Bảng băm
(Hash Tables)

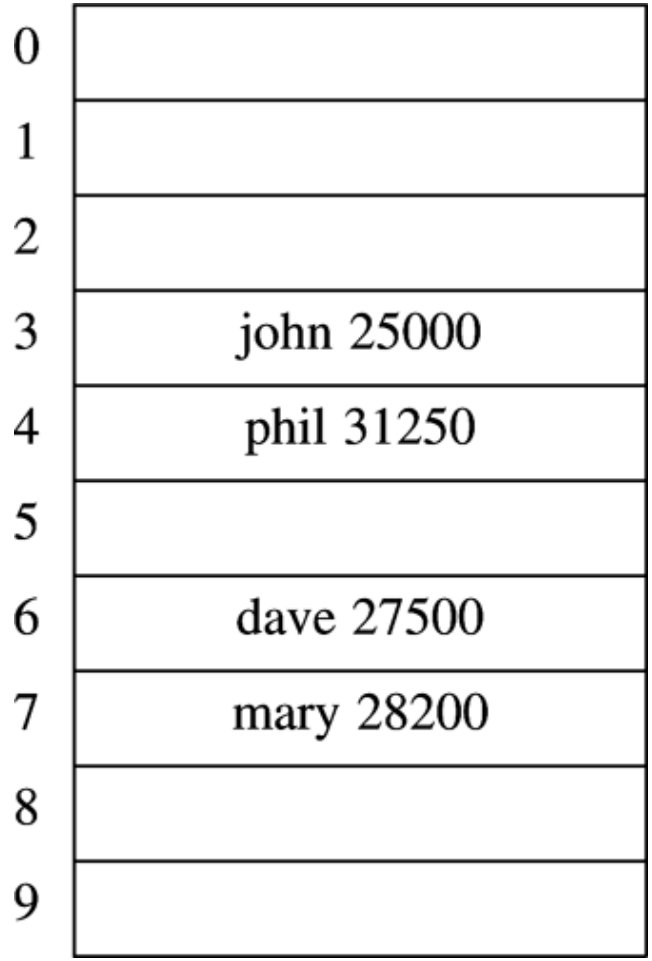
Bài giảng môn Cấu trúc dữ liệu và giải thuật

Khoa Công nghệ thông tin
Trường Đại học Thủy Lợi

Bảng băm

- Các phần tử dưới dạng cặp khóa-giá trị (key-value).
- Mỗi phần tử được lưu trữ vào một ô nào đó trong mảng tùy theo khóa của nó là gì.
- Thực hiện các phép tìm/chèn/xóa trong thời gian O(1).
- Không hiệu quả với các thao tác đòi hỏi thông tin thứ tự:
  - VD: Tìm phần tử lớn nhất và nhỏ nhất.

Ví dụ bảng băm



Mỗi phần tử là một cặp khóa-giá trị:

- Tên là khóa
- Thu nhập là giá trị

So sánh các cấu trúc dữ liệu

- So sánh thời gian tìm kiếm:
  - Vector và danh sách liên kết: O(n)
  - Cây AVL: O(log n)
  - Bảng băm: O(1)

Hàm băm (hash function)

- Ánh xạ khóa sang số nguyên (vị trí trong bảng băm).
- Nếu nhiều khóa ánh xạ sang cùng một số nguyên (cùng vị trí trong bảng băm) thì sẽ dẫn đến **đụng độ**:
  - Đụng độ sẽ giảm nếu các khóa phân bố đồng đều hơn trên bảng băm.
  - Khi đụng độ xảy ra, phải tìm cách phân giải sao cho các phần tử không ghi đè lên nhau.

Một hàm băm đơn giản

- Gọi:
  - key: Khóa có giá trị nguyên.
  - tableSize: Kích thước bảng băm.
- Một hàm băm đơn giản dùng phép chia lấy phần dư:

  hash(key) = key % tableSize
- Giả sử:
  - key = 24, 48, 51, 78, 15
  - tableSize = 10
- Thế thì: key % tableSize = 4, 8, 1, 8, 5
- Để giảm đụng độ, ta thường chọn kích thước bảng là một số nguyên tố.

