# Cấu trúc dữ liệu và giải thuật

**Lecture 6: Bảng băm**

Ta Viet Cuong, Ph.D

HMI laboratory, FIT-UET

# I. Ý tưởng

- Việc sắp xếp, tổ chức lại dữ liệu phép truy xuất vào dữ liệu nhanh hơn.
- Do bị giới hạn bởi phép so sánh và thứ tự của dữ liệu, O(lg N)
- **Có thể giảm xuống** ! nhưng chúng ta phải dùng mô hình tính toán khác thay vì sử dụng cây quyết định.

| Cấu trúc dữ liệu | Tìm kiếm | Chèn | Xoá |
|---|---|---|---|
| Mảng không sắp xếp | $O(N)$ | $O(N)$ | $O(N)$ |
| Mảng được sắp xếp | $O(\lg N)$ | $O(N)$ | $O(N)$ |
| Cây nhị phân cân bằng, cây đỏ đen | $O(\lg N)$ | $O(\lg N)$ | $O(\lg N)$ |

# II. Bảng băm

- **Bảng băm (hash tables)** là cấu trúc dữ liệu cho phép truy xuất các phần tử bằng chỉ số (indexed array - giống như mảng) tuy nhiên, chỉ số của mỗi phần tử được tính toán bằng hàm băm. Bảng băm T có kích thước N được đánh chỉ số các phần tử T[0], T[1], … , T[N-1].
- Các thao tác chính của bảng băm:
  - Chèn vào bảng băm: **Insert(T, h, x): T[h(x)] = x**
  - Tìm kiếm: **Search(T, h, x): return Yes if T[h(x)] = x else return N**
- Các vấn đề của bảng băm:
  - **Tính hàm băm**: làm thế nào để hàm băm phủ tập chỉ số một cách đều đặn cũng như thoả mãn một số tính chất khác (sẽ bàn dưới đây).
  - **Kiểm tra xung đột:** kiểm tra hai khoá có xung đột.
  - **Giải quyết xung đột:** khi hai khoá được băm đến cùng vị trí trong bảng băm, xử lý thế nào.

# III. Hàm băm

- **Hàm băm (hash function)**: cách tính chỉ số của dữ liệu:
  - Gọi x ∈ U là dữ liệu cần lưu trữ, hàm băm: U -> {0, 1, 2, 3, …, N-1} với N là kích thước bảng băm. Khi đó h(x) là vị trí lưu trữ x trong bảng băm. Tức là x được lưu tại phần tử T[h(x)] trong bảng băm T.
  - Như vậy, hàm băm là một ánh xạ từ tập các giá trị khóa của dữ liệu vào các tập số nguyên {0, 1, 2, …, N-1}.
- **Hướng tiếp cận:**
  - Kiểu dữ liệu cơ bản
  - Kiểu dữ liệu tự định nghĩa

# III.3 Hàm băm modulo

- Khi có mã băm của x, lấy modulo của mã băm này với N ta sẽ được chỉ số để truy xuất vào bảng băm.
  - h(x) = hashCode(x) mod N

# Ví dụ

Tìm cách map một key có giá trị bất kì về 1 khoảng cho trước.

Sử dụng hàm h(x): x -> [0..N-1]

Ví dụ: h(x) = x % N

| Key | 0 | 3 | 1001 | 1002 | 1005 |
|---|---|---|---|---|---|
| h(key) | 0 | 3 | 1 | 2 | 5 |
| Value | A | B | B | G | F |

h(x) = x % 1000

# Ví dụ

hash table: bảng lưu giá trị sau khi sử dụng hash funcion

Ví dụ: Hashing keys by a hash function

| Key | 0 | 3 | 1001 | 1002 | 1005 |
|---|---|---|---|---|---|
| h(key) | 0 | 3 | 1 | 2 | 5 |
| Value | A | B | B | G | F |

Hash table

| 0 | 1 | 2 | 3 | | 5 | | | | 1000 |
|---|---|---|---|---|---|---|---|---|---|
| A | B | G | B | | F | | | | |

# III.4 Giả thiết hàm băm đều

- Một hàm băm tốt là hàm băm có xác suất đụng độ nhỏ. Ta xét cách sử dụng hàm băm như sau
  - Chọn ngẫu nhiên một hàm băm h từ họ các hàm băm H.
  - (Giả thiết) Xác suất đụng độ khi sử dụng hàm băm ngẫu nhiên:

$$\mathbf{P}_{h\in H}\{h(x) = h(y) = \frac{1}{N}, \forall x \neq y$$

  - Lưu ý, tính ngẫu nhiên xuất phát từ việc chọn hàm băm chứ không phải từ dữ liệu nhằm tránh bị tấn công (lựa chọn dữ liệu để băm vào cùng một vị trí).

