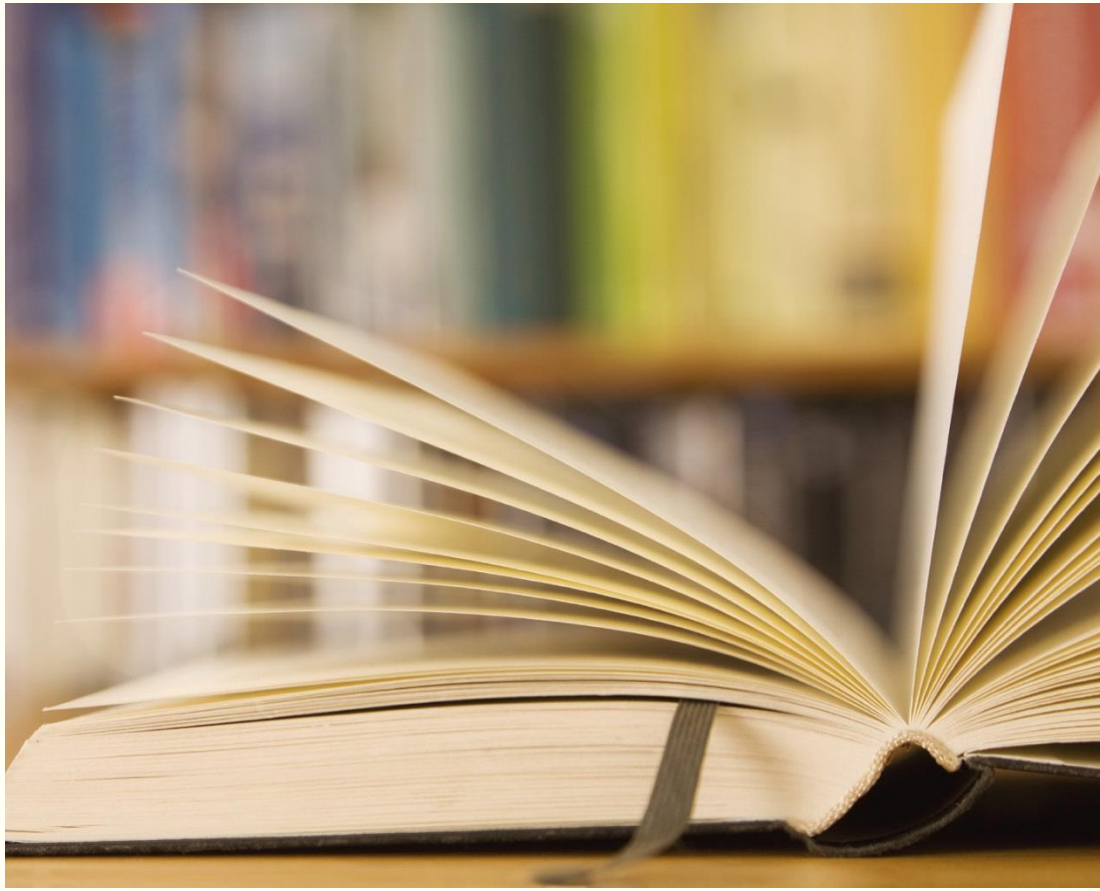
BẢNG BĂM (HASH TABLE)



Mục tiêu

- Tìm hiểu cấu trúc dữ liệu trong đó các phép toán: Thêm, Tìm kiếm, Loại bỏ có độ phức tạp trung bình là O(1).

Nội dung

- Giới thiệu phương pháp băm
- Các hàm băm
- Các chiến lược giải quyết va chạm

Khái niệm

- Hashing là kỹ thuật được sử dụng để lưu trữ và truy xuất thông tin nhanh nhất, độ phức tạp trung bình của cac thao tác thêm, xóa, và tìm kiếm các phần tử O(1).
- Thường được sử dụng trong xây dựng từ điển

Các phép toán

- CreatHashTable: Tạo bảng băm mới
- HashSearch: Tìm kiếm khóa trong bảng băm
- HashInsert: Chèn một khóa mới vào bảng băm
- HashDelete: Xóa khóa khỏi bảng băm
- DeleteHashTable: Xóa bảng băm

Phương pháp băm

- Nếu như các giá trị khoá của các dữ liệu là số nguyên không âm và nằm trong khoảng [0..SIZE-1], chúng ta có thể sử dụng một mảng data có cỡ SIZE để lưu tập dữ liệu đó.
- Trong thực tế, tập hợp các giá trị khóa có thể là vô cùng (hoặc ít nhất là rất lớn). Tạo một mảng lớn và lưu trữ chúng là không thể.
- Kết quả là việc sử dụng các mảng đơn giản không phải là lựa chọn đúng đắn để giải quyết các vấn đề mà các khóa có thể là rất lớn. Quá trình ánh xạ các khóa tới các vị trí được gọi là băm

Phương pháp băm

- Các vấn đề cần giải quyết khi sử dụng
  - Bảng băm (Hash table)
  - Hàm băm (Hash functions)
  - Sự va chạm(Collisions)
  - Kỹ thuật giải quyết va chạm(Collision Resolution Techichques)

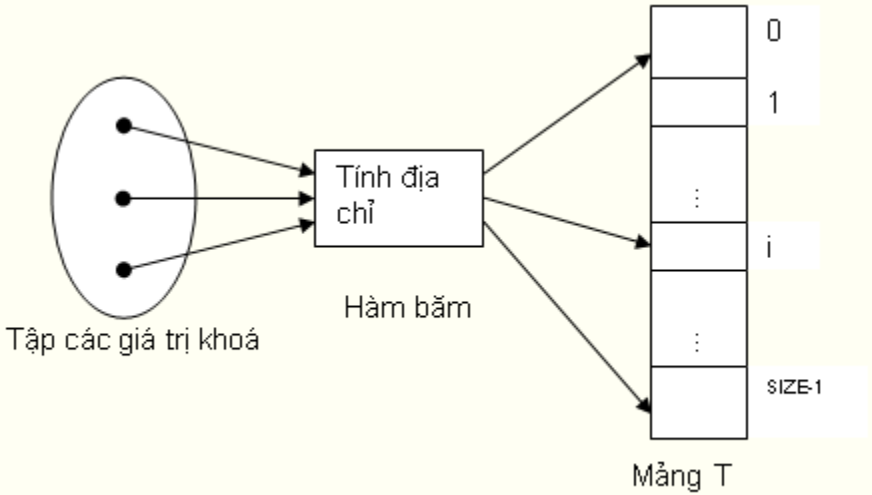
Hash Table

- Bảng băm là một sự tổng quát của mảng. Với một mảng, ta lưu trữ phần tử có khóa là k tại vị trí k của mảng. (Địa chỉ trực tiếp).

- Q: Giải quyết thế nào khi có nhiều khóa hơn số lượng vị trí lưu trữ?
- A: Bảng băm hoặc bản đồ băm là cấu trúc dữ liệu lưu trữ khóa và giá trị được liên kết của chúng và bảng băm sử dụng hàm băm để ánh xạ khóa và giá trị được liên kết với chúng.

Hash function

- Hàm băm được sử dụng để biến khóa thành chỉ mục. Lý tưởng nhất, hàm băm nên ánh xạ mỗi khóa có thể có với một chỉ mục duy nhất. Rất khó để đạt được trong thực tế.
- Mục tiêu tạo ra hàm băm giúp giảm thiểu số lần va chạm, dễ tính toán, và phân phối đều các phần tử trong bảng băm.



Hash function

- Chọn hàm băm như thế nào?
  - Một hàm băm hiệu quả nên được thiết kế sao cho nó phân phối các giá trị chỉ mục các đối tượng được thêm đều trong bảng.
  - Tính toán một chỉ mục thay thế cho một khóa có chỉ mục băm tương ứng với một vị trí được chèn trước đó trong bảng băm.
  - Hàm băm có thể được tính toán một cách nhanh chóng, trả về các giá trị trong phạm vi các vị trí trong bảng và giảm thiểu va chạm.

Hash function

- Hàm băm tốt?
  - Giảm thiểu va chạm
  - Tính toán dễ dàng và nhanh
  - Phân phối các giá trị khóa đồng đều trong bảng băm
  - Sử dụng tất cả thông tin được cung cấp bởi khóa
  - Có hệ số tải cao cho một bộ khóa nhất định
    - Load Factor = Số lượng phần tử trong bảng băm/ Kích thước của bảng băm

Hash function

- Khóa là số nguyên không âm
  - Phương pháp chia
  - Phương pháp nhân
- Khóa là xâu ký tự: đổi xâu thành số nguyên không âm

Hash function

Phương pháp chia

- Phương pháp chia
- h(k) = k mod SIZE
- nhạy cảm với cỡ của bảng băm
- chọn SIZE để hạn chế xảy ra va chạm
- số nguyên tố có dạng đặc biệt, chẳng hạn có dạng 4k+3

Phương pháp nhân

- $h(k) = \lfloor (\alpha k - \lfloor \alpha k \rfloor) . SIZE \rfloor$
- Thực tế thường chọn

$$\alpha = \Phi^{-1} \approx 0{,}61803399$$

Sự va chạm

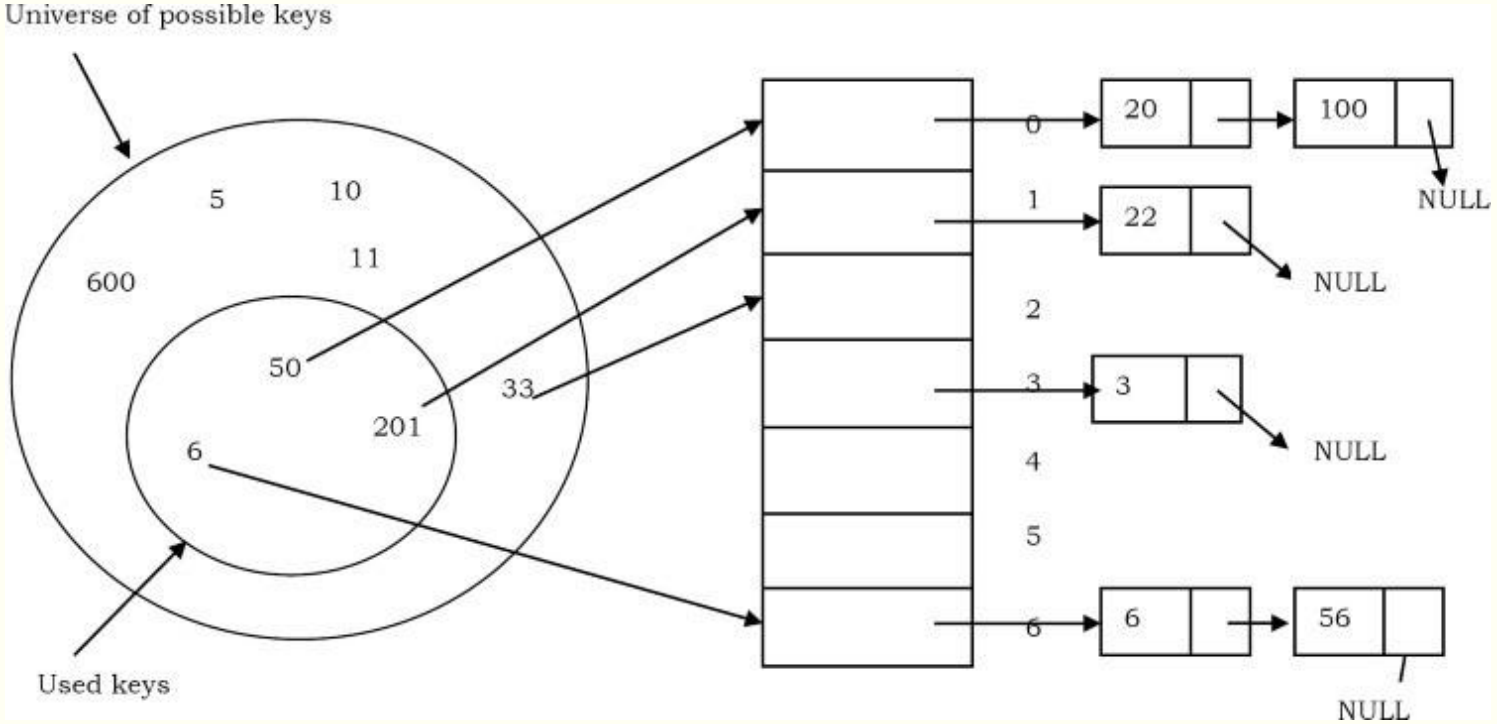
- Va chạm xảy ra khi nhiều bản ghi được lưu trữ cùng một vị trí trong bảng

Một số kỹ thuật giải quyết sự va chạm

- Direct Chaining: (Tạo dây chuyền): Ứng dụng danh sách liên kết
  - ○ Separate chaining
- Open Addressing (Định địa chỉ mở):Dựa trên cấu trúc mảng
  - ○ Linear probing (linear search) (Thăm dò tuyến tính)
  - ○ Quadratic probing (nonlinear search) (Thăm dò bình phương)
  - ○ Double hashing (Băm kép)

Tạo dây chuyền

- Mỗi thành phần của trong bảng băm sẽ chứa một con trỏ trỏ tới danh sách liên kết



Định địa chỉ mở

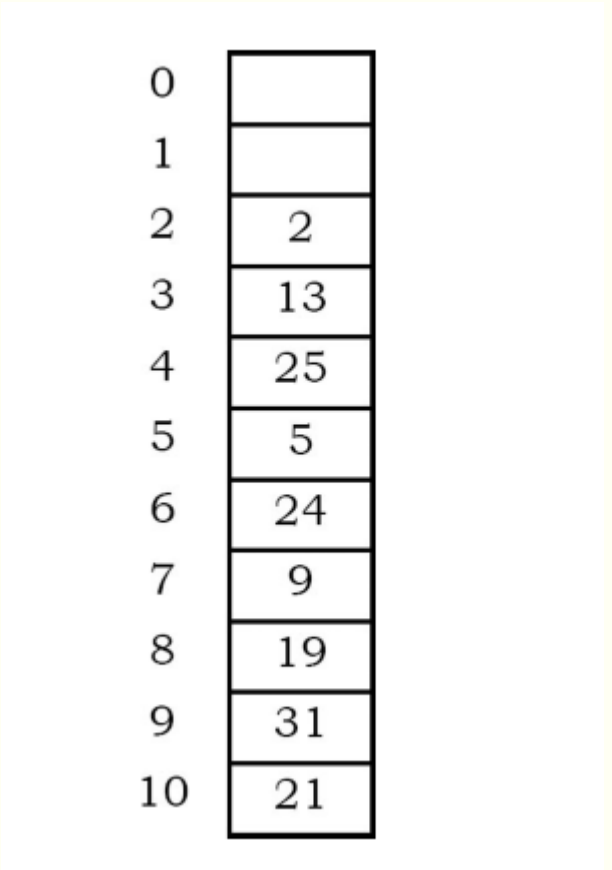
- Thăm dò tuyến tính
  - Trong thăm dò tuyến tính, chúng ta tìm kiếm bảng băm tuần tự, bắt đầu từ vị trí băm ban đầu.
  - Nếu một vị trí đã được sử dụng, ta sẽ kiểm tra vị trí tiếp theo:
    - rehash(key) = (n+1)% tablesize
    - Nên lựa chọn tablesize là số nguyên tố dạng 4k+3
- Thăm dò bậc hai
  - Trước tiên sử dụng thăm dò tuyến tính, nếu va chạm, sử dụng:
    - rehash(key) =$(n + k^2)$ % tablesize

Định địa chỉ mở

- h(key) =key mod 11

Insert keys

31 *mod* 11 = 9
19 *mod* 11 = 8
2 *mod* 11 = 2
13 *mod* 11 = 2 → $2 + 1^2 = 3$
25 *mod* 11 = 3 → $3 + 1^2 = 4$
24 *mod* 11 = 2 → $2 + 1^2$, $2 + 2^2 = 6$
21 *mod* 11 = 10
9 *mod* 11 = 9 → $9 + 1^2$, $9 + 2^2$ *mod* 11, $9 + 3^2$ *mod* 11=7



Định địa chỉ mở

- Băm kép
  - Sử dụng hai hàm băm h1 và h2:
  - Hàm băm h1 đóng vai trò như hàm băm h trong các phương pháp trước, nó xác định vị trí thăm dò đầu tiên
  - Hàm băm h2 xác định bước thăm dò.
  - Điều đó có nghĩa là, ứng với mỗi khoá k, dãy thăm dò là:
- $h1(k) + m\ h2(k)$, với $m = 0, 1, 2, \ldots$
  - Bởi vì h2(k) là bước thăm dò, nên hàm băm h2 phải thoả mãn điều kiện h2(k) khác 0 với mọi k.

Định địa chỉ mở

Table size is 11 (0..10)
Hash Function: assume $h1(key) = key \bmod 11$ and $h2(key) = 7 - (key \bmod 7)$

Insert keys:

58 *mod* 11 = 3
14 *mod* 11 = 3 → 3 + 7 = 10
91 *mod* 11 = 3 → 3+ 7,3+ 2* 7 *mod* 11 = 6
25 *mod* 11 = 3 → 3 + 3,3 + 2*3 = 9

0	
1	
2	
3	58
4	25
5	
6	91
7	
8	
9	25
10	14

Cài đặt bảng băm

- Các phép toán
  - find(k) trả về 1 phần tử có khóa k. Nếu không thấy trả về NULL.
  - findAll(k)
  - insert(k, v) thêm phần tử (k, v) và trả về con trỏ tới nó
  - erase(k) loại bỏ phần tử bất kì có khóa bằng k
  - size() trả về số lượng phần tử
  - empty() kiểm tra xem từ bảng băm rỗng hay không
- Xem mục 9.4 (p256), 9.5 (p261) giáo trình

Tổng kết

- Cài đặt bảng băm
- Các kỹ thuật giải quyết va chạm

Tiếp theo…

- Hàng ưu tiên