CHƯƠNG VI

BẢNG BĂM (Hash Tables)

Nội dung chính

5.1. Tổng quan về bảng băm

5.2. Phương pháp xây dựng hàm băm

5.3. Các phương pháp giải quyết đụng độ

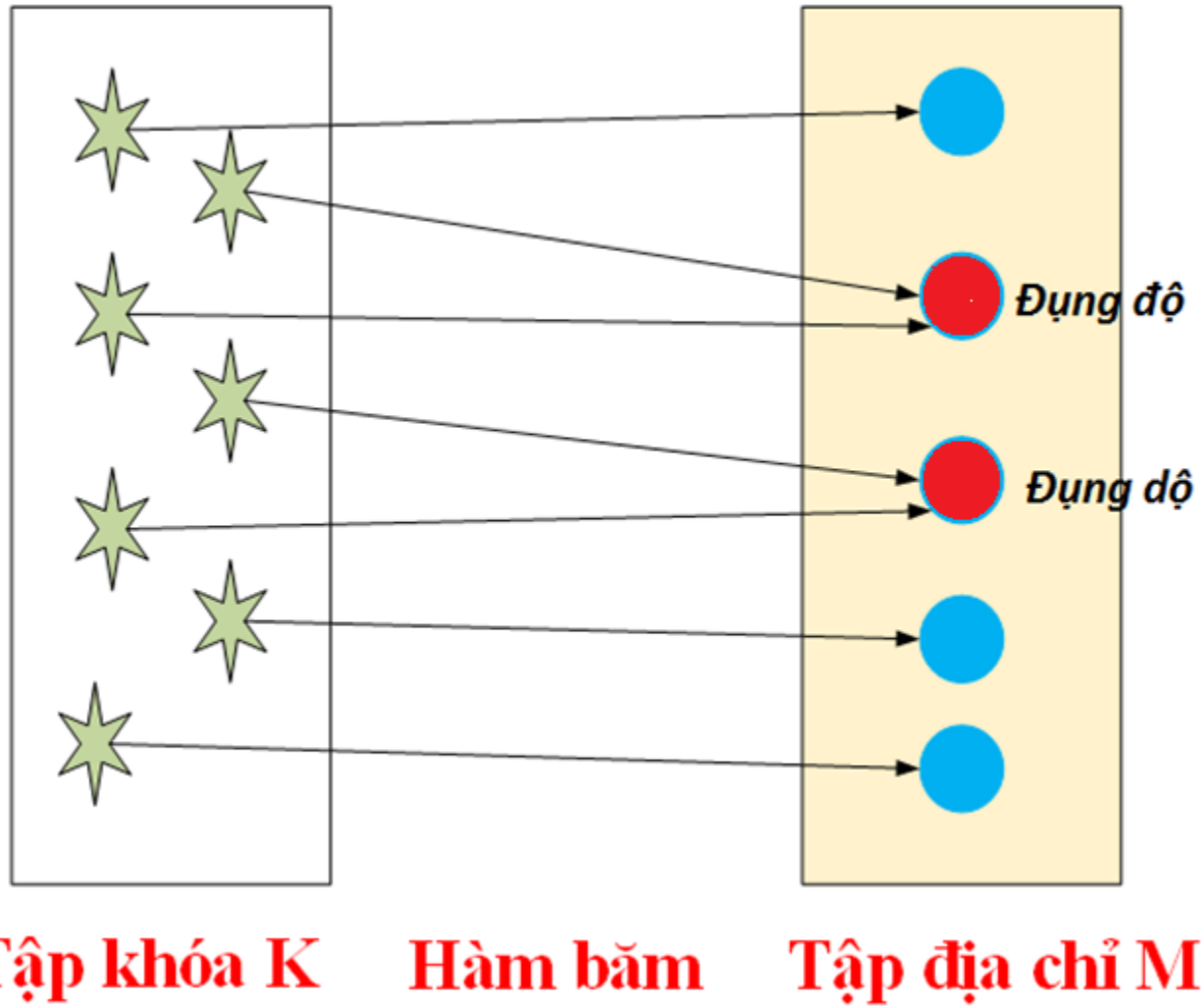
5.1 Tổng quan về bảng băm

- Tư tưởng của phép băm là dựa vào giá trị các khóa k[1..n], chia các khóa đó thành các nhóm.
- Những khóa cùng một nhóm sẽ có cùng một đặc điểm chung và đặc điểm này không có trong nhóm khác.
- Khi có một khóa tìm kiếm X, ta xác định xem nếu X có trong dãy khóa thì nó thuộc nhóm nào và ta tiến hành tìm trên nhóm đó.

Một số khái niệm

- **Bảng băm** là 1 CTDL tương tự như mảng nhưng kèm theo một hàm băm để ánh xạ nhiều giá trị vào cùng một phần tử trong mảng.
- **Hàm băm** hay phép băm (hash function) là hàm được dùng để ánh xạ (hoặc biến đổi) giá trị của khóa vào một dãy các địa chỉ của bảng băm.
- **Khóa** là những giá trị không trùng nhau, có thể là dạng số hay số dạng chuỗi.
- Giả sử có 2 khóa phân biệt k_i và k_j nếu $h(k_i) = h(k_j)$ thì ta nói **hàm băm bị đụng độ** (xung đột) *(với h(x) là hàm băm)*.

Minh họa



Minh họa

Ví dụ minh họa hàm băm đơn giản

- k: là khóa các số nguyên, tập k={34, 58, 21,78, 15}.
- M: kích thước bảng băm (tableSize), M =10.
- h(k)=k% M ={4, 8, 1, 8, 5}.

0	
1	21
2	
3	
4	34
5	15
6	
7	
8	58, 78 (đụng độ)
9	

Ưu điểm của bảng băm

- Bảng băm là một cấu trúc dung hòa giữa thời gian truy xuất và dung lượng bộ nhớ:
  - *Nếu không có sự giới hạn về bộ nhớ thì chúng ta có thể xây dựng bảng băm với mỗi khóa 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 khóa có cùng địa chỉ, khi đó tốc độ truy xuất sẽ giảm.*
- 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.

Các phép toán trên bảng băm

- Khởi tạo (Initialize).
- Kiểm tra rỗng (Empty).
- Lấy kích thước của bảng băm (Size).
- **Tìm kiếm (Search).**
- **Thêm mới phần tử (Insert).**
- **Loại bỏ (Remove).**
- Sao chép (Copy).
- Duyệt (Traverse).

5.2 Phương pháp xây dựng hàm băm

- **Một hàm băm tốt phải thỏa mãn các điều kiện sau:**
  - *Tính toán nhanh.*
  - *Các khoá được phân bố đều trong bảng.*
  - *Ít xảy ra đụng độ.*
- **Một số phương pháp xây dựng hàm băm:**
  - *Hàm băm dạng bảng tra.*
  - *Phương pháp chia.*
  - *Phương pháp nhân.*

Hàm băm dạng bảng tra

- Hàm băm có thể tổ chức ở dạng bảng tra hoặc thông dụng nhất là ở dạng công thức.
- Ví dụ: Bảng tra với khóa là bộ chữ cái, bảng băm có 26 địa chỉ từ 0 đến 25. Khóa a ứng với địa chỉ 0, khoá b ứng với địa chỉ 1,… , z ứng với địa chỉ 25.

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

Phương pháp chia

$$h(k) = k \% m$$

- **m** (số nguyên ≤ M kích thước của bảng). Thông thường nên chọn m là số nguyên tố.
- **k** là khóa.
- Giá trị của **h(k)** sẽ là: ***0, 1, 2, 3,…, m-1***.