# Today

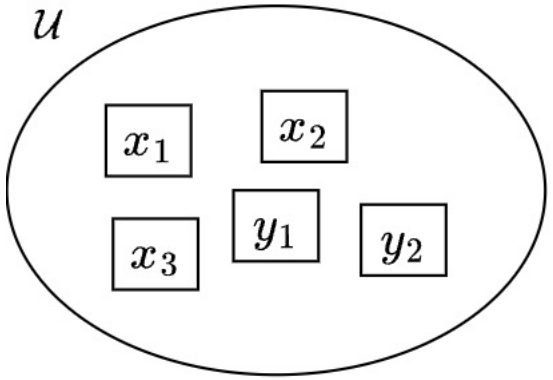
- I. Ý tưởng
- II. Bảng băm
- III. Hàm băm
- IV. Giải quyết đụng độ
- V. Bài tập

# IV. Giải quyết đụng độ

- Xích ngăn cách: Khi hai khoá x, y được băm vào cùng một địa chỉ (đụng độ):

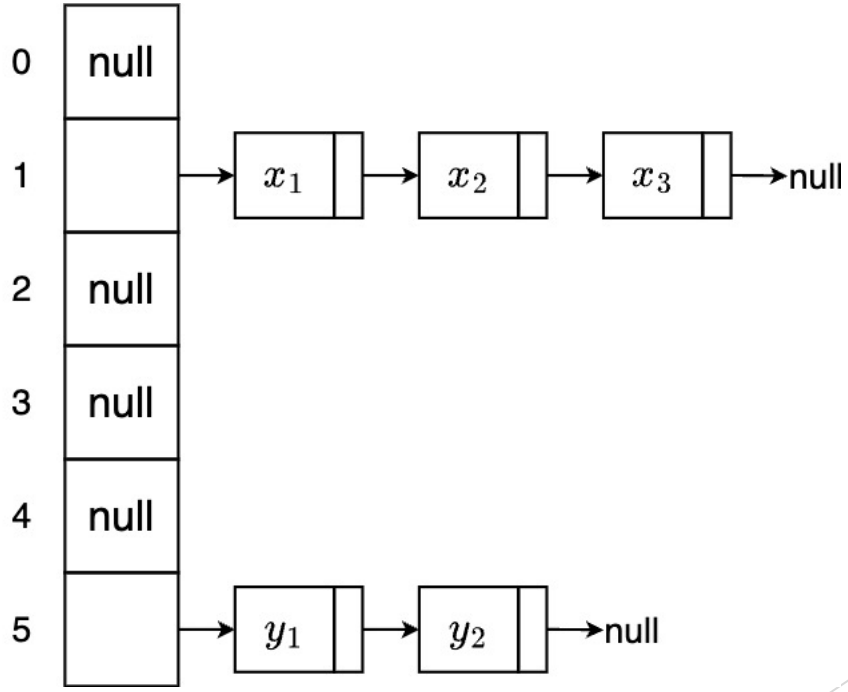
$$i = h(x) = h(y)$$

=> ta đưa cả x và y vào một danh sách liên kết đặt tại vị trí i.




$$1 = h(x_1) = h(x_2) = h(x_3)$$

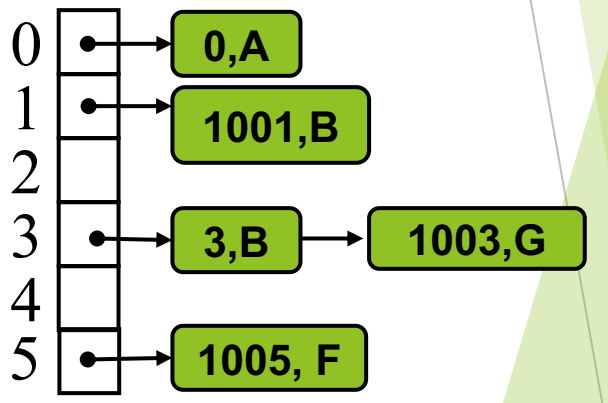
$$5 = h(y_1) = h(y_2)$$



# Ví dụ

| key | 0 | 3 | 1001 | 1003 | 1005 |
|---|---|---|---|---|---|
| h(key) | 0 | 3 | 1 | 3 | 5 |
| Value | A | B | B | G | F |

h(key) = key mod 1000



- Như vậy, bảng băm bản chất là một mảng các danh sách liên kết:

`insert` ($T$: mảng các con trỏ đến đầu các danh sách liên kết, $h$: hàm băm, $x$):

1. $T[h(x)] =$ `push` $(T[h(x)], x)$
2. // TODO: resize

`push` ($first$: Node, $x$):

1. `return new Node` (value = $x$, next = $first$)

`search` $(T, h, x)$:

1. `return` `find` $(T[h(x)], x)$

`find` (n: Node, x):

1. `while` $n \neq null$:
   a. `if` $n.value = x$: `return` $n$
   b. $n = n.next$
2. `return` $null$

`remove` $(T, h, x)$:

1. $T[h(x)] =$ `delete` $(T[h(x)], x)$
2. // TODO: resize

`delete` (n: Node, x):

1. `if` $n = null$: `return` $n$
2. `if` $n.value = x$: `return` $n.next$
3. $p \leftarrow n$
4. `while` $p.next \neq null$:
   a. `if` $p.next.value = x$: $p.next \leftarrow p.next.next$, `break`
   b. $p \leftarrow p.next$
5. `return` $n$

# IV.1 Đánh giá xích ngăn cách

- Ta xét bộ dữ liệu D có M phần tử được băm vào bảng băm T có N vị trí.
  - Độ dài trung bình của xích ngăn cách: xét dữ liệu x∈U, gọi l(x) là chiều dài xích ngăn cách của khoá x (số phần tử y trong bộ dữ liệu được băm vào cùng vị trí với x). Suy ra:

$$\ell(x) = \sum_{y \in D} \mathbb{I}[h(x) = h(y)] \Rightarrow \mathbb{E}[\ell(x)] = \sum_{y \in D} \mathbb{P}\{h(x) = h(y)\} = M/N = \alpha$$

  - Độ dài trung bình của xích ngăn cách tỉ lệ thuận với hệ số tải $\alpha$=M/N.
  - Nếu chọn N = M/4 thì \alpha = 4, suy ra thời gian tìm kiếm, chèn, xoá trung bình là O(1)
- Để đảm bảo N/M nằm trong giới hạn cho phép, ta cần thay đổi kích thước M của bảng băm khi số lượng dữ liệu N vượt quá $\alpha$M.
- **Bài tập**: cài đặt phép resize(N) cho bảng băm (phải băm lại các phần tử đã đưa vào bảng băm).
- Độ dài lớn nhất của xích ngăn cách: trường hợp xấu nhất (bị tấn công), các dữ liệu được băm vào cùng một vị trí thì độ dài lớn nhất của xích là O(M). Việc truy xuất vào bảng băm sẽ rất chậm.

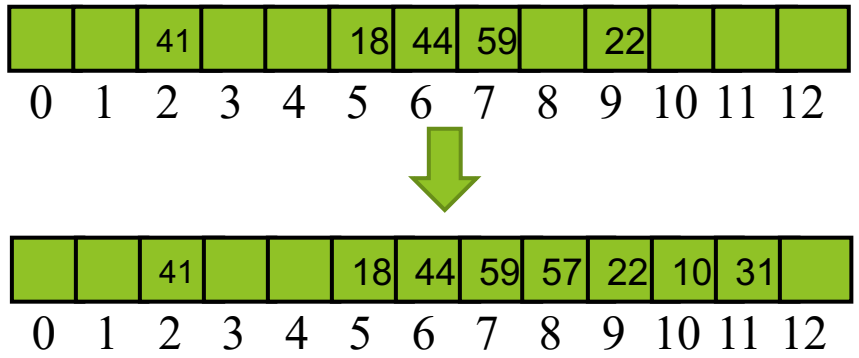
# Định địa chỉ mở

Một số phương pháp định địa chỉ mở:

- Dò tuyến tính
- Dò nhị phân
- Dò bậc hai
- Băm kép (two-probe)
- Băm đúp (double hashing)

# Linear probing – Thăm dò tuyến tính

- Khi thêm vào: 57, 10, 31




- Tìm kiếm x = 13p + 5 -> duyệt qua dãy các số = ?
  - Từ O(1) -> O(N)

- Hàm h(x) có vai trò quan trọng trong giảm collision

# Bài tập

Cho danh sách theo dạng (key, value) được chèn vào hash tables theo thứ tự từ trái qua phải

(4,A), (3,B), (11, C), (8, D) , (12, G), (16, E), (20, B), (5, G)

a) Vẽ hash tables với h(key) = x % 12, sử dụng linked-list và linear probing

b) Vẽ hash tables với h(key) = x % 13, sử dụng linked-list và linear probing

c) Đưa ra giải thích về độ collision của 2 trường hợp