. Giới thiệu

1.1. Cấu trúc dữ liệu dùng cho bảng băm

- Minh họa:

0	15
1	NULL
2	27
...	...
...	NULL
17	89
18	NULL

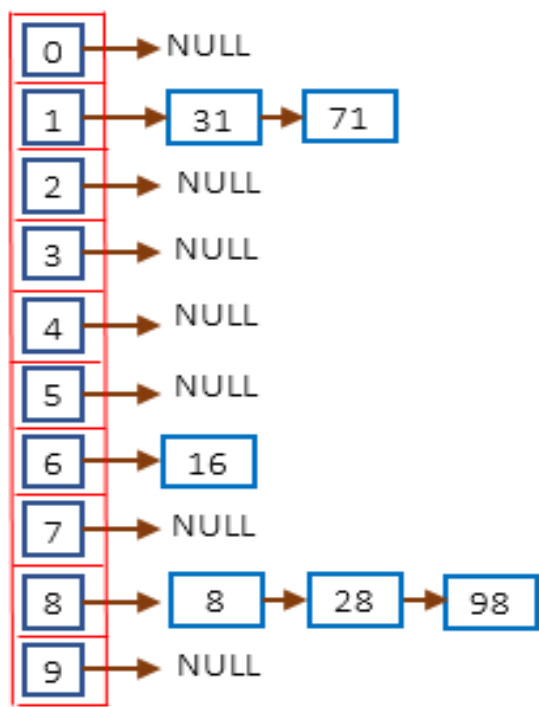
Bảng băm
với kiểu dữ
liệu là int

0	NULL
1	Alice
2	Brown
...	...
...	NULL
24	Paul
25	Bob

Bảng băm
với kiểu dữ
liệu là string

0	NULL			
1	0987654301	Patrick
2	1234567802	John
...
...
98	9876543298	White
99	NULL			

Bảng băm
với kiểu dữ liệu là kiểu struct



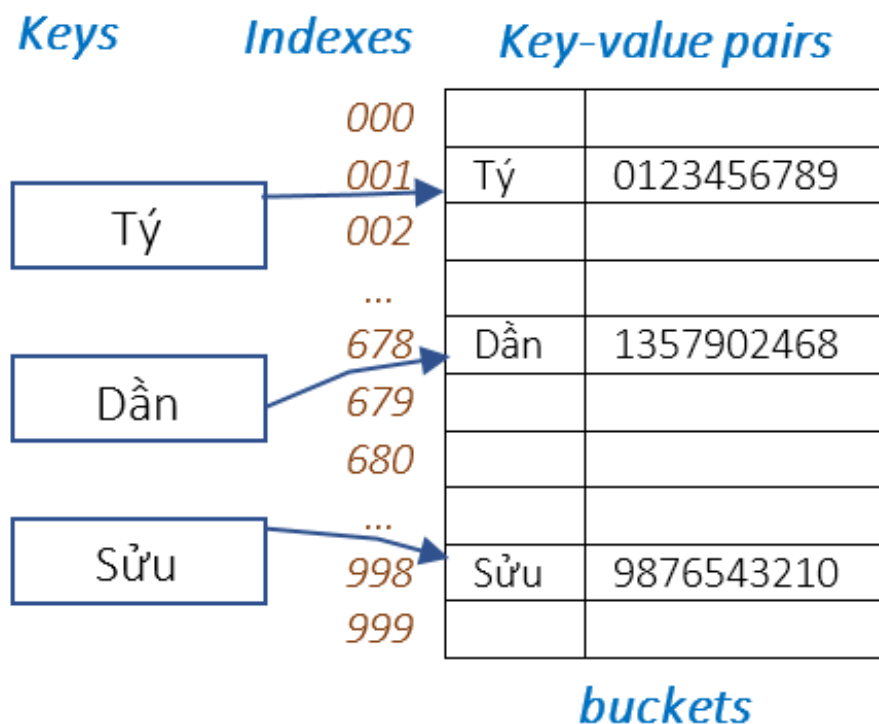
Kiểu dữ liệu của bảng băm không phải kiểu con trỏ

Kiểu dữ liệu của bảng băm
là kiểu con trỏ

<pre>#define sizeM 11 #define NULLKEY -1 int HT[sizeM];</pre> <p>Bảng băm kiểu số nguyên</p>	<pre>#define sizeM 11 #define NULLKEY -1 char* HT[sizeM];</pre> <p>Bảng băm kiểu chuỗi</p>	<pre>#define sizeM 11 #define NULLKEY -1 typedef struct NODE { int key; char value[30]; }; NODE HT[sizeM];</pre> <p>Bảng băm kiểu cấu trúc</p>	<pre>#define sizeM 11 #define NULLKEY -1 typedef struct NODE { int key; char value[30]; }; NODE *HT[sizeM];</pre> <p>Bảng băm kiểu con trỏ</p>
---	---	--	--

1.2. Hàm băm (hash function)

- Ý tưởng của băm (hashing) là phân phối các mục (cặp khóa-giá trị) thống nhất trên một mảng. Mỗi phần tử được gán một khóa (hoặc khóa được chuyển đổi). Bằng cách sử dụng khóa đó, ta có thể truy cập phần tử trong thời gian $O(1)$.
- Khi thực hiện băm, hàm băm giúp ánh xạ một giá trị khóa vào một vị trí trong bảng (tức là chuyển đổi key thành index).

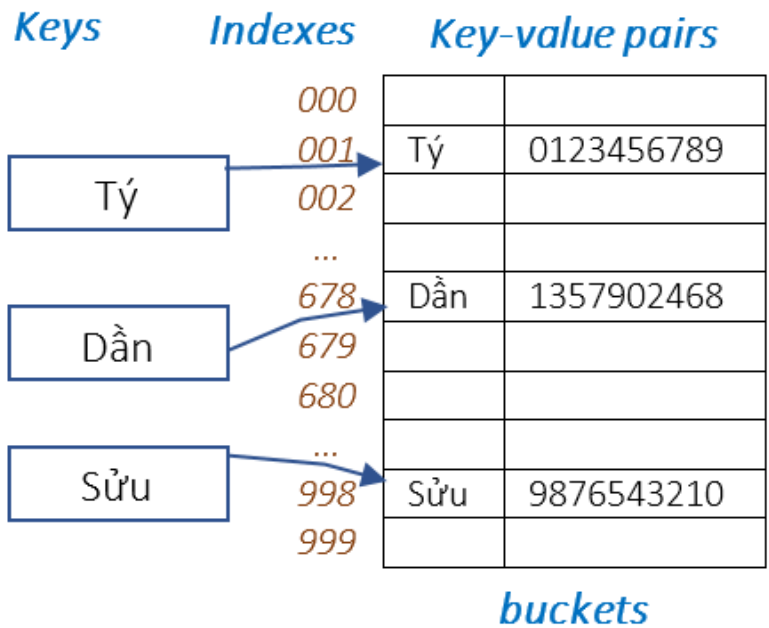


1. Giới thiệu

1.2. Hàm băm (hash function)

- Các mục (cặp khóa/giá trị) được lưu trữ trong một cấu trúc dữ liệu gọi là bảng băm (A). Số lượng indexes có trong bảng băm là M (được đánh số từ 0 đến M-1).

Trong đó, M thường được chọn là 1 số nguyên tố hoặc là số lũy thừa của 2 (2^p) vì giúp giảm các đụng độ về vị trí lưu trữ giữa các phần tử.



1.3. Hệ số tải (load factor)

- Gọi:
 - n là số lượng phần tử cần lưu trong Hash table.
 - M là kích thước của Hash table.
- Giá trị n/M được gọi là *load factor*.
- Khi *load factor* nhỏ (xấp xỉ 1), và giá trị của hàm Hash phân bố đều, độ phức tạp của các thao tác trên Hash table là $O(1)$.

1. Giới thiệu

1.4. Các vấn đề cần xác định khi tổ chức dữ liệu dưới dạng cấu trúc bảng băm

Khi tổ chức dữ liệu dưới dạng cấu trúc dữ liệu bảng băm, phải xác định được 3 vấn đề:

- i. Tập khoá K (set of keys).* Giá trị gì sẽ được chọn làm khóa (*key*) để khả năng bị trùng giữa các khóa là thấp nhất có thể. Vì khi các khóa hoàn toàn không bị trùng nhau sẽ giúp tốc độ truy xuất đạt là $O(1)$.
- ii. Tập địa chỉ M (set of addresses)* có trong bảng băm A . Với M là kích thước của bảng băm.
- iii. Hàm băm $hf(k)$:* dùng để ánh xạ một khoá k từ tập các khoá k thành một địa chỉ tương ứng trong tập M .

1. Giới thiệu

1.5. Ưu điểm của bảng băm

- Bảng băm là 1 cấu trúc dung hoà tốt giữa thời gian truy xuất và dung lượng bộ nhớ:
 - Nếu không có sự giới hạn về bộ nhớ thì có thể xây dựng bảng băm với mỗi khoá ứng với một địa chỉ với mong muốn thời gian truy xuất tức thời.
 - Nếu dung lượng bộ nhớ có giới hạn thì tổ chức một số khoá có cùng địa chỉ. Lúc này thời gian truy xuất có bị giảm đi.
- Bảng băm được ứng dụng nhiều trong thực tế, rất thích hợp khi tổ chức dữ liệu có kích thước lớn và được lưu trữ ở bộ nhớ ngoài.

NỘI DUNG

1. Giới thiệu
2. Hàm băm (hash function)
3. Các tác vụ trên bảng băm
4. Phân loại bảng băm
5. Bảng băm với phương pháp kết nối trực tiếp
6. Bảng băm với phương pháp kết nối hợp nhất
7. Bảng băm với phương pháp dò tuyến tính
8. Bảng băm với phương pháp dò bậc hai
9. Bảng băm với phương pháp băm kép
10. Một số bảng băm theo phân loại cấu trúc

2. HÀM BĂM (hash function)

2.1. Giới thiệu

- Bảng băm luôn có 1 hàm băm đi kèm.
- Hàm băm là hàm được sử dụng để ánh xạ tập các khoá đại diện cho các mục dữ liệu trong bảng thành địa chỉ trên bảng băm (A) nơi chứa mục dữ liệu đó.
- Khoá trong bảng băm có thể là dạng số hoặc chuỗi.
- Hàm băm là bất kỳ hàm nào có thể được sử dụng để ánh xạ tập dữ liệu có kích thước tùy ý thành tập dữ liệu có kích thước cố định được chứa vào bảng băm.

2. Hàm băm (Hash function)

2.2. Hoạt động của hàm băm

- Thực hiện ánh xạ các khóa **key** thành địa chỉ (**index**) nơi chứa mục dữ liệu (**NODE**) đó trên mảng. Bằng cách sử dụng khóa đó, ta có thể truy cập phần tử trong thời gian $O(1)$.
- Tìm cách biến đổi khóa **key** của **NODE** thành 1 giá trị số nguyên thuộc khoảng từ 0 đến $M-1$. Sau đó sử dụng các hàm băm chuẩn trên số nguyên để xác định vị trí của **NODE** trên bảng băm.
- Các bước thực hiện băm
 - **B1**: sử dụng hàm băm để biến đổi khóa **key** của **NODE** thành 1 giá trị số nguyên thuộc khoảng từ 0 đến $M-1$. Số nguyên này sẽ được sử dụng như một chỉ mục để lưu trữ NODE.
 - **B2**: Lưu trữ phần tử trong bảng băm nơi phần tử có thể được truy xuất nhanh bằng khóa băm.