- Ví dụ: Tập khoá là các giá trị số gồm 3 chữ số, và vùng nhớ cho bảng địa chỉ có khoảng 100 mục, như vậy ta sẽ lấy hai số cuối của khoá để làm địa chỉ theo phép chia lấy dư cho 100 : chẳng hạn 325 % 100 = 25

Phương pháp chia

- Tuy nhiên ta nhận thấy nếu hàm băm dùng công thức như trên thì địa chỉ của khoá tính được chỉ căn cứ và 2 ký số cuối. Vì thế, để hàm băm có thể tính địa chỉ khoá một cách "ngẫu nhiên" hơn ta nên chọn m=97 thay vì 100.

m=100	
Khoá	Địa chỉ
325	25
125	25
147	47

m=97 (nguyên tố)	
Khoá	Địa chỉ
325	34
125	28
147	50

Phương pháp nhân

$$h(k) = |\ m*(k*A\ \%\ 1)\ |$$

- **k** là khóa, **m** ≤ M là kích thước của bảng – *thường chọn* $m = 2^n$, A là hằng số và 0<A<1.
- Theo Knuth chọn $A = (\sqrt{5}-1)/2 = 0.6180339887$ *là tốt.*

…

m=100, A=0.61803	
Khoá	Địa chỉ
325	86
125	25
147	85

5.3 Phương pháp giải quyết đụng độ

- *Phương pháp kết nối*
- *Phương pháp dò tuần tự*
- *Phương pháp dò bậc hai*
- *Phương pháp băm kép*

Phương pháp kết nối (Chaining Method)

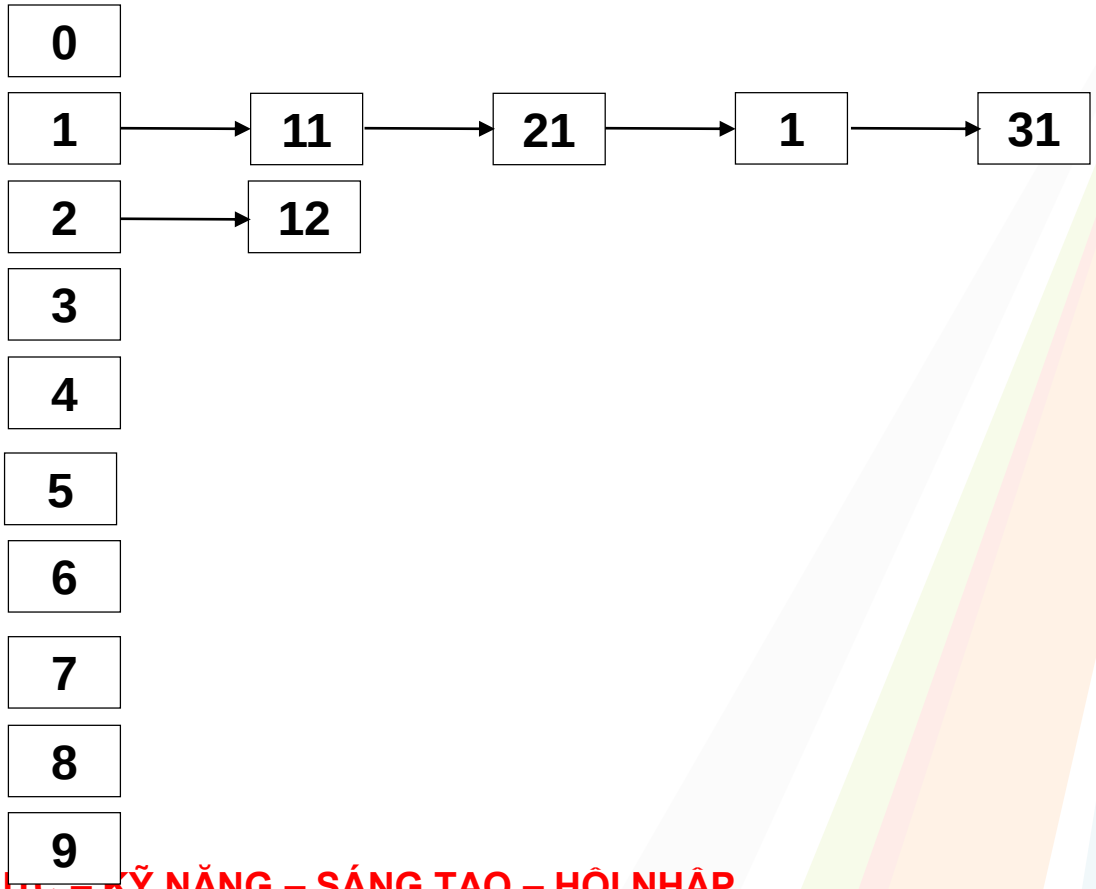
Phương pháp kết nối trực tiếp (Direct chaining)