Một hàm băm cho các xâu ký tự

```
int hash(const string & key, int tableSize) {
    int hashVal = 0;
    for (int i = 0; i < key.size(); i++)
        hashVal += key[i];
    return hashVal % tableSize;
}
```

- Ví dụ:
  - tableSize = 100
  - key = "ABC" (mã ASCII của A, B, C là 65, 66, 67)
  - hashVal = (65 + 66 + 67) % 100 = 198 % 100 = 98
  - Nếu key = "CBA" thì hashVal = ?
- Một hàm băm tốt hơn cho xâu ký tự key = $x_0x_1...x_{k-2}x_{k-1}$ như sau:

  $$\text{hash(key)} = (x_0 * a^{k-1} + x_1 * a^{k-2} + \cdots + x_{k-2} * a + x_{k-1}) \% \text{ tableSize}$$

  - Nếu xâu ký tự là từ tiếng Anh, có thể chọn a = 33, 37, 39 hoặc 41.

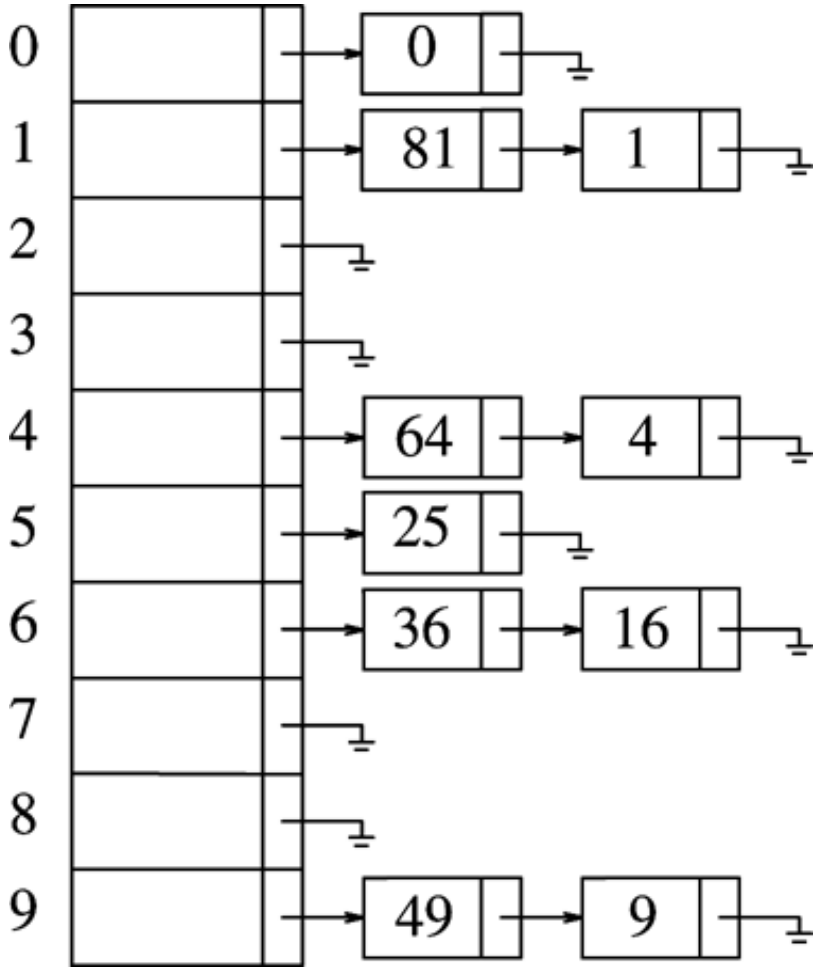
Thiết kế bảng băm

1. Hàm băm: Ánh xạ khóa sang một vị trí trong bảng băm.
   - VD: hash(key) = key % tableSize

2. Phân giải đụng độ:
   - Giải quyết trường hợp nhiều khóa ánh xạ đến cùng một vị trí.
   - Hai giải pháp thường gặp:
     - Dây chuyền (separate chaining)
     - Thăm dò (probing)

Giải pháp dây chuyền

- Mỗi ô trong mảng giữ một danh sách liên kết các phần tử (dây chuyền).
- Các phần tử có khóa ánh xạ tới cùng một ô được giữ trong cùng một danh sách liên kết.
- Ví dụ:
  - hash(key) = key % 10
  - Chèn 10 số chính phương đầu tiên vào bảng băm (hình bên).



Phân tích giải pháp dây chuyền

- Xét bảng băm có m ô và chứa n phần tử.
- Chèn không kiểm tra tính duy nhất mất thời gian O(1), vì gồm 2 bước:
  1. Tìm vị trí chèn dùng hàm băm: O(1)
  2. Gọi pushFront của danh sách liên kết: O(1)
- Tìm/xóa/chèn có kiểm tra tính duy nhất, trường hợp tồi nhất (phải quét hết danh sách liên kết có n phần tử): O(n)
- Tìm/xóa/chèn có kiểm tra tính duy nhất, trường hợp trung bình: O(n/m)
  - Ta sẽ tăng kích thước bảng nếu số phần tử trung bình trong một ô (n/m) vượt quá hệ số tải λ (≤ 1).
  - Do đó, thời gian trung bình = $O(\lambda) = O(1)$.

Bảng băm thăm dò

- Bảng băm dây chuyên phức tạp do phải duy trì một danh sách liên kết ở mỗi ô.
- Giải pháp thăm dò ô trống:
  - Nếu đụng độ xảy ra, thử các ô khác trong bảng.
  - Thử lần lượt các ô $h_0(x)$, $h_1(x)$, $h_2(x)$, $h_3(x)$… cho đến khi tìm được một ô trống:
    - $h_i(x)$ = [hash(x) + f(i)] % tableSize
    - f(0) = 0 (vì ta bắt đầu thăm dò từ vị trí thu được sau khi áp dụng phép băm)

Thăm dò tuyến tính

$h_i(x) = [hash(x) + f(i)] \% tableSize$, trong đó **f(i) = i**

- **Phép chèn** (giả sử các khóa không trùng nhau):
  1. index = hash(x);
  2. Nếu table[index] rỗng, đặt phần tử mới (gồm khóa và giá trị) vào ô table[index];
  3. Nếu table[index] không rỗng:
     - index++; index = index % tableSize;
     - Quay lại bước 2;
- **Tìm kiếm:**
  1. index = hash(x);
  2. Nếu table[index] rỗng, trả về -1 (không tìm thấy);
  3. Nếu table[index].key == x, trả về index (tìm thấy);
  4. index++; index = index % tableSize; quay lại bước 2;

Ví dụ

Chèn 89, 18, 49, 58, 69 (hash(x) = x % 10)

	Empty Table	After 89	After 18	After 49	After 58	After 69
0				49	49	49
1					58	58
2						69
3						
4						
5						
6						
7						
8			18	18	18	18
9		89	89	89	89	89

Thăm dò bậc hai

$h_i(x)$ = [hash(x) + f(i)] % tableSize, trong đó **f(i) = i^2**

	Empty Table	After 89	After 18	After 49	After 58	After 69
0				49	49	49
1						
2					58	58
3						69
4						
5						
6						
7						
8			18	18	18	18
9		89	89	89	89	89

Tổ chức lại bảng băm

- Nếu bảng băm đầy, không thể chèn thêm được nữa.
- Nếu bảng băm khá đầy (nhưng chưa đầy 100%), chèn/xóa và tìm kiếm sẽ mất nhiều thời gian hơn.
- Giải pháp là tổ chức lại bảng băm:
  1. Tạo bảng mới có kích thước lớn hơn (VD: gấp hai lần).
  2. Định nghĩa hàm băm mới.
  3. Chuyển các phần tử từ bảng cũ sang bảng mới.
  4. Xóa bảng cũ.
- Chi phí tổ chức lại bảng băm là O(n):
  - Khá tốn kém nhưng xảy ra không thường xuyên.
  - Chỉ xảy ra khi bảng băm vượt quá hệ số tải λ.

Ví dụ tổ chức lại bảng băm thăm dò tuyến tính

1. Bảng băm ban đầu

0	6
1	15
2	
3	24
4	
5	
6	13

2. Chèn thêm 23

0	6
1	15
2	23
3	24
4	
5	
6	13

3. Tổ chức lại bảng băm (giả sử $\lambda = 0.7$)

- Gấp đôi kích thước cũ rồi tìm số nguyên tố kế tiếp (ở đây là 17) để dùng làm kích thước mới.
- Quét bảng băm cũ từ đầu đến cuối để lấy ra các giá trị rồi chèn vào bảng mới theo đúng thứ tự lấy ra.

0	
1	
2	
3	
4	
5	
6	6
7	23
8	24
9	
10	
11	
12	
13	13
14	
15	15
16	

Bài tập

1. Xét bảng băm đang rỗng và có hàm băm là hash(x) = x % 10. Hãy chèn lần lượt vào bảng các giá trị { 4371, 1323, 6173, 4199, 4344, 9679, 1989 } cho mỗi trường hợp sau:

(a) Bảng băm dây chuyền

(b) Bảng băm thăm dò tuyến tính

(c) Bảng băm thăm dò bậc hai

2. Hãy tổ chức lại các bảng băm sau khi chèn trong bài tập 1 sao cho kích thước bảng băm gấp đôi.