2.3. Một số tiêu chuẩn đánh giá hàm băm

- ***Dễ tính toán***: sử dụng phép tính bình thường chứ không phải là một thuật toán.
- ***Giá trị băm phải độc lập với bất cứ phần nào của dữ liệu*** nghĩa là phải phù hợp và có tính ngẫu nhiên.
- ***Phân phối đồng đều các khóa trong bảng***: các phần tử được phân phối đồng đều trên bảng băm và không dẫn đến phân cụm (có đoạn quá nhiều phần tử, lại có những đoạn để trống).

2.3. Một số tiêu chuẩn đánh giá hàm băm

- *Ít xảy ra đụng độ* (less collision): đụng độ xảy ra khi các cặp phần tử có khóa khác nhau nhưng được ánh xạ tới cùng một giá trị băm.

Gọi $P(k)$ là xác suất khoá k xuất hiện trong bảng. Khi đó với mỗi $i = 0, 1, \dots, m - 1$ thì ta có:

$$\sum_{\forall k \in H(k)=i} P(k) = \frac{1}{m}$$

2.4. Phân loại Hàm băm

- *Hàm băm dạng bảng tra*

Ví dụ

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z

- *Hàm băm dạng công thức:*

- Dạng tổng quát là **hf(key)**. Ví dụ: công thức $hf(key) = key \% M$ với M là độ lớn của bảng băm.
- Hàm băm dạng này rất đa dạng và không bị ràng buộc bởi bất cứ tiêu chuẩn nào.

2.5. Một số hàm băm đơn giản và phổ biến

2.5.1. Hàm băm sử dụng phương pháp chia

- Đặc điểm:
 - Một khoá được ánh xạ vào một trong m ô của bảng thông qua hàm:

$$HF(k) = k \bmod m$$

Trong đó: k là khoá.
 m là kích thước bảng.

- Chỉ sử dụng phép chia đơn do đó tốc độ tính toán nhanh.
- Ví dụ: lần lượt đưa các giá trị 15, 10, 26 vào bảng băm với $M=10$

$$\begin{aligned} hF(15) &= 15 \bmod M = 5 \\ hF(10) &= 10 \bmod M = 0 \\ hF(16) &= 16 \bmod M = 6 \end{aligned}$$

0	10
1	
2	
3	
4	
5	15
6	26
7	
8	
9	

2. Hàm băm (Hash function)

2.5. Một số hàm băm đơn giản và phổ biến

2.5.1. Hàm băm sử dụng phương pháp chia

- Đặc điểm:

- Cần chọn một giá trị cho m phù hợp.

- m chọn không tốt khi là một trong các giá trị sau:
 - $m = 2^P$: khi đó $h(k)$ sẽ chọn cùng giá trị là p bit cuối của k .
 - $m = 10^P$: khi đó hàm băm không phụ thuộc vào tất cả các số thập phân của khoá.
 - $m = 2^P - 1$: Nếu khoá là một xâu ký tự được dịch thành các giá trị là lũy thừa của 2, thì hai xâu có thể được băm thành cùng một giá trị địa chỉ trên bảng.
- m là tốt khi giá trị là một số nguyên tố và không quá gần với giá trị là lũy thừa của 2.

2. Hàm băm (Hash function)

2.5. Một số hàm băm đơn giản và phổ biến

2.5.1. Hàm băm sử dụng phương pháp chia

- Ví dụ về cài đặt một hàm băm sử dụng phép chia :

```
int Hash(int Key, int HashTableSize)
{
    return (Key % HashTableSize);
}
```

2. Hàm băm (Hash function)

2.5. Một số hàm băm đơn giản và phổ biến

2.5.2. Hàm băm sử dụng phương pháp nhân

- Thực hiện qua 2 bước:
 - Khoá **k** được nhân với hằng số **A** nằm trong khoảng $0 < A < 1$. Sau đó người ta sẽ sử dụng phần phân số của $k \cdot A$.
 - Phần phân số nói trên được nhân với **m** sau đó lấy phần nguyên. Do đó hàm băm có dạng: $HF(k) = m * (k \cdot A \bmod 1)$

Trong đó: **k** là khoá,
 m là kích thước bảng,
 A là hằng số.

- Ví dụ: lần lượt đưa các giá trị 5, 1, 6 vào bảng băm với $m=10, A=0.7$

$hF(5) = 10 * (5 \cdot 0.7 \bmod 1) = 3$
 $hF(1) = 10 * (1 \cdot 0.7 \bmod 1) = 0$
 $hF(6) = 10 * (6 \cdot 0.7 \bmod 1) = 4$

0	1
1	
2	
3	5
4	6
5	
6	
7	
8	
9	

2. Hàm băm (Hash function)

2.5. Một số hàm băm đơn giản và phổ biến

2.5.2. Hàm băm sử dụng phương pháp nhân

- Đặc điểm:

- Một hàm băm sử dụng phép nhân muốn có hiệu quả cao phải lựa chọn giá trị **m** và **A** cho phù hợp.

□ **m** thường được chọn là $m = 2^p$

□ **A** được chọn phụ thuộc vào đặc trưng của dữ liệu. Một giá trị **A** tốt được đề xuất có giá trị là:

$$A = 1/((1 + \sqrt{5})/2) = (\sqrt{5} - 1)/2 \approx 0.6180339887...$$

2. HÀM BĂM (hash function)

2.7. Một số ví dụ

i. Ví dụ 1: Cần tổ chức quản lý các số điện thoại gồm 10 chữ số có dạng *XXXX-XXXXXX* và tên người sở hữu số điện thoại đó:

- Khóa *k*: Lấy số nguyên được chuyển đổi từ hàm băm là khóa
- Tập địa chỉ *m*: gồm 10.000.000.000 địa chỉ.
- Hàm băm ở đây là $f(key) = key$. Hàm thực hiện loại bỏ dấu ‘-’ để số điện thoại trở thành 1 số nguyên. Vậy với số điện thoại là 0913-158020 sẽ được loại bỏ dấu trừ rồi đưa về số nguyên 913158020 (số 0 ở đầu lúc này vô nghĩa nên bị loại bỏ).

Kết quả

<i>Index</i>	<i>Key</i>	<i>Name</i>
0	0	Tý
1	1	
...	...	
913158020	0913158020	Dần
...	...	
9999999998	9999999998	Ngọ
9999999999	9999999999	Hợi

2. HÀM BĂM (hash function)

2.7. Một số ví dụ

ii. Ví dụ 2: cần lưu thông tin danh bạ điện thoại của khoảng 1000 người, trong đó:

- Khóa *k*: Lấy 3 số cuối của số điện thoại làm khóa.
- Tập địa chỉ *M*; phải gồm 1000 địa chỉ
- Hàm băm ở đây là $f(key) = key \% 1000$

Kết quả

Họ tên	Số điện thoại
Phạm văn Tý	0123456789
Trần Văn Sửu	9876543210
Lý thị Dần	8123456790
Hoàng thị Tý	0234567891

Kết quả dữ liệu sẽ được lưu trữ như sau:



Buckets		
000		
...		
210	9876543210	Trần Văn Sửu
...		
789	0123456789	Phạm văn Tý
790	8123456790	Lý thị Dần
...		
891	0234567891	Hoàng thị Tý

2. HÀM BĂM (hash function)

2.7. Một số ví dụ

iii. Ví dụ 3: cần tổ chức bảng băm để lưu trữ các tên khách hàng như sau: Mike, Kelly, Olivia, Isabelle, David, Kendy sao cho kích thước bảng băm là nhỏ nhất và không xảy ra đụng độ.

- Khóa *k*: là ký tự thứ 3 vì các ký tự này không trùng nhau.
- Tập địa chỉ *M*: gồm 26 địa chỉ.
- Hàm băm sử dụng là: $hf(key) = (int)Name[2] - 97$

Kết quả

<u>0</u>	<u>1</u>	<u>2</u>	<u>3</u>	<u>4</u>	<u>5</u>	<u>6</u>	<u>7</u>	<u>8</u>	<u>9</u>	<u>10</u>	<u>11</u>	<u>12</u>	<u>13</u>	<u>14</u>	<u>15</u>	<u>16</u>	<u>17</u>	<u>18</u>	<u>19</u>	<u>20</u>	<u>21</u>	<u>22</u>	<u>23</u>	<u>24</u>	<u>25</u>
<u>a</u>	b	c	d	e	f	g	h	<u>i</u>	j	<u>k</u>	<u>l</u>	m	<u>n</u>	o	p	q	r	s	t	u	<u>v</u>	w	x	y	z

Name	STT của ký tự thứ 3 (<i>k</i>)	<i>k mod 6</i>
Mike	10	4
Kelly	11	5
Olivia	8	2
Isabelle	0	0
David	21	3
Kendy	13	1

2. HÀM BĂM (hash function)

2.7. Một số ví dụ

iv. Ví dụ 4: cần đếm tần suất xuất hiện của các ký tự trong 1 chuỗi S (chỉ chứa các ký tự từ a-z). Giả sử với S=”ababdc”:

Do chuỗi chỉ chứa các ký tự từ a-z nên đề xuất tổ chức bảng băm *hTable* gồm 26 phần tử, phần tử *hTable[0]* sẽ chứa tần xuất xuất hiện của ký tự ‘a’, phần tử *hTable[1]* sẽ chứa tần xuất xuất hiện của ký tự ‘b’, ...

- Khóa *k*: Lấy *index* từ kết quả của hàm băm làm khóa
- Tập địa chỉ *M*: gồm 26 địa chỉ.
- Hàm băm để tìm *index* của ký tự *ch* là

$$f(ch) = ch - 'a'$$

Kết quả

char	index	value
a	0	2
b	1	2
c	2	1
d	3	1
e	4	0
...	...	0
...	...	0
y	24	0
z	25	0

2. HÀM BĂM (hash function)

2.7. Một số ví dụ

v. **Ví dụ 5:** cần tổ chức bảng băm để lưu trữ các giá trị sau: “abcdef”, “bcdefa”, “cdefab”, “defabc”.

- **Đề xuất thứ 1:**

- Khóa *k*: tổng mã ASCII của từng ký tự có trong chuỗi. Vì tất cả các chuỗi đã cho chứa các ký tự giống nhau với các hoán vị khác nhau, nên tổng mã ASCII của tất cả các chuỗi đều sẽ là 597. Nhắc lại mã ASCII của a=97, b=98, c=99, d=100, e=101, f=102.
- Tập địa chỉ *M*: gồm 7 địa chỉ.
- Hàm băm là $f(k) = k \% 7$ (7 là số nguyên tố).

Nhận xét: mất $O(n)$ thời gian (trong đó *n* là số lượng chuỗi) để truy cập một chuỗi cụ thể.
⇒ hàm băm không phải là hàm băm tốt.

<i>index</i>				
0				
1				
2	abcdef	bcdefa	cdefab	defabc
3				
4				
5				
6				

2. HÀM BĂM (hash function)

2.7. Một số ví dụ

v. Ví dụ 5: cần tổ chức bảng băm để lưu trữ các giá trị sau: “abcdef”, “bcdefa”, “cdefab”, “defabc”.

- Đề xuất thứ 2:

- Khóa *k*: tổng giá trị ASCII của các ký tự nhân với thứ tự tương ứng của chúng trong chuỗi.
- Tập địa chỉ *m*: gồm 2069 địa chỉ (2069 là số nguyên tố).
- Hàm băm $f(k) = k \% 2069$

String	Hash function	Index
abcdef	$((97*1 + 98*2 + 99*3 + 100*4 + 101*5 + 102*6)=2107)\%2069$	38
bcdefa	$((98*1 + 99*2 + 100*3 + 101*4 + 102*5 + 97*6)=2092)\%2069$	23
cdefab	$((99*1 + 100*2 + 101*3 + 102*4 + 97*5 + 98*6)=2083)\%2069$	14
defabc	$((100*1 + 101*2 + 102*3 + 97*4 + 98*5 + 99*6)=2080)\%2069$	11

Index	...	
10		
11		defabc
12		
13		
14		cdefab
15		
...		
23		bcdefa
24		
...		
38		abcdef
...		
2068		

NỘI DUNG

1. Giới thiệu
2. Hàm băm (hash function)
3. Các tác vụ trên bảng băm
4. Phân loại bảng băm
5. Bảng băm với phương pháp kết nối trực tiếp
6. Bảng băm với phương pháp kết nối hợp nhất
7. Bảng băm với phương pháp dò tuyến tính
8. Bảng băm với phương pháp dò bậc hai
9. Bảng băm với phương pháp băm kép
10. Một số bảng băm theo phân loại cấu trúc

3. CÁC TÁC VỤ TRÊN BẢNG BĂM

- i. Khởi tạo (Initialize):** Khởi tạo bảng băm, cấp phát vùng nhớ hay quy định số phần tử (kích thước) của bảng băm.
- ii. Kiểm tra rỗng (isEmpty):** kiểm tra bảng băm có rỗng hay không?
- iii. Lấy số lượng thực tế các phần tử đang có trên bảng băm (Count)**
- iv. Tìm kiếm (Search):** Tìm kiếm một phần tử trong bảng băm theo khoá k chỉ định trước.
- v. Thêm mới phần tử (Insert):** Thêm một phần tử vào bảng băm và số lượng phần tử hiện có của bảng băm tăng thêm 1.
- vi. Loại bỏ (Remove):** Loại bỏ một phần tử ra khỏi bảng băm và số lượng phần tử hiện có của bảng băm sẽ giảm đi 1.
- vii. Sao chép (Copy):** Tạo một bảng băm mới từ một bảng băm cũ đã có.
- viii. Duyệt (Traverse):** duyệt bảng băm theo thứ tự địa chỉ từ nhỏ đến lớn.

NỘI DUNG

1. Giới thiệu
2. Hàm băm (hash function)
3. Các tác vụ trên bảng băm
4. Phân loại bảng băm
5. Bảng băm với phương pháp kết nối trực tiếp
6. Bảng băm với phương pháp kết nối hợp nhất
7. Bảng băm với phương pháp dò tuyến tính
8. Bảng băm với phương pháp dò bậc hai
9. Bảng băm với phương pháp băm kép
10. Một số bảng băm theo phân loại cấu trúc

4. PHÂN LOẠI BẢNG BĂM

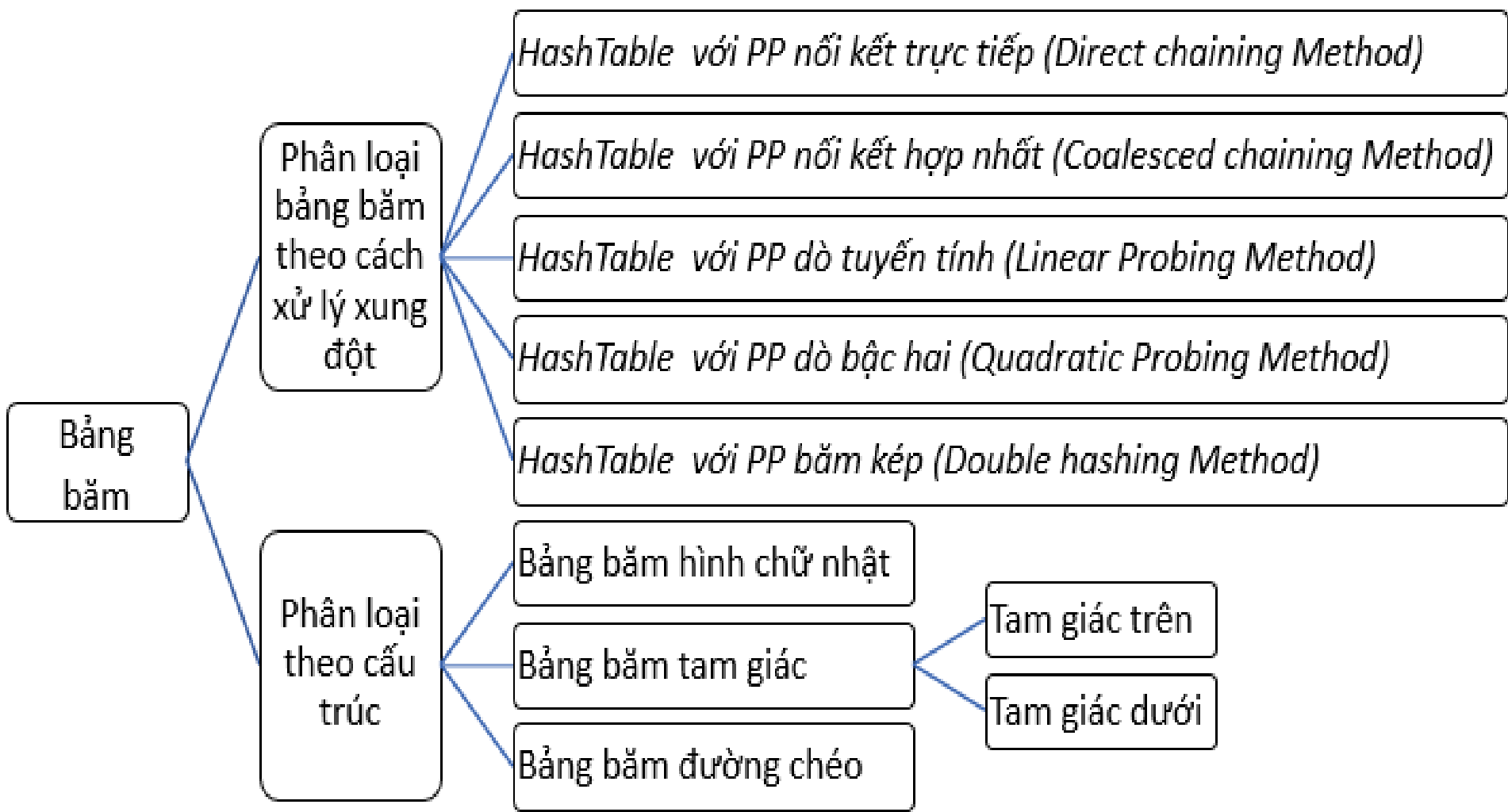
Khi xây dựng hàm băm chúng ta muốn các khoá khác nhau sẽ ánh xạ vào các địa chỉ khác nhau, nhưng trong thực tế dù hàm băm tốt như thế nào thì khi thực hiện ánh xạ các khoá khác nhau lại cho cùng 1 địa chỉ, trường hợp này gọi là xung đột (hay va chạm - *Hash collision*).

Do đó, để duy trì hiệu suất của bảng băm, điều quan trọng là phải quản lý va chạm thông qua các kỹ thuật giải quyết va chạm khác nhau.

Có nhiều cách phân loại bảng băm, trong phần này giới thiệu 2 dạng phân loại bảng băm

- Phân loại theo cấu trúc
- Phân loại theo cách xử lý xung đột

4. Phân loại bảng băm



NỘI DUNG

1. Giới thiệu
2. Hàm băm (hash function)
3. Các tác vụ trên bảng băm
4. Phân loại bảng băm
5. Bảng băm với phương pháp kết nối trực tiếp
6. Bảng băm với phương pháp kết nối hợp nhất
7. Bảng băm với phương pháp dò tuyến tính
8. Bảng băm với phương pháp dò bậc hai
9. Bảng băm với phương pháp băm kép
10. Một số bảng băm theo phân loại cấu trúc

5. BẢNG BĂM VỚI PHƯƠNG PHÁP KẾT NỐI TRỰC TIẾP (Direct chaining Method)

5.1. Cấu trúc dữ liệu

- *Node*: là thông tin của node, có thể thuộc các kiểu dữ liệu số nguyên, chuỗi, cấu trúc, ...
- Khai báo cấu trúc node

```
struct NODE
{
    int key;
    struct NODE *pNext;
};
```



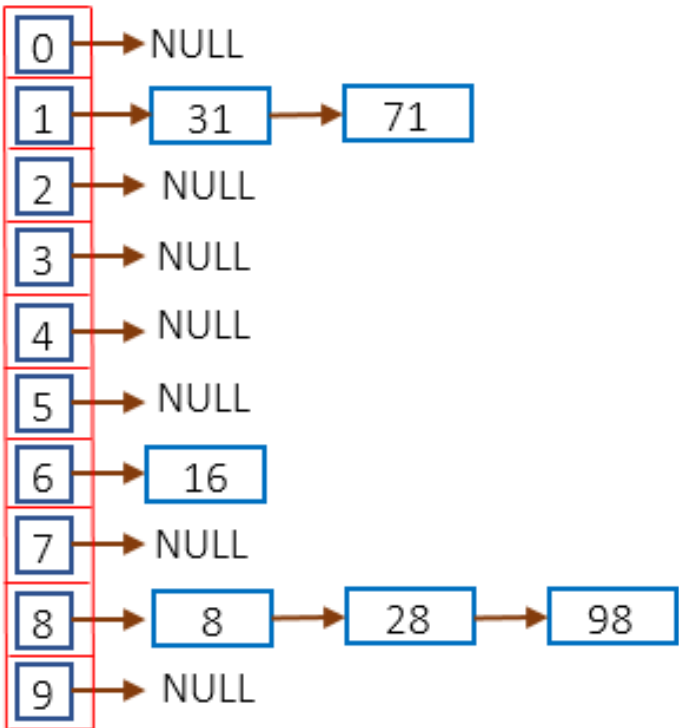
5. Bảng băm với phương pháp kết nối trực tiếp (Direct chaining Method)

5.2. Tổ chức bảng băm

- Là một danh sách kê có M node (được đánh số từ 0 đến M-1), mỗi node của danh sách có kiểu là 1 con trỏ, trỏ đến các node có cấu trúc dữ liệu đã được khai báo ở trên.

```
typedef struct NODE *NodePointer;  
NodePointer bucket[M];
```

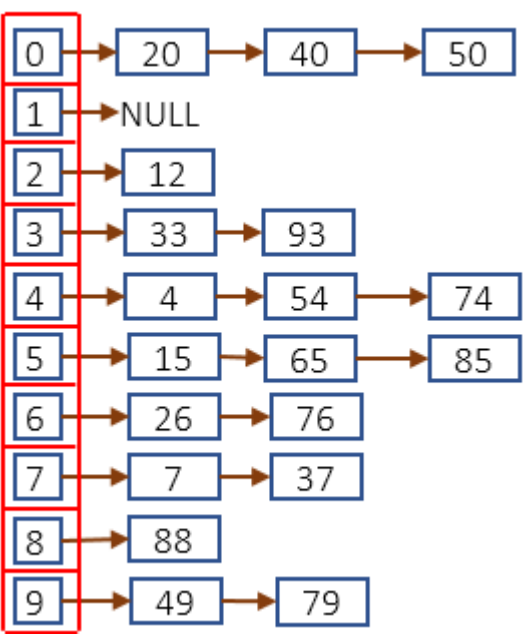
```
//khai báo cấu trúc node  
struct NODE  
{ int key;  
  struct NODE *pNext;  
};
```



5.3. Một số tác vụ

5.3.1. Tác vụ thêm

- Khi thêm 1 node có khoá *key* vào bảng băm, hàm băm $hf(key)$ sẽ xác định địa chỉ index trong khoảng từ 0 đến $M-1$ ứng với danh sách liên kết thứ *index* mà node này sẽ được thêm vào.
- Khi node được chèn vào danh sách liên kết, sẽ được chèn sao cho giá trị của *key* tăng dần.
- Ví dụ: minh hoạ bảng băm có tập khoá là tập số tự nhiên, tập địa chỉ có 10 địa chỉ và chọn hàm băm là $hf(key) = key \% 10$.

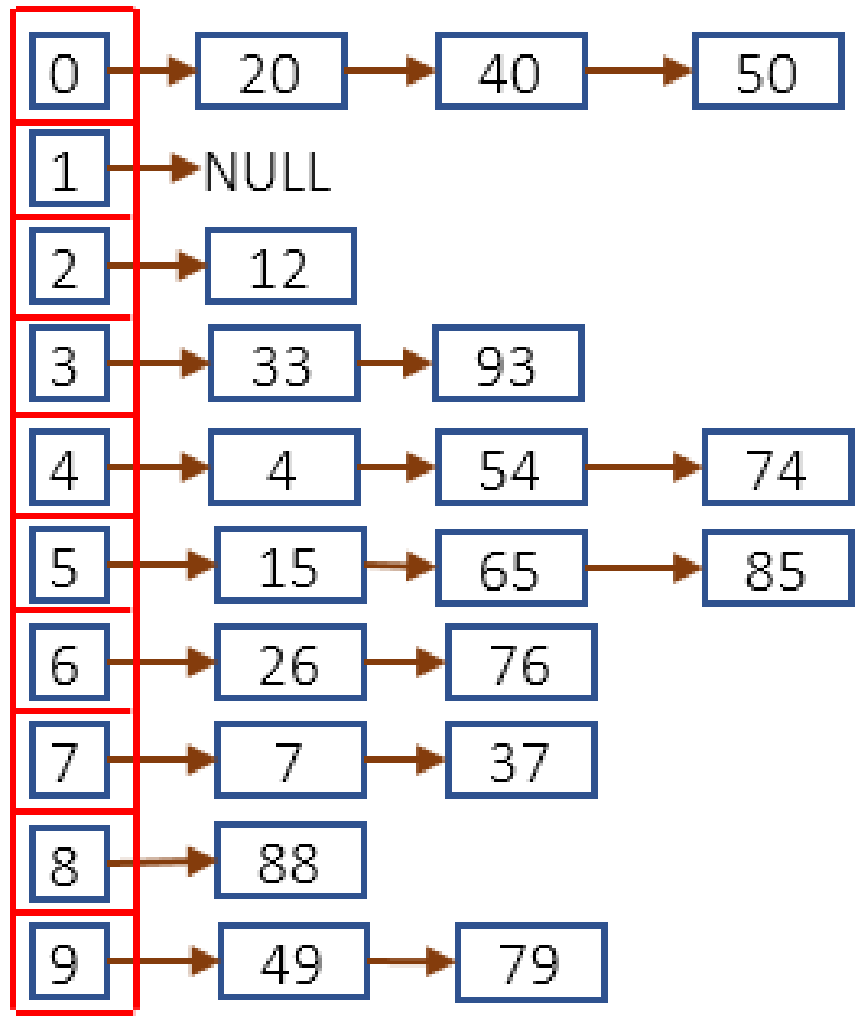


5. Bảng băm với phương pháp kết nối trực tiếp (Direct chaining Method)

5.3. Một số tác vụ

5.3.2. Tác vụ tìm

- Khi tìm kiếm một node có khoá key trên bảng băm, hàm băm hf(key) sẽ xác định địa chỉ index trong khoảng từ 0 đến M-1 ứng với danh sách liên kết index có thể chứa node, việc tìm kiếm node trên bảng băm quy về bài toán tìm kiếm trên danh sách liên kết.

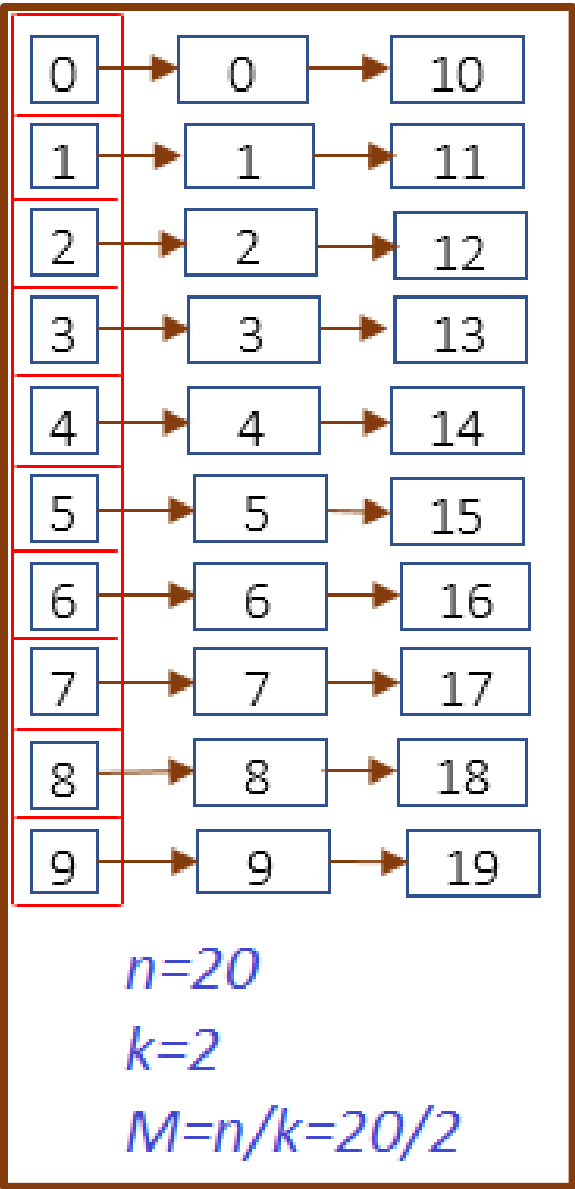


5.4. Nhận xét

- Về hàm băm

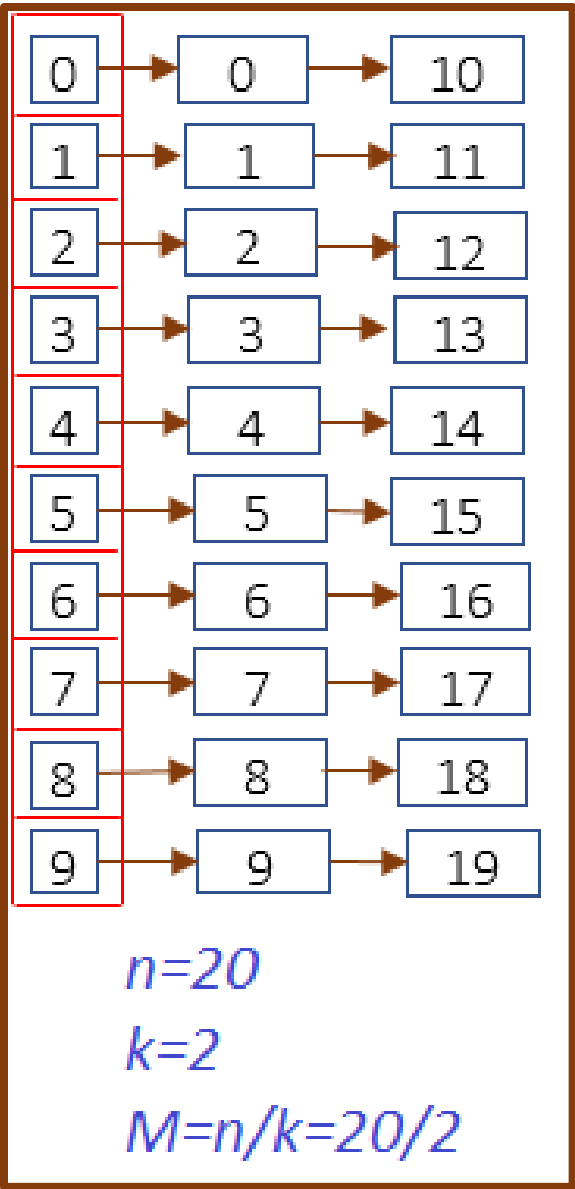
Để tốc độ thực hiện các phép toán trên bảng hiệu quả thì cần chọn hàm băm sao cho băm đều n phần tử của bảng băm cho M bucket, lúc này trung bình mỗi bucket sẽ có n/M phần tử.

⇒ phép toán search sẽ thực hiện việc tìm kiếm tuyến tính trên bucket nên thời gian tìm kiếm lúc này là $O(n/M)$ – nghĩa là, nhanh gấp n lần so với việc tìm kiếm trên một danh sách liên kết có n phần tử.



5.4. Nhận xét

- Về bảng băm: chọn giá trị của M:
 - M càng lớn:
 - Tốc độ thực hiện các phép toán trên bảng băm càng nhanh.
 - Cần dùng nhiều bộ nhớ.
⇒ cần điều chỉnh M để dung hòa giữa tốc độ truy xuất và dung lượng bộ nhớ.
 - $M = n$: năng xuất tương đương với truy xuất trên mảng (có bậc $O(1)$), tuy nhiên tốn nhiều bộ nhớ.
 - $M = n/k$ ($k = 2, 3, 4, \dots$): ít tốn bộ nhớ hơn k lần, nhưng tốc độ chậm đi k lần.



NỘI DUNG

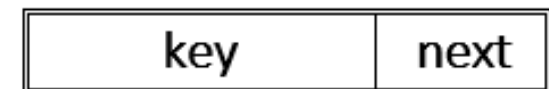
1. Giới thiệu
2. Hàm băm (hash function)
3. Các tác vụ trên bảng băm
4. Phân loại bảng băm
5. Bảng băm với phương pháp kết nối trực tiếp
6. Bảng băm với phương pháp kết nối hợp nhất
7. Bảng băm với phương pháp dò tuyến tính
8. Bảng băm với phương pháp dò bậc hai
9. Bảng băm với phương pháp băm kép
10. Một số bảng băm theo phân loại cấu trúc

6. BẢNG BĂM VỚI PHƯƠNG PHÁP KẾT NỐI HỢP NHẤT (Coalesced chaining Method)

6.1. Cấu trúc dữ liệu

- **Node**: sử dụng trong trường hợp này có cấu trúc như sau:
 - Trường **key**: có kiểu dữ liệu là số nguyên, chứa khóa của node. Khi khởi tạo, key sẽ có giá trị là **NULLKEY** ($=-1$).
 - Trường **next**: có kiểu dữ liệu là số nguyên, đóng vai trò làm con trỏ chỉ vị trí của node tiếp trên mảng nếu có xung đột về **key**. Được gọi là đóng vai trò vì giá trị này là một số nguyên cụ thể chứ không phải là địa chỉ như trong danh sách liên kết. Khi khởi tạo, **next** sẽ có giá trị là **NULLKEY** ($=-1$).

```
//khai báo cấu trúc  
typedef struct NODE  
{  
    int key;  
    int next;  
};
```



6.1. Cấu trúc dữ liệu

- Cấu trúc bảng băm

- Sử dụng mảng gồm M node.
- Các node bị xung đột địa chỉ được nối kết với nhau qua trường next tương tự như cách tổ chức của danh sách liên kết đơn.

```
//khai báo mảng cấu trúc  
struct NODE hashtable[M];
```

NULLKEY	-1
NULLKEY	-1
NULLKEY	-1
...	...
...	...
...	...
NULLKEY	-1

6.2. Một số tác vụ

6.2.1. Tác vụ thêm

- Khi thêm một node có khóa key vào bảng băm, hàm băm hf(key) sẽ xác định địa chỉ index trong khoảng từ 0 đến M – 1.
- Nếu chưa bị xung đột thì thêm node mới sẽ được đặt tại địa chỉ index này.
- Nếu bị xung đột thì node mới sẽ được cấp phát là node trống gần nhất tính từ phía cuối mảng. Thực hiện cập nhập liên kết next sao cho các node bị xung đột hình thành một danh sách liên kết.

HashTable
có 10 địa chỉ,
hf(key)=key % 10.

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

Khởi đầu

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

Sau khi thêm 10

0	10	-1
1		-1
2		-1
3		-1
4		-1
5		-1
6		-1
7		-1
8		-1
9	79	-1

Sau khi thêm 79

0	10	8
1		-1
2		-1
3		-1
4		-1
5		-1
6		-1
7		-1
8	40	-1
9	79	-1

Sau khi thêm 40

6. BẢNG BĂM VỚI PHƯƠNG PHÁP KẾT NỐI HỢP NHẤT (Coalesced chaining Method)

6.2. Một số tác vụ

6.2.1. Tác vụ thêm

- Ví dụ: Cho bảng băm có tập khóa K là số tự nhiên, tập địa chỉ có 10 địa chỉ (M=10), chọn hàm băm $hf(key)=key \% 10$. Thực hiện thêm các node 10, 15, 26, 30, 25, 35 vào bảng băm.

• *Hình (a)*: sau khi thêm 3 node 10, 15, 26 vào bảng băm – lúc này chưa bị xung đột.

(a)

0	10	-1
1		-1
2		-1
3		-1
4		-1
5	15	-1
6	26	-1
7		-1
8		-1
9		-1

• *Hình (b)*: thêm node 30 vào bảng băm – bị xung đột tại địa chỉ 0 – node 30 được cấp phát tại địa chỉ 9, trường next của node tại địa chỉ 0 được gán là 9.

(b)

0	10	9
1		-1
2		-1
3		-1
4		-1
5	15	-1
6	26	-1
7		-1
8		-1
9	30	-1

6. BẢNG BĂM VỚI PHƯƠNG PHÁP KẾT NỐI HỢP NHẤT (Coalesced chaining Method)

6.2. Một số tác vụ

6.2.1. Tác vụ thêm

- Ví dụ: Cho bảng băm có tập khóa K là số tự nhiên, tập địa chỉ có 10 địa chỉ (M=10), chọn hàm băm $hf(key)=key \% 10$. Thực hiện thêm các node 10, 15, 26, 30, 25, 35 vào bảng băm.

- *Hình (c)*: thêm node 25 vào bảng băm – bị xung đột tại địa chỉ 5 – node 25 được cấp phát tại địa chỉ 8, trường next của node tại địa chỉ 5 được gán là 8.
- *Hình (d)*: thêm node 35 vào bảng băm – bị xung đột tại địa chỉ 5 và địa chỉ 8 – node 35 được cấp phát tại địa chỉ 7, trường next của node tại địa chỉ 8 được gán là 7.

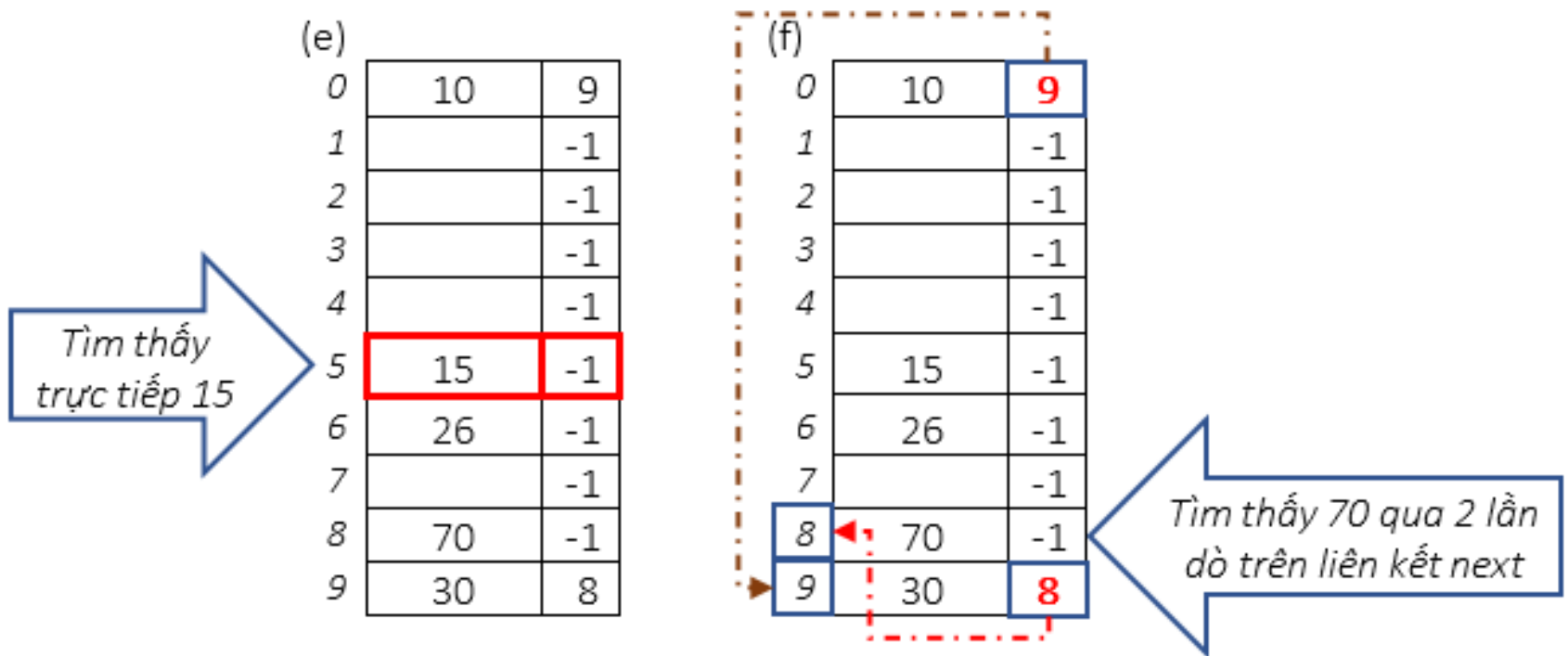
(c)			(d)		
0	10	9	0	10	9
1		-1	1		-1
2		-1	2		-1
3		-1	3		-1
4		-1	4		-1
5	15	8	5	15	8
6	26	-1	6	26	-1
7		-1	7	35	-1
8	25	-1	8	25	7
9	30	-1	9	30	-1

6. BẢNG BĂM VỚI PHƯƠNG PHÁP KẾT NỐI HỢP NHẤT (Coalesced chaining Method)

6.2. Một số tác vụ

6.2.2. Tác vụ tìm

- Khi tìm một node có khóa key trong bảng băm, hàm băm $hf(key)$ sẽ xác định địa chỉ $index$ trong khoảng từ 0 đến $M-1$. Thực hiện tìm node khóa key tương tự như cách tìm trong danh sách liên kết với vị trí xuất phát từ địa chỉ $index$.



NỘI DUNG

1. Giới thiệu
2. Hàm băm (hash function)
3. Các tác vụ trên bảng băm
4. Phân loại bảng băm
5. Bảng băm với phương pháp kết nối trực tiếp
6. Bảng băm với phương pháp kết nối hợp nhất
7. Bảng băm với phương pháp dò tuyến tính
8. Bảng băm với phương pháp dò bậc hai
9. Bảng băm với phương pháp băm kép
10. Một số bảng băm theo phân loại cấu trúc

7. BẢNG BĂM VỚI PHƯƠNG PHÁP DÒ TUYẾN TÍNH

(Linear Probing Method)

7.1. Cấu trúc dữ liệu

- **Node**: là thông tin của node, có thể thuộc các kiểu dữ liệu số nguyên, chuỗi, cấu trúc, ... Nếu kiểu dữ liệu không phải là kiểu cấu trúc thì key chính là dữ liệu đó, ngược lại, nếu kiểu dữ liệu của node là cấu trúc thì trong cấu trúc đó cần khai báo 1 trường làm key.
- **Bảng băm**: là một danh sách kê có M node

0	node
1	node
...	...
...	...
M-2	node
M-1	node

7.2. Một số tác vụ

7.2.1. Tác vụ thêm

- Khi thêm một node có khóa key vào bảng băm:
 - Hàm băm $hf(key)$ sẽ xác định địa chỉ index trong khoảng từ 0 đến $M-1$.
 - Nếu chưa bị xung đột thì thêm node mới tại địa chỉ index này.
 - Nếu bị xung đột thì hàm băm sẽ tìm địa chỉ kế tiếp, lớn hơn và gần với index nhất để lưu node có khóa key.

7. BẢNG BĂM VỚI PHƯƠNG PHÁP DÒ TUYẾN TÍNH (Linear Probing Method)

7.2. Một số tác vụ

7.2.1. Tác vụ thêm

- Ví dụ: cho bảng băm có tập khóa là số tự nhiên, tập địa chỉ M có 10 địa chỉ, chọn hàm băm $hf(key)=key \% 10$. Tiến trình thêm các node sau vào bảng băm 32, 53, 22, 92, 17, 34

- Hình (a): sau khi thêm 2 node 32, 53 vào bảng băm – lúc này chưa bị xung đột.
- Hình (b): thêm node 22 và 92 vào bảng băm – bị xung đột tại địa chỉ 2, node 22 được cấp phát tại địa chỉ 4, node 92 được cấp phát tại địa chỉ 5.

(a)	(b)
0	0
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9

7. BẢNG BĂM VỚI PHƯƠNG PHÁP DÒ TUYẾN TÍNH (Linear Probing Method)

7.2. Một số tác vụ

7.2.1. Tác vụ thêm

- Ví dụ: cho bảng băm có tập khóa là số tự nhiên, tập địa chỉ M có 10 địa chỉ, chọn hàm băm $hf(key) = key \% 10$. Tiến trình thêm các node sau vào bảng băm 32, 53, 22, 92, 17, 34
- Hình (c): thêm node 17 và 34 vào bảng băm: node 17 không bị xung đột được cấp phát tại địa chỉ 7, node 34 bị xung đột tại địa chỉ 4, được cấp tại địa chỉ 6.

(c)

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

7. BẢNG BĂM VỚI PHƯƠNG PHÁP DÒ TUYẾN TÍNH (Linear Probing Method)

7.2. Một số tác vụ

7.2.2. Tác vụ tìm

- Khi tìm một node có khóa key trong bảng băm, hàm băm $hf(key)$ sẽ xác định địa chỉ index trong khoảng từ 0 đến $M-1$, sau đó thực hiện tìm node khóa key trong đoạn từ vị trí $index$ đến vị trí trống đầu tiên có chỉ số $>index$ (do đó nếu không có phần tử nào trống thì đoạn cần tìm là từ $index$ đến $M-1$).
- Hàm băm lại của phương pháp dò tìm tuyến tính là truy xuất địa chỉ kế tiếp. Hàm băm lại được biểu diễn bằng công thức sau:

$$HF(key)=(hf(key)+i)\%M$$

(d)

0	
1	
2	32
3	53
4	22
5	92
6	34
7	12
8	62
9	72

Để tìm
key=72,
gần như
phải tìm
tuyến
tính trên
đoạn này

NỘI DUNG

1. Giới thiệu
2. Hàm băm (hash function)
3. Các tác vụ trên bảng băm
4. Phân loại bảng băm
5. Bảng băm với phương pháp kết nối trực tiếp
6. Bảng băm với phương pháp kết nối hợp nhất
7. Bảng băm với phương pháp dò tuyến tính
8. Bảng băm với phương pháp dò bậc hai
9. Bảng băm với phương pháp băm kép
10. Một số bảng băm theo phân loại cấu trúc

8. BẢNG BĂM VỚI PHƯƠNG PHÁP DÒ BẬC HAI

(Quadratic Probing Method)

8.1. Cấu trúc dữ liệu

- **Node**: là thông tin của node, có thể thuộc các kiểu dữ liệu số nguyên, chuỗi, cấu trúc.
- **Bảng băm**:
 - Bảng băm được cài đặt bằng danh sách kê có M phần tử, mỗi phần tử của bảng băm là một mẫu tin có một trường key để chứa khóa các phần tử. Trong đó số địa chỉ M nên chọn là số nguyên tố.
 - Khi khởi động bảng băm, tất cả trường key bị gán NULL.
 - Nhận xét Bảng băm dùng phương pháp dò tuyến tính bị hạn chế do rải các phần tử không đều, bảng băm với phương pháp dò bậc hai rải các phần tử đều hơn.

0	node
1	node
...	...
...	...
M-2	node
M-1	node

8. Bảng băm với phương pháp dò bậc hai (Quadratic Probing Method)

8.1. Cấu trúc dữ liệu

- *Hàm băm*:

- Hàm băm lần đầu (hàm băm chính) thường có dạng:

$$f_0(key) = key \% 10$$

- Hàm băm lại của phương pháp dò bậc hai là truy xuất các địa chỉ cách bậc 2. Do đó hàm băm lại lần thứ i được biểu diễn bằng công thức sau:

$$f_i(key) = (f_0(key) + i^2) \% M$$

với $f(key)$ là hàm băm chính của bảng băm.

8.2. Một số tác vụ

8.2.1. Tác vụ thêm

- Khi thêm một node có khóa key vào bảng băm, hàm băm $hf(key)$ sẽ xác định địa chỉ index trong khoảng từ 0 đến $M-1$.
 - Nếu chưa bị xung đột thì thêm node mới tại địa chỉ index này.
 - Nếu bị xung đột thì hàm băm lại lần 1 (f_1) sẽ xác định địa chỉ mới cách địa chỉ của i là 1^2 vị trí, nếu lại bị xung đột thì hàm băm lại lần 2 (f_2) sẽ xác định địa chỉ mới cách địa chỉ của i là 2^2 vị trí, ... Quá trình cứ tiếp tục cho đến khi tìm được vị trí trống và thêm phần tử vào địa chỉ này.

8. Bảng băm với phương pháp dò bậc hai (Quadratic Probing Method)

8.2. Một số tác vụ

8.2.1. Tác vụ thêm

- Ví dụ: cho bảng băm có tập khóa là số tự nhiên, tập địa chỉ M có 10 địa chỉ, chọn 2 hàm băm

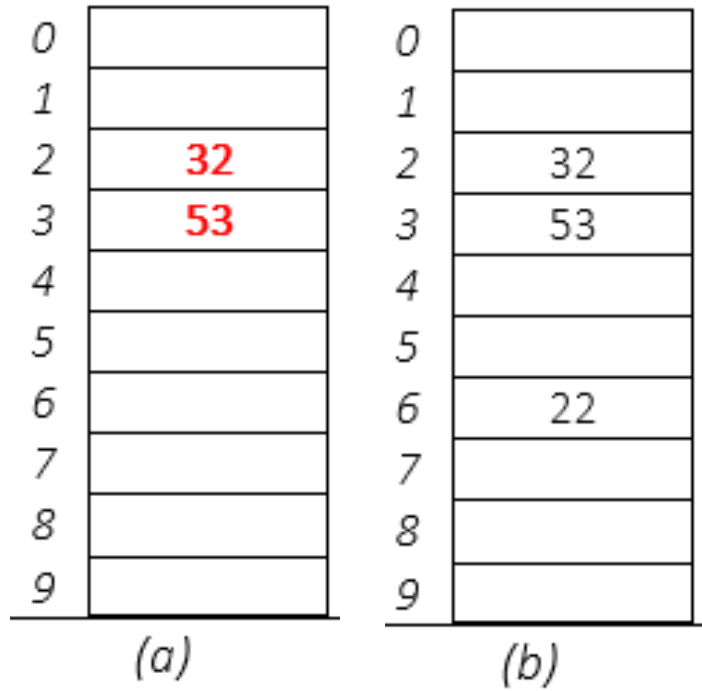
$f_0(key) = key \% M$

$f_i(key) = (f_0(key) + i^2) \% M.$

Tiến trình thêm các node sau vào bảng băm 32, 53, 22, 92, 17, 34

- Hình (a): sau khi thêm 2 node 32, 53 vào bảng băm – lúc này chưa bị xung đột.

- Hình (b): thêm node 22
 - Với $f_0 = 22 \% 10 = 2$; vị trí này đã chứa số 32.
 - Băm lại lần 1 (f_1) = $(i+1^2) \% 10 = (2+1) \% 10 = 3$; vị trí này đã chứa số 53.
 - Băm lại lần 2 (f_2) = $(i+2^2) \% 10 = (2+4) \% 10 = 6 \Rightarrow$ node 22 được đưa vào vị trí 6.



8. Bảng băm với phương pháp dò bậc hai (Quadratic Probing Method)

8.2. Một số tác vụ

8.2.1. Tác vụ thêm

- Ví dụ: cho bảng băm có tập địa chỉ M=10, chọn 2 hàm băm

$f_0(key) = key \% M$

$f_i(key) = (f_0(key) + i^2) \% M.$

Tiến trình thêm các node sau vào bảng băm 32, 53, 22, 92, 17, 34

- Hình (c): thêm node 92
 - Với $f_0 = 92 \% 10 = 2$; vị trí này đã chứa số 32.
 - Băm lại lần 1 (f_1) = $(i+1^2) \% 10 = (2+1) \% 10 = 3$; vị trí này đã chứa số 53.
 - Băm lại lần 2 (f_2) = $(i+2^2) \% 10 = (2+4) \% 10 = 6$; vị trí này đã chứa số 22.
 - Băm lại lần 3 (f_3) = $(i+3^2) \% 10 = (2+9) \% 10 = 1 \Rightarrow$ node 92 được đưa vào vị trí 1.

0		0	
1		1	92
2	32	2	32
3	53	3	53
4		4	
5		5	
6	22	6	22
7		7	
8		8	
9		9	
(b)		(c)	

8. Bảng băm với phương pháp dò bậc hai (Quadratic Probing Method)

8.2. Một số tác vụ

8.2.1. Tác vụ thêm

- Ví dụ: cho bảng băm có tập địa chỉ M=10, chọn 2 hàm băm

$f_0(key) = key \% M$ $f_i(key) = (f_0(key) + i^2) \% M.$

Tiến trình thêm các node sau vào bảng băm 32, 53, 22, 92, 17, 34

- Hình (d): thêm node 17 và 62
 - ▢ Node 17 với $i = 17 \% 10 = 7 \Rightarrow$ không bị xung đột, cấp phát tại địa chỉ 7.
 - ▢ Node 62: với $i = 62 \% 10 = 2$; vị trí này đã chứa số 32.
 - Băm lại lần 1(f_1) = $(i+1^2) \% 10 = (2+1)\%10 = 3$; vị trí này đã chứa số 53.
 - Băm lại lần 2(f_2) = $(i+2^2) \% 10 = (2+4)\%10 = 6$; vị trí này đã chứa số 22.
 - Băm lại lần 3(f_3) = $(i+3^2) \% 10 = (2+9)\%10 = 1$; vị trí này đã chứa số 92.
 - Băm lại lần 4(f_4) = $(i+4^2) \% 10 = (2+16)\%10 = 8 \Rightarrow$ node 62 được đưa vào vị trí 8.

0	
1	92
2	32
3	53
4	
5	
6	22
7	
8	
9	

(c)

0	
1	92
2	32
3	53
4	
5	
6	22
7	17
8	62
9	

(d)

8. Bảng băm với phương pháp dò bậc hai (*Quadratic Probing Method*)

8.2. Một số tác vụ

8.2.2. Tác vụ tìm

Tìm kiếm một phần tử có khóa key trong bảng băm thì xét phần tử tại địa chỉ $i=f(key)$, nếu chưa tìm thấy thì xét phần tử cách i $1^2, 2^2, \dots$. Quá trình cứ tiếp tục đến khi tìm được khóa (trường hợp tìm thấy) hoặc rơi vào địa chỉ trống (trường hợp không tìm thấy).

NỘI DUNG

1. Giới thiệu
2. Hàm băm (hash function)
3. Các tác vụ trên bảng băm
4. Phân loại bảng băm
5. Bảng băm với phương pháp kết nối trực tiếp
6. Bảng băm với phương pháp kết nối hợp nhất
7. Bảng băm với phương pháp dò tuyến tính
8. Bảng băm với phương pháp dò bậc hai
9. Bảng băm với phương pháp băm kép
10. Một số bảng băm theo phân loại cấu trúc

9. BẢNG BĂM VỚI PHƯƠNG PHÁP BĂM KÉP

(Double Hashing Method)

9.1. Cấu trúc dữ liệu

- **Bảng băm:** cài đặt bằng danh sách kê có M phần tử tương tự như 2 phương pháp dò tuyến tính và dò bậc hai (với M được đề nghị là số nguyên tố).
- **Node:** là thông tin của node, có thể thuộc các kiểu dữ liệu số nguyên, chuỗi, cấu trúc, ... Nếu kiểu dữ liệu không phải là kiểu cấu trúc thì key chính là dữ liệu đó, ngược lại, nếu kiểu dữ liệu của node là cấu trúc thì trong cấu trúc đó cần khai báo 1 trường làm key. Khi khởi động bảng băm, tất cả trường key được gán trị NULL.
- **Hàm băm:** bảng băm này dùng hai hàm băm khác nhau với mục đích để rải rác đều các phần tử trên bảng băm.

0	node
1	node
...	...
...	...
M-2	node
M-1	node

9.2. Một số tác vụ

9.2.1. Tác vụ thêm

Khi thêm phần tử có khoá key vào bảng băm, cần xác định 2 giá trị: $i = f_1(\text{key})$ và $j = f_2(\text{key})$. Thông qua i và j sẽ giúp xác định địa chỉ i và j trong khoảng từ 0 đến $M-1$:

- Nếu không bị xung đột thì thêm phần tử mới tại địa chỉ i .
- Nếu bị xung đột thì hàm băm lại lần 1, f_1 sẽ xét địa chỉ mới $i+j$. Nếu lại bị xung đột thì hàm băm lại lần 2, f_2 sẽ xét địa chỉ $i+2j$, ... Quá trình sẽ được tiếp tục cho đến khi nào tìm được địa chỉ trống và thêm phần tử vào địa chỉ này.

9. Bảng băm với phương pháp băm kép (Double Hashing Method)

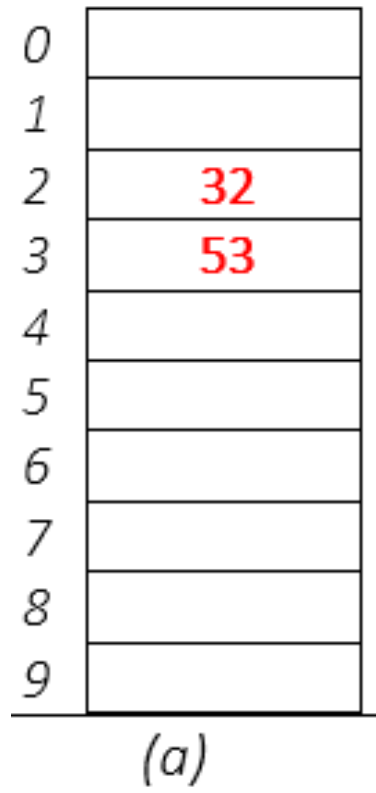
9.2. Một số tác vụ

9.2.1. Tác vụ thêm

- Ví dụ: cho bảng băm có tập khóa là số tự nhiên, tập địa chỉ M có 10 địa chỉ, chọn 2 hàm băm là:
 $f1(key) = key \% M$
 $f2(key) = (M-2) - key \% (M-2)$

Tiến trình thêm các node sau vào bảng băm 32, 53, 22, 92, 17, 62:

- Hình (a): sau khi thêm 2 node 32, 53 vào bảng băm – lúc này chưa bị xung đột.



9. Bảng băm với phương pháp băm kép (Double Hashing Method)

9.2. Một số tác vụ

9.2.1. Tác vụ thêm

- Ví dụ: thêm các node sau vào bảng băm 32, 53, 22, 92, 17, 62

$f1(key) = key \% M$ và $f2(key) = (M-2) - key \% (M-2)$

- *Hình (b)*: thêm node 22 và 92 vào bảng băm
 - ▢ Node **22**: với $i = 22 \% 10 = 2$; $j = (10-2) - (22 \% (10-2)) = 8-6 = 2$.
 - Khi $i = 2 \Rightarrow$ bị xung đột với node 32
 - Băm lại lần 1 với địa chỉ $f1 = i+j = 2+2 = 4 \Rightarrow$ node 22 được đưa vào vị trí 4
 - ▢ Node **92**: với $i = 92 \% 10 = 2$; $j = (10-2) - (92 \% (10-2)) = 8 - (92 \% 8) = 8-4 = 4$.
 - Khi $i = 2 \Rightarrow$ bị xung đột với node 32
 - Băm lại lần 1 với địa chỉ $i+j = 2+4 = 6 \Rightarrow$ node 92 được đưa vào vị trí 6

0	
1	
2	32
3	53
4	22
5	
6	92
7	
8	
9	

(b)

9. Bảng băm với phương pháp băm kép (Double Hashing Method)

9.2. Một số tác vụ

9.2.1. Tác vụ thêm

- Ví dụ: thêm các node sau vào bảng băm 32, 53, 22, 92, 17, 62

$f1(key) = key \% M$ và $f2(key) = (M-2) - key \% (M-2)$

• Hình (c): thêm node 17 và 62 vào bảng băm:

- Node 17 với $i = 17 \% 10 = 7 \Rightarrow$ không bị xung đột được cấp phát tại địa chỉ 7.
- Node 62: với $i = 62 \% 10 = 2; j = (10-2) - (62 \% (10-2)) = 8 - (62 \% 8) = 8-6 = 2$.
 - Khi $i = 2 \Rightarrow$ bị xung đột với node 32
 - Băm lại lần 1 với địa chỉ $i+j = 2+2 = 4 \Rightarrow$ bị xung đột với node 22
 - Băm lại lần 2 với địa chỉ $i+2j = 2+4 = 6 \Rightarrow$ bị xung đột với node 92
 - Băm lại lần 3 với địa chỉ $i+3j = 2+6 = 8 \Rightarrow$ node 62 được đưa vào vị trí 8.

0	
1	
2	32
3	53
4	22
5	
6	92
7	17
8	62
9	

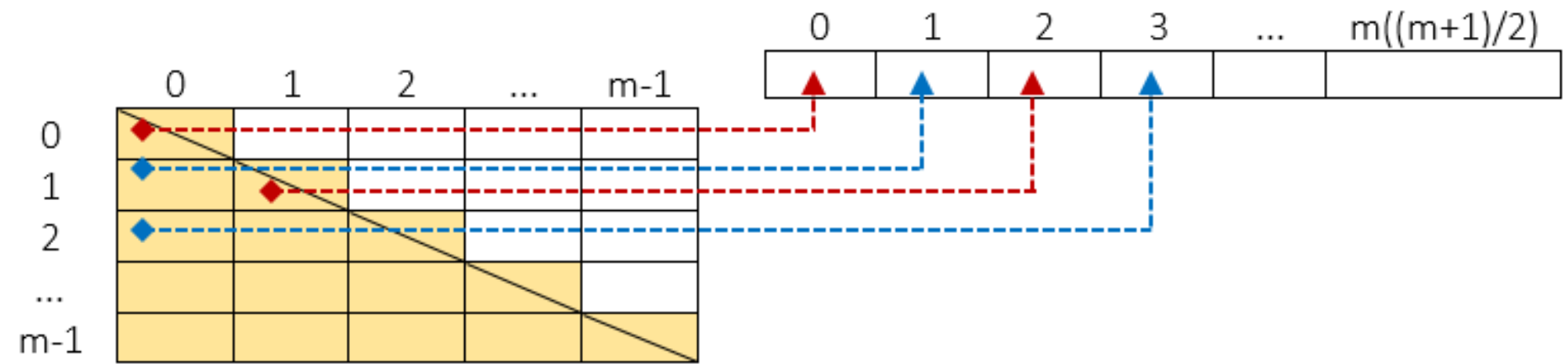
(c)

NỘI DUNG

1. Giới thiệu
2. Hàm băm (hash function)
3. Các tác vụ trên bảng băm
4. Phân loại bảng băm
5. Bảng băm với phương pháp kết nối trực tiếp
6. Bảng băm với phương pháp kết nối hợp nhất
7. Bảng băm với phương pháp dò tuyến tính
8. Bảng băm với phương pháp dò bậc hai
9. Bảng băm với phương pháp băm kép
10. Một số bảng băm theo phân loại cấu trúc

[illegible]

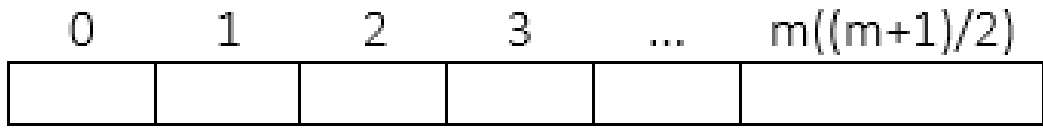
10.2. Hash Table tam giác dưới (trong ma trận vuông $m \times m$)



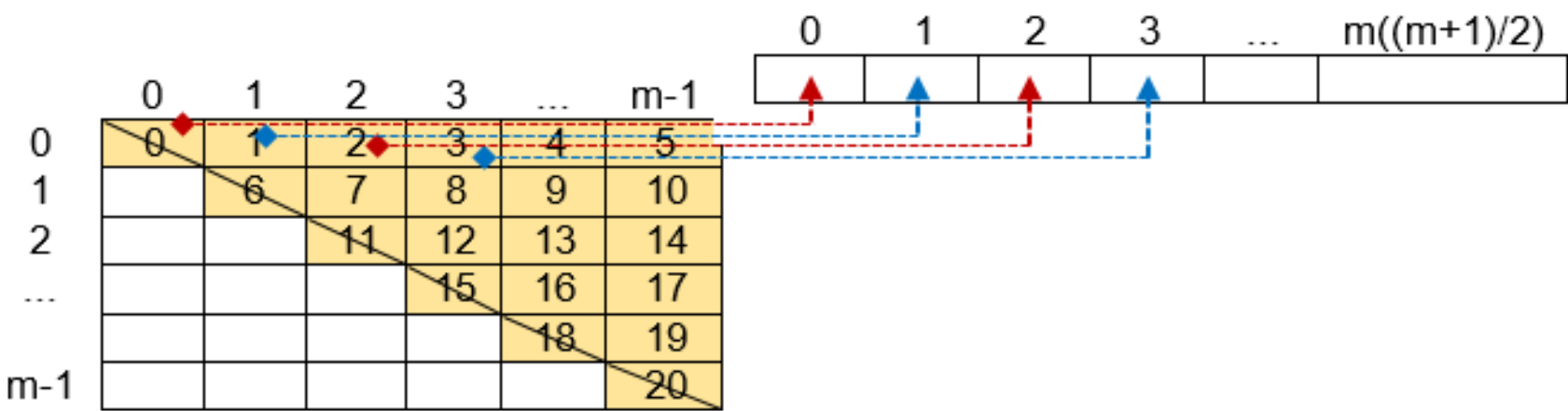
- Mỗi phần tử trên bảng tam giác dưới tương ứng với hàng i , cột j . Với $m-1 \geq i \geq 0$ và $i \geq j \geq 0$.
- Chuyển đổi phần tử thuộc hàng i , cột j trong bảng băm tam giác dưới sang danh sách kê (mảng 1 chiều) được cho bởi hàm băm:

$$hf(i,j) = i*(i+1)/2 + j$$

- HashTable tam giác dưới được mô tả bởi một danh sách kê:



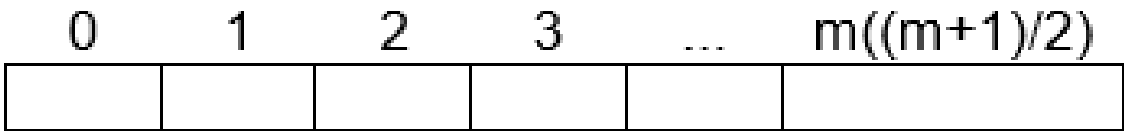
10.3. Hash Table tam giác trên (trong ma trận vuông $m \times m$)



- Mỗi phần tử trên bảng tam giác tương ứng với hàng i , cột j . Với $m-1 \geq j \geq 0$ và $j \leq i \leq m-1$.
- Chuyển đổi phần tử thuộc hàng i , cột j trong bảng băm tam giác trên sang danh sách kê (mảng 1 chiều) được cho bởi hàm băm:

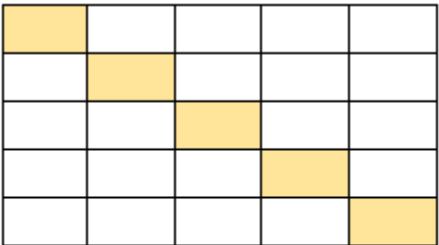
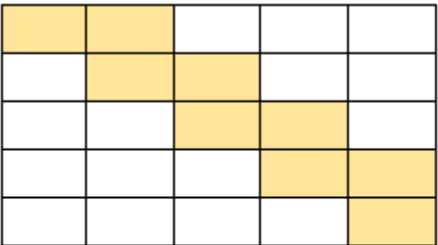
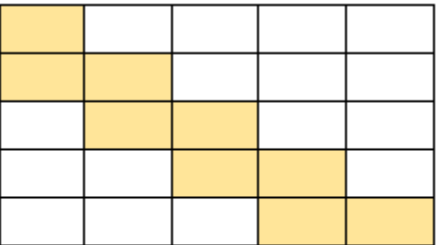
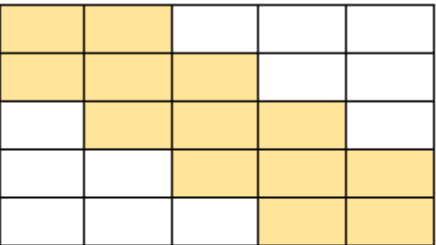
$$hf(i,j) = (i * ((2*m) - i + 1) / 2) + (j - i) :$$

- HashTable tam giác trên được mô tả bởi một danh sách kê:



10.4. Hash Table đường chéo (trong ma trận vuông $m \times m$)

- Gồm các dạng:

			
① $hf(i,j) = i = j$	② $hf(i,j) = i = j$ or $hf(i,j) = i = j - 1$	③ $hf(i,j) = i = j$ or $hf(i,j) = i = j + 1$	④ $hf(i,j) = i = j$ or $hf(i,j) = i = j \pm 1$

- Mỗi phần tử trên bảng tam giác tương ứng với hàng i , cột j . Với $m-1 \geq j \geq 0$ và $j \leq i \leq m-1$.
- Chuyển đổi phần tử thuộc hàng i , cột j trong bảng băm tam giác trên sang danh sách kê (mảng 1 chiều) được cho bởi hàm băm:
$$hf(i,j) = (i * ((2*m) - i + 1) / 2) + (j - i) :$$
- HashTable tam giác trên được mô tả bởi một danh sách kê:

11. TRẮC NGHIỆM

1. Nhập vào bảng băm các giá trị sau: 4322, 1334, 1471, 9679, 1989, 6171, 6173, 4199 với hàm băm được sử dụng là $x \bmod 10$. Có các ý kiến sau đây:

i. 9679, 1989, 4199 băm thành cùng một giá trị

ii. 1471, 6171 băm thành cùng một giá trị

iii. Tất cả các phần tử băm thành cùng một giá trị

iv. Mỗi phần tử băm thành một giá trị khác nhau

Chọn phát biểu đúng nhất trong các phát biểu sau:

(A) i đúng

(B) ii đúng

(C) i và ii đúng

(D) iii hoặc iv đúng

11. TRẮC NGHIỆM

2. Nhập các khóa sau 12, 18, 13, 2, 3, 23, 5 và 15 vào bảng băm trống ban đầu có độ dài 10 bằng cách sử dụng địa chỉ mở với hàm băm $h(k) = k \bmod 10$ và sử dụng phương pháp dò tuyến tính để giải quyết xung đột. Bảng băm kết quả là gì?

0	
1	
2	2
3	23
4	
5	15
6	
7	
8	18
9	

(A)

0	
1	
2	12
3	13
4	
5	5
6	
7	
8	18
9	

(B)

0	
1	
2	12
3	13
4	2
5	3
6	23
7	5
8	18
9	15

(C)

0	
1	
2	12, 2
3	13, 3, 23
4	
5	5, 15
6	
7	
8	18
9	

(D)

11. TRẮC NGHIỆM

3. Một bảng băm có độ dài 10 sử dụng địa chỉ mở với hàm băm $h(k) = k \bmod 10$ và thăm dò tuyến tính. Sau khi chèn 6 giá trị vào một bảng băm trống, bảng sẽ như hình dưới đây.

Lựa chọn nào sau đây cung cấp thứ tự có thể có trong đó các giá trị khóa có thể được chèn vào bảng?

- (A) 46, 42, 34, 52, 23, 33
- (B) 34, 42, 23, 52, 33, 46
- (C) 46, 34, 42, 23, 52, 33
- (D) 42, 46, 33, 23, 34, 52

0	
1	
2	42
3	23
4	34
5	52
6	46
7	33
8	
9	

11. TRẮC NGHIỆM

4. Với dữ liệu tương tự như câu 3. Cho biết có bao nhiêu trình tự chèn khác nhau để chèn các số 46, 34, 42, 23, 52, 33 để có bảng băm như hình vẽ?

(A) 10

(B) 20

(C) 30

(D) 40

0	
1	
2	42
3	23
4	34
5	52
6	46
7	33
8	
9	

11. TRẮC NGHIỆM

4. Với dữ liệu tương tự như câu 3. Cho biết có bao nhiêu trình tự chèn khác nhau để chèn các số 46, 34, 42, 23, 52, 33 để có bảng băm như hình vẽ?

(A) 10

(B) 20

(C) 30

(D) 40

11. TRẮC NGHIỆM

5. Hàm băm nào sau đây trên số nguyên sẽ phân phối có 10 khóa có giá trị từ 0 đến 9 không bị đụng độ?

(A) $h(i) = (11 * i^2) \bmod 10$

(B) $h(i) = (12 * i) \bmod 10$?

(C) $h(i) = i^2 \bmod 10$

(D) $h(i) = i^3 \bmod 10$

12. BÀI TẬP

1. Cho bảng băm có tập khóa K là số tự nhiên, tập địa chỉ có 10 địa chỉ ($M=10$), chọn hàm băm $hf(key)=key \% 10$. Thực hiện thêm các node **21, 2, 4, 16, 89, 51, 35, 91, 8, 62** vào bảng băm. Yêu cầu lần lượt tổ chức theo 5 phương pháp:
 - a. Bảng băm với phương pháp kết nối trực tiếp (*Direct chaining method*)
 - b. Bảng băm với phương pháp kết nối hợp nhất (*Coalesced chaining method*)
 - c. Bảng băm với phương pháp dò tuyến tính (*Linear Probing method*)
 - d. Bảng băm với phương pháp dò bậc hai (*Quadratic Probing method*)
 - e. Bảng băm với phương pháp băm kép (*Double Hashing method*).
Với $f2 = key \% (M-3)$

12. BÀI TẬP

2. Cho danh sách các tỉnh/thành phố thuộc miền Nam của Việt Nam (Nam bộ). Biết rằng các tỉnh/thành phố này được chia thành 2 tiểu vùng chính là:

- **Vùng Đông Nam Bộ có 5 tỉnh và 1 thành phố:** 5 tỉnh: *Bình Phước, Bình Dương, Đồng Nai, Tây Ninh, Bà Rịa-Vũng Tàu* và *Thành phố Hồ Chí Minh*.
- **Vùng đồng bằng sông Cửu Long, còn gọi là Tây Nam Bộ hay miền Tây,** có 12 tỉnh và 1 thành phố: 12 tỉnh: *Long An, Đồng Tháp, Tiền Giang, An Giang, Bến Tre, Vĩnh Long, Trà Vinh, Hậu Giang, Kiên Giang, Sóc Trăng, Bạc Liêu, Cà Mau* và *Thành phố Cần Thơ*
- **Yêu cầu:** Sinh viên lần lượt tự đề xuất cách tổ chức bảng băm, hàm băm để quản lý 17 tỉnh thành này theo các phương pháp sau:
 - a. Bảng băm với phương pháp kết nối trực tiếp (*Direct chaining method*)
 - b. Bảng băm với phương pháp kết nối hợp nhất (*Coalesced chaining method*)
 - c. Bảng băm với phương pháp dò tuyến tính (*Linear Probing method*)
 - d. Bảng băm với phương pháp dò bậc hai (*Quadratic Probing method*)
 - e. Bảng băm với phương pháp băm kép (*Double Hashing method*).

12. BÀI TẬP

3. Đưa ra nội dung của bảng băm có kết quả khi bạn chèn các mục bằng các phím **D, E, M, O, C, R, A, T** theo thứ tự đó vào một bảng trống ban đầu gồm các danh sách **M=5**, sử dụng các chuỗi riêng biệt với các danh sách không có thứ tự. Sử dụng hàm băm $(11*k)\%M$ để chuyển đổi chữ cái thứ k của bảng chữ cái thành chỉ mục bảng, ví dụ: hàm băm của ký tự $hf(I) = hash(8) = 88\%5 = 3$. (Vị trí các ký tự được tính từ A=0, B=1, ...)
4. Tương tự, đưa ra nội dung của bảng băm có kết quả khi chèn các mục bằng các phím **R, E, P, U, B, L, I, C, A, N** theo thứ tự đó vào một bảng trống ban đầu có kích thước **M=16** bằng cách sử dụng phương pháp dò tuyến tính. Sử dụng hàm băm $(9*k)\%M$ để chuyển đổi chữ thứ k của bảng chữ cái thành một chỉ mục bảng.

12. BÀI TẬP

5. Đưa ra nội dung của bảng băm có kết quả khi bạn chen các mục bằng các phím **A N O T H E R X M P L** theo thứ tự đó vào một bảng trống ban đầu có kích thước $M=16$ bằng cách sử dụng băm kép. Sử dụng hàm băm thứ nhất là $(11*k)\%M$ cho đầu dò băm sinh và hàm băm thứ hai $(k\%3) + 1$ cho mức tăng tìm kiếm.