- Bảng băm được cài đặt bằng các danh sách liên kết, các phần tử trên bảng băm được "băm" thành M danh sách liên kết (từ danh sách 0 đến danh sách M–1).
- Các phần tử bị xung đột tại địa chỉ i được kết nối trực tiếp với nhau qua danh sách liên kết i.

Ví dụ minh họa pp kết nối trực tiếp

- Tập khóa k = {11, 12, 21, 1, 31}, tập địa chỉ M = {0, 1, …, 9}, Hàm băm h(k) = k % 10.

k	h(k)
11	1
12	2
21	1
1	1
31	1



Phương pháp kết nối

Phương pháp kết nối hợp nhất (Coalesced chaining)

- Bảng băm trong trường hợp này được cài đặt bằng danh sách liên kết dùng mảng, có M phần tử.
- Các phần tử bị xung đột tại một địa chỉ được kết nối với nhau qua một danh sách liên kết.
- Mỗi phần tử của bảng băm gồm hai trường:
  - Trường key: chứa khóa của mỗi phần tử.
  - Trường next: con trỏ chỉ đến phần tử kế tiếp nếu có xung đột.
- Khởi động: Khi khởi động, tất cả trường key của các phần tử trong bảng băm được gán bởi giá trị NullKey, còn tất cả các trường next được gán –1.

Phương pháp kết nối

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

Ví dụ minh họa pp kết nối hợp nhất

- Ví dụ:
  - Tập khóa k = {11, 12, 21, 1, 31}
  - Tập địa chỉ M = {0, 1, …, 9}
  - Hàm băm h(k) = k % 10.

k	h(k)
11	1
12	2
21	1
1	1
31	1

	key	Next
0	Null	-1
1	Null	-1
2	Null	-1
…	Null	-1
7	Null	-1
8	Null	-1
9	Null	-1

	key	Next
0	Null	-1
1	11	-1
2	12	-1
…	Null	-1
7	Null	-1
8	Null	-1
9	Null	-1

	key	Next
0	Null	-1
1	11	9
2	12	-1
…	Null	-1
7	Null	-1
8	Null	-1
9	21	-1

	key	Next
0	Null	-1
1	11	9
2	12	-1
…	Null	-1
7	31	-1
8	1	7
9	21	8

Phương pháp dò tuyến tính (Linear Probe)

- Bảng băm trong trường hợp này được cài đặt bằng danh sách kề có M phần tử (kích thước bảng băm), mỗi phần tử của bảng băm là một mẫu tin có một trường key để chứa khoá của phần tử (ban đầu khởi tạo bằng -1).
- Khi thêm phần tử có khoá key vào bảng băm, hàm băm h(key) sẽ xác định địa chỉ i trong khoảng từ 0 đến M-1:
  - *Nếu chưa bị xung đột thì thêm phần tử mới vào địa chỉ này.*
  - *Nếu bị xung đột thì hàm băm lại lần 1, hàm h_1 sẽ xét địa chỉ kế tiếp, nếu lại bị xung đột thì hàm băm lại lần 2, hàm h_2 sẽ xét địa chỉ kế tiếp nữa, ..., và quá trình cứ thế cho đến khi nào tìm được địa chỉ trống và thêm phần tử mới vào địa chỉ này.*

Phương pháp dò tuyến tính

- Khi tìm một phần tử có khoá key trong bảng băm, hàm băm h(key) sẽ xác định địa chỉ i trong khoảng từ 0 đến M-1, tìm phần tử khoá key trong bảng băm xuất phát từ địa chỉ i.
- Hàm băm lại lần i được biểu diễn bằng công thức sau:

$$h_i(key) = (h(key)+i) \% M$$

với h(key) là hàm băm chính của bảng băm.

- *Lưu ý:*
  - *Địa chỉ dò tìm kế tiếp là địa chỉ 0 nếu đã dò đến cuối bảng.*
  - *Hàm băm lần thứ i ta sẽ % cho M (kích thước bảng băm).*

Ví dụ

k={76,93,40,47,10,55}
M = {0,1, …, 6}
h(k)=k%7

k	h(k)
76	76%7=6
93	93%7=2
40	40%7=5
47	47%7=5
10	10%7=3
55	55%7=6

h = 47%7=5
h_1=(5+1)%7=6
h_2=(5+2)%7=0

h=55%7=6
h_1=(6+1)%7=0
h_2=(6+2)%7=1

76 ⇨ 6

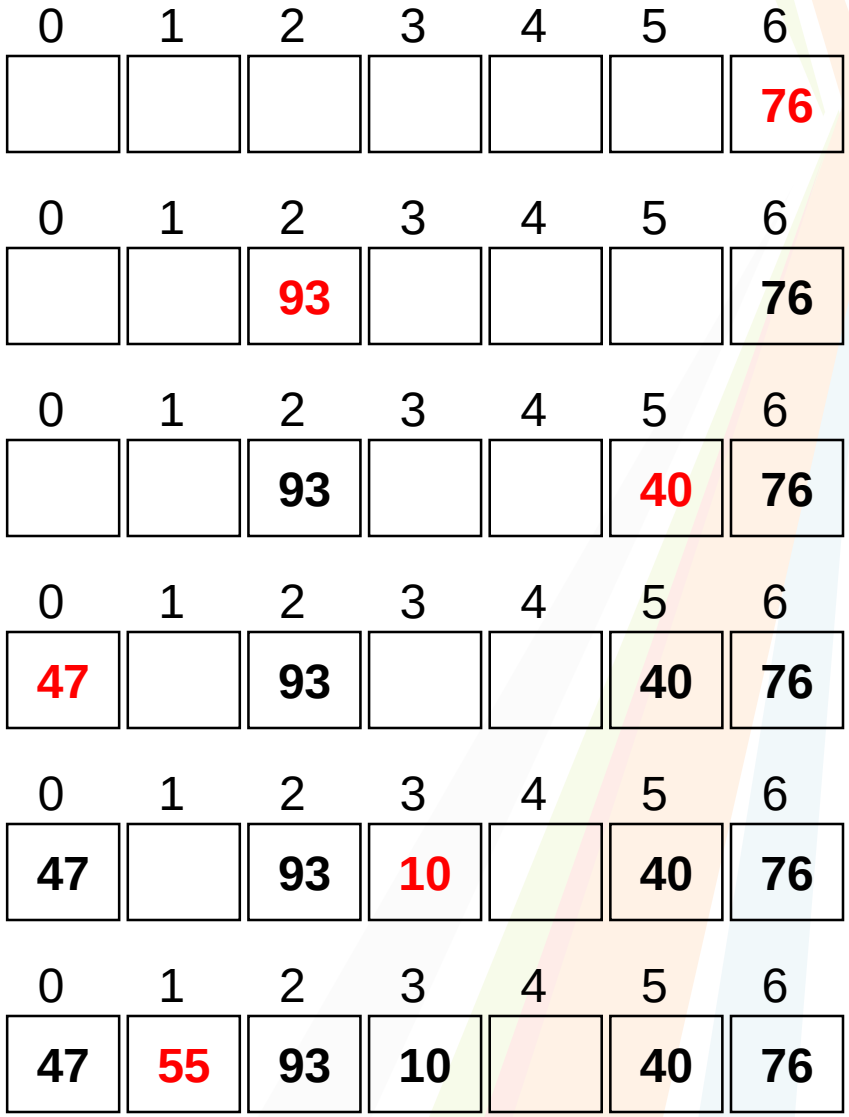
93 ⇨ 2

40 ⇨ 5

47 ⇨ 5
47 ⇨ 6
47 ⇨ 0

10 ⇨ 3

55 ⇨ 6
55 ⇨ 0
55 ⇨ 1



P/P dò bậc hai (Quadratic Probing)

- Tương tự dò tuần tự nhưng hàm băm lại lần thứ i được biểu diễn bằng công thức sau:

$$h_i(key) = (h(key) + i^2) \% M$$

với h(key) là hàm băm chính của bảng băm.

Ví dụ

k={76,40,48,5,55}
M = {0,1, …, 6}
Hàm băm h(k)=k%7

k	h(k)
76	76%7=6
40	40%7=5
48	48%7=6
5	5%7=5
55	55%7=6

76 ⇨ 6

0	1	2	3	4	5	6
						76

40 ⇨ 5

0	1	2	3	4	5	6
					40	76

h=48%7=6
h_1=(6+1^2)%7=0

48 ⇨ 6
48 ⇨ 0

0	1	2	3	4	5	6
48					40	76

h=5%7=5
h_1=(5+1^2)%7=6
h_2=(5+2^2)%7=2

5 ⇨ 5
5 ⇨ 6
5 ⇨ 2

0	1	2	3	4	5	6
48		5			40	76

h=55%7=6
h_1=(6+1^2)%7=0
h_2=(6+2^2)%7=3

55 ⇨ 6
55 ⇨ 0
55 ⇨ 3

0	1	2	3	4	5	6
48		5	55		40	76

P/P băm kép (Double hashing)

- Phương pháp băm kép dùng thêm 1 hàm băm thứ hai, thông thường người ta chọn hàm băm thứ 2 như sau:

  Hàm băm 1: *h(key) = key % m*

  Hàm băm 2: thường ở 1 trong 2 dạng sau:

  d(key) = key% p +1

  hay d(key) = p - key% p *(Trong đó* ***m, p <= M)***

- Khi thêm phần tử có khoá key vào bảng băm theo hàm h:
  - *Nếu bị xung đột thì xét địa chỉ mới (h+d)%M.*
  - *Nếu bị xung đột lần 2 thì mới (h+2*d)%M.*
  - *...*
  - *Nếu bị xung đột lần i thì xét địa chỉ:* ***(h+i*d)%M***

Ví dụ

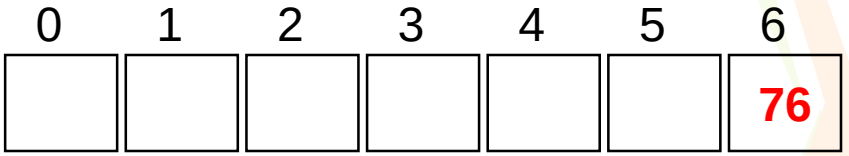
k={76,93,40,47,10,55}
M = {0,1, …, 6}
h(k)=k%7
d(k)=5-(k%5)

k	h(k)	d(k)
76	6	4
93	2	2
40	5	5
47	5	3
10	3	5
55	6	5

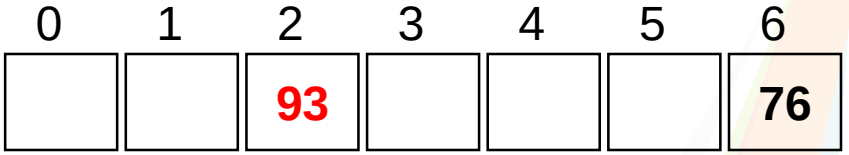
h=5, d=3
(h+d)%7=1

h=6, d=5
(h+d)%7=4

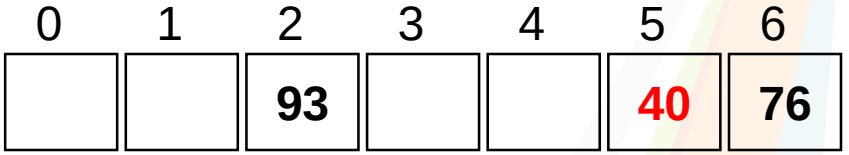
76 ⇨ 6



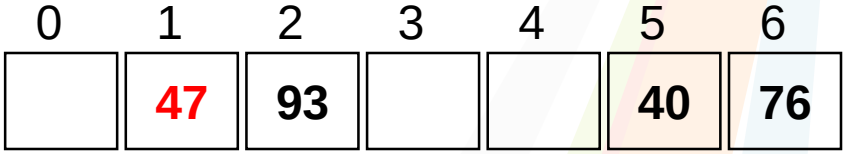
93 ⇨ 2



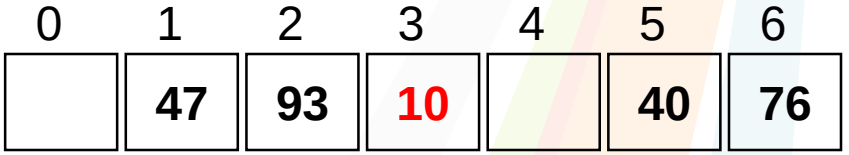
40 ⇨ 5



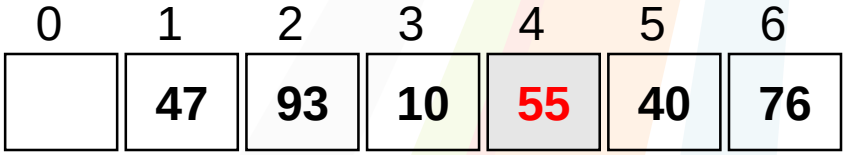
47 ⇨ 5
47 ⇨ 1



10 ⇨ 3



55 ⇨ 5
55 ⇨ 4



Ví dụ

k={18,41,22,
44,59,32,31,73}
M={0, 1,..,12}
h(k)=k%13
d(k)=7-(k%7)

k	h(k)	d(k)
18	5	3
41	2	1
22	9	6
44	5	5
59	7	4
32	6	3
31	5	4
73	8	4

h=5,d=5
(5+5)%13=10

h=5,d=4
(5+4)%13=9
(5+8)%13=0

	0	1	2	3	4	5	6	7	8	9	10	11	12
18⇨5						18							
41⇨2			41			18							
22⇨9			41			18				22			
44⇨5 44⇨10			41			18				22	44		
59⇨7			41			18		59		22	44		
32⇨6			41			18	32	59		22	44		
31⇨5 31⇨9 31⇨0	31		41			18	32	59		22	44		
73⇨8	31		41			18	32	59	73	22	44		

Ví dụ

k={14,17,25,37,34,16,26}
M = {0, 1, …, 10}
h(k)=k%11
d(k)=k%7+1

k	h(k)	d(k)
14	3	1
17	6	4
25	3	5
37	4	3
34	1	7
16	5	3
26	4	6

h=3, d=5
(3+5)%11=8

h=4, d=6
(4+6)%11=10

	0	1	2	3	4	5	6	7	8	9	10
14 ⇨ 3				14							
17 ⇨ 6				14			17				
25 ⇨ 8				14			17		25		
37 ⇨ 4				14	37		17		25		
34 ⇨ 1		34		14	37		17		25		
16 ⇨ 5		34		14	37	16	17		25		
26 ⇨ 4 26 ⇨ 10		34		14	37	16	17		25		26

Lưu ý

Đối với dữ liệu lớn:

- Việc cấp phát một mảng quá lớn:
  - Lãng phí bộ nhớ không đáng có.
  - Nhưng đụng độ ít xảy ra do các khóa phân bố đều.
- Ngược lại:
  - Để tiết kiệm bộ nhớ.
  - Giảm hiệu suất của bảng băm do việc đụng độ xảy ra với tần suất cao hơn.

Nên khi thao tác với bảng băm, cần phải cân nhắc giữa hiệu suất và dung lượng lưu trữ.


Thank You!