HK I, 2012-2013

Bài 9: Bảng băm

Giảng viên: Hoàng Thị Điệp
Khoa Công nghệ Thông tin – Đại học Công Nghệ

Mục tiêu bài học

- Phương pháp băm
- Các hàm băm
  - Hash function
- Các chiến lược giải quyết va chạm
  - Collision resolution

Tập động và Từ điển

- KDLTT tập động
  - **find**
  - insert
  - remove/erase
  - max
  - min
  - next
  - previous
- KDLTT từ điển

Cài KDLTT từ điển bằng các CTDL đã học

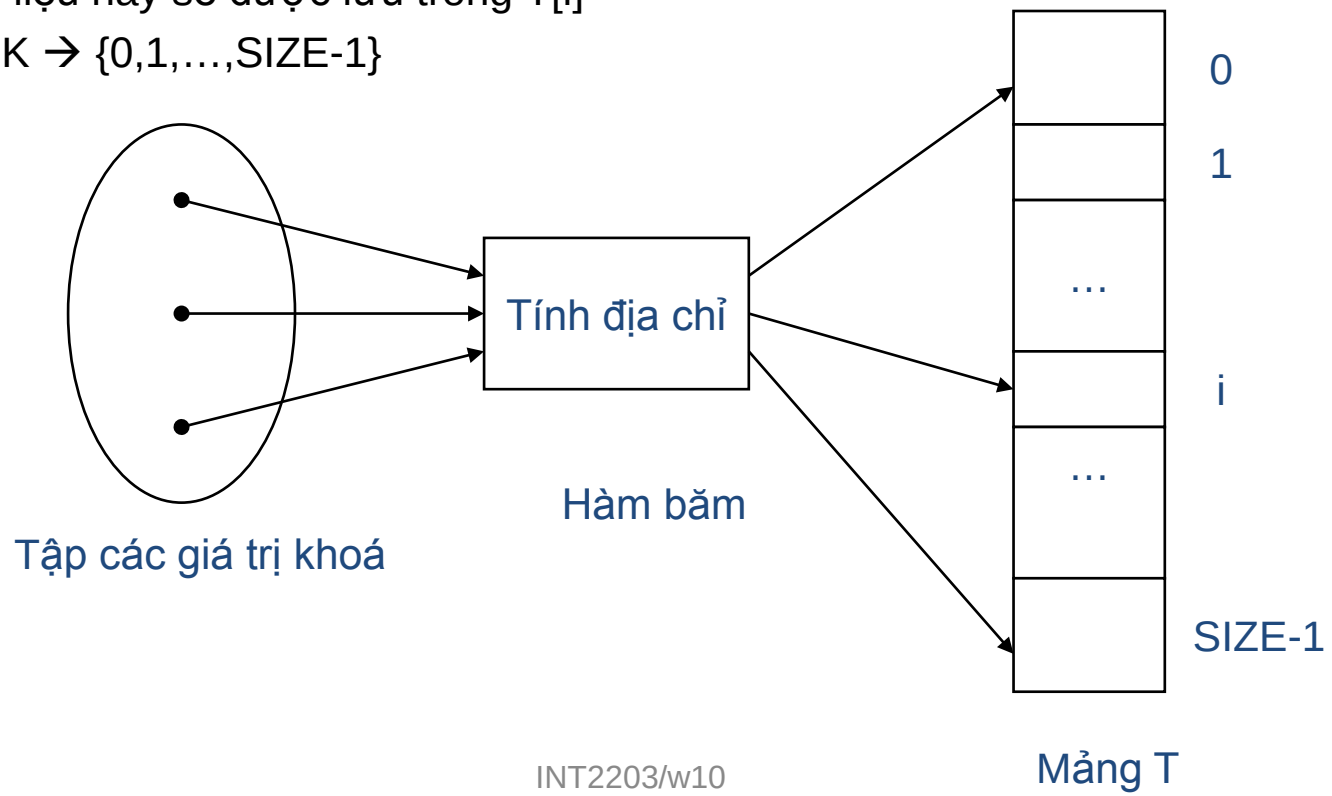
- Mảng được sắp / không được sắp
- DSLK đơn/kép được sắp / không được sắp
- Cây tìm kiếm nhị phân

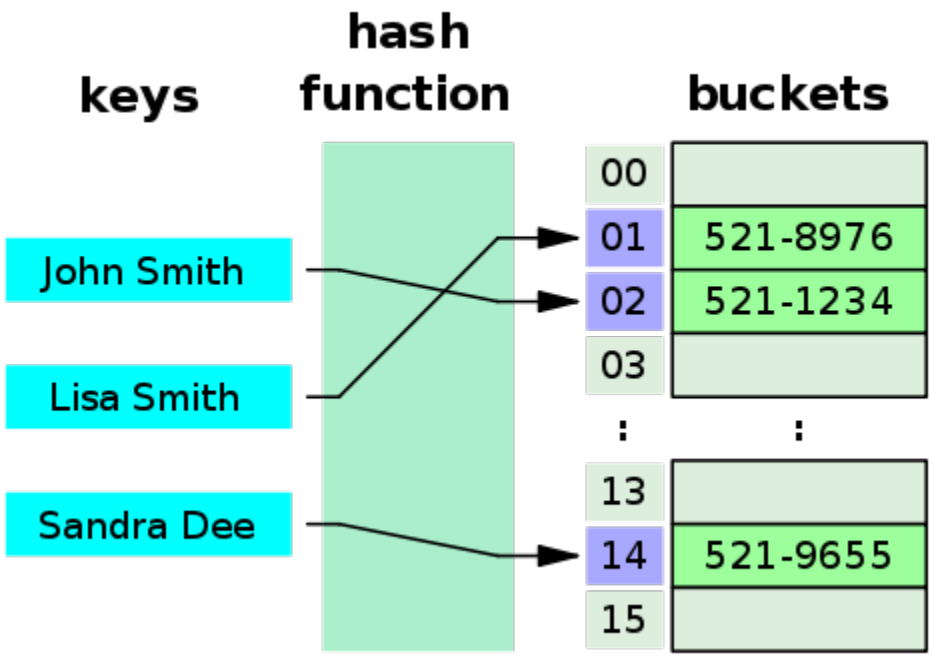
Cài KDLTT từ điển bằng mảng

- Nếu khoá của dữ liệu là số nguyên không âm và nằm trong khoảng [0..SIZE-1]
  - có thể sử dụng một mảng data có cỡ SIZE
  - dữ liệu có khoá k sẽ được lưu trong data[k]
  - tìm kiếm, xen, loại đều thực hiện trong thời gian O(1)
- Thực tế không khả thi vì
  - số phần tử dữ liệu có thể rất nhỏ so với SIZE
  - khoá có thể không phải là số nguyên
- Ta muốn lợi dụng tính ưu việt của phép truy cập trực tiếp của mảng

Phương pháp băm

- Lưu tập dữ liệu trong mảng T với cỡ là SIZE
- Hàm băm: là hàm ứng với mỗi giá trị khoá k của dữ liệu với một chỉ số i (0 <= i <= SIZE-1)
  - Dữ liệu này sẽ được lưu trong T[i]
  - h : K → {0,1,…,SIZE-1}




keys
hash function
buckets
John Smith
Lisa Smith
Sandra Dee
00
01
02
03
13
14
15
521-8976
521-1234
521-9655

Sự va chạm

- Nếu
  - có $k_1 \neq k_2$ thì $h(k_1) \neq h(k_2)$, và
  - việc tính chỉ số h(k) ứng với mỗi khoá k chỉ đòi hỏi thời gian hằng

thì các phép toán tìm kiếm, xen, loại chỉ cần thời gian O(1)

- Va chạm
  - Trong thực tế $k_1 \neq k_2$ có thể cho $h(k_1) = h(k_2)$
- Giải quyết va chạm như thế nào?

Hàm băm

- Hàm băm tốt
  - tính nhanh và dễ dàng
  - đảm bảo ít va chạm
- Một số hàm băm
  - Khóa là số nguyên không âm
    - Phương pháp chia
    - Phương pháp nhân
  - Khóa là xâu ký tự: đổi xâu thành số nguyên không âm

Khóa là số nguyên không âm

- Phương pháp chia
  - h(k) = k mod SIZE
  - nhạy cảm với cỡ của bảng băm
    - chọn SIZE để hạn chế xảy ra va chạm
    - chọn số nguyên tố có dạng đặc biệt, chẳng hạn có dạng 4k+3
- Phương pháp nhân
  - $h(k) = \lfloor (\alpha k - \lfloor \alpha k \rfloor) . SIZE \rfloor$
  - Ký hiệu $\lfloor x \rfloor$ chỉ phần nguyên của số thực x
  - Thực tế thường chọn

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

Khoá là xâu ký tự

- Trước tiên, đổi các xâu ký tự thành các số nguyên, dùng bảng mã ASCII
  - Xâu ký tự có thể xem như một số trong hệ đếm cơ số 128
    - Sau đó chuyển sang hệ đếm cơ số 10
    - Ví dụ

      "NOTE" → 'N'.128^3 + 'O'.128^2 + 'T'.128 + 'E' =
      = 78.128^3 + 79.128^2 + 84.128 + 69
    - Nhược điểm: xâu dài cho kết quả vượt quá khả năng biểu diễn của máy tính
  - Cải tiến: Xâu ký tự thường được tạo thành từ 26 chữ cái và 10 chữ số, và một vài ký tự khác. Thay 128 bởi 37
    - Tính số nguyên ứng với xâu ký tự theo luật Horner
    - Ví dụ

      "NOTE" → 78.37^3 + 79.37^2 + 84.37 + 69=
      = ((78.37 + 79).37 +84).37 +69

Giải quyết va chạm

- Dữ liệu d_1 với khoá k_1 đã được lưu trong T[i], i = $h(k_1)$. Ta cần thêm dữ liệu d_2 với khoá k_2
  - nếu $h(k_2)$ = i thì dữ liệu d_2 cần được đặt vào vị trí nào?
- Các phương pháp
  - Phương pháp định địa chỉ mở (open addressing/probing)
    - mỗi khi xảy ra va chạm, tiến hành thăm dò để tìm một vị trí còn trống trong bảng và đặt dữ liệu mới vào đó
  - Phương pháp tạo dây chuyền (separate chaining)
    - tạo ra một CTDL lưu giữ tất cả các dữ liệu có cùng vị trí i và "gắn" CTDL này vào vị trí đó trong bảng

Phương pháp định địa chỉ mở

- Giả sử vị trí ứng với khoá k là i, i=h(k)
  - Từ vị trí này chúng ta lần lượt xem xét các vị trí i_0 , i_1 , i_2 ,…, i_m ,…
  - Trong đó i_0 = i, i_m(m=0,1,2,…) là vị trí thăm dò ở lần thứ m.
  - Dãy các vị trí này sẽ được gọi là dãy thăm dò.
- Xác định dãy thăm dò
  - Thăm dò tuyến tính (linear probing)
    - Dãy thăm dò là i , i+1, i+2 , …
  - Thăm dò bình phương (quadratic probing)
    - Dãy thăm dò là i , $i + 1^2$, $i + 2^2$,… , $i + m^2$,…
  - Băm kép (double hashing)
    - Dãy thăm dò là $h_1(k) + m\ h_2(k)$, với m = 0, 1, 2, …

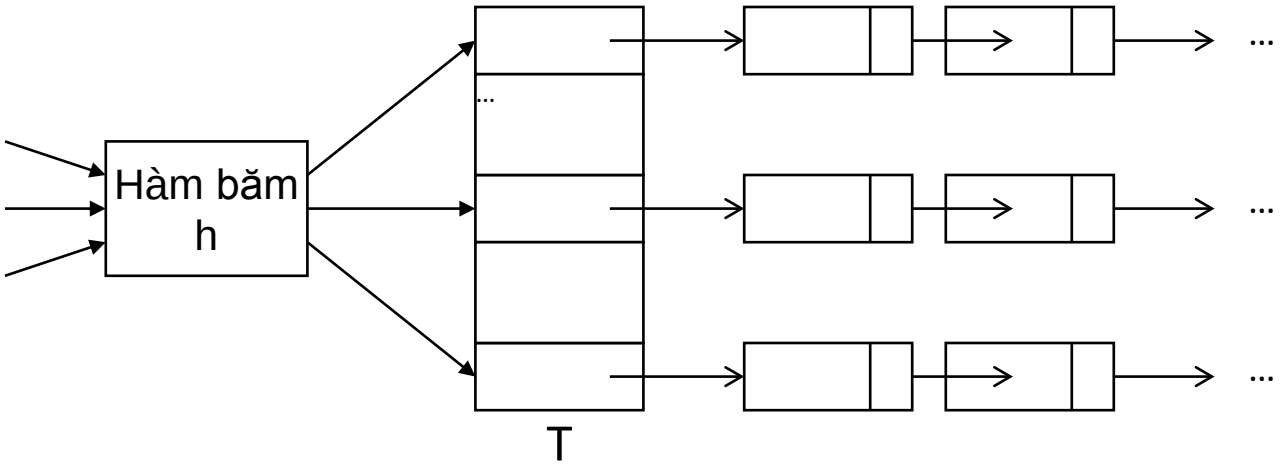
Nhận xét

- Thăm dò tuyến tính
  - Ưu điểm: cho phép xét tất cả các vị trí trong mảng
    - phép insert luôn thực hiện được, trừ khi mảng đầy
  - Nhược điểm:
    - dữ liệu tập trung thành các đoạn
    - tìm kiếm tuần tự trong từng đoạn
- Thăm dò bình phương
  - Ưu điểm: tránh được nhược điểm của thăm dò tuyến tính
  - Nhược điểm: không cho phép ta tìm đến tất cả các vị trí trong mảng
    - phép insert có thể không thực hiện được
    - nếu cỡ của mảng là số nguyên tố, thì thăm dò bình phương cho phép ta tìm đến một nửa số vị trí trong mảng
- Băm kép
  - nếu cỡ của mảng và bước thăm dò $h_2(k)$ nguyên tố cùng nhau thì phương pháp băm kép cho phép tìm đến tất cả các vị trí trong mảng

Các phép toán

- Tìm kiếm? Xen? Loại?
- Minh họa
  - SIZE = 11
  - Thăm dò tuyến tính
  - insert(388), insert(130), insert(13), insert(14), insert(926)
  - find(47)
  - remove(388), find(926)

Phương pháp tạo dây chuyền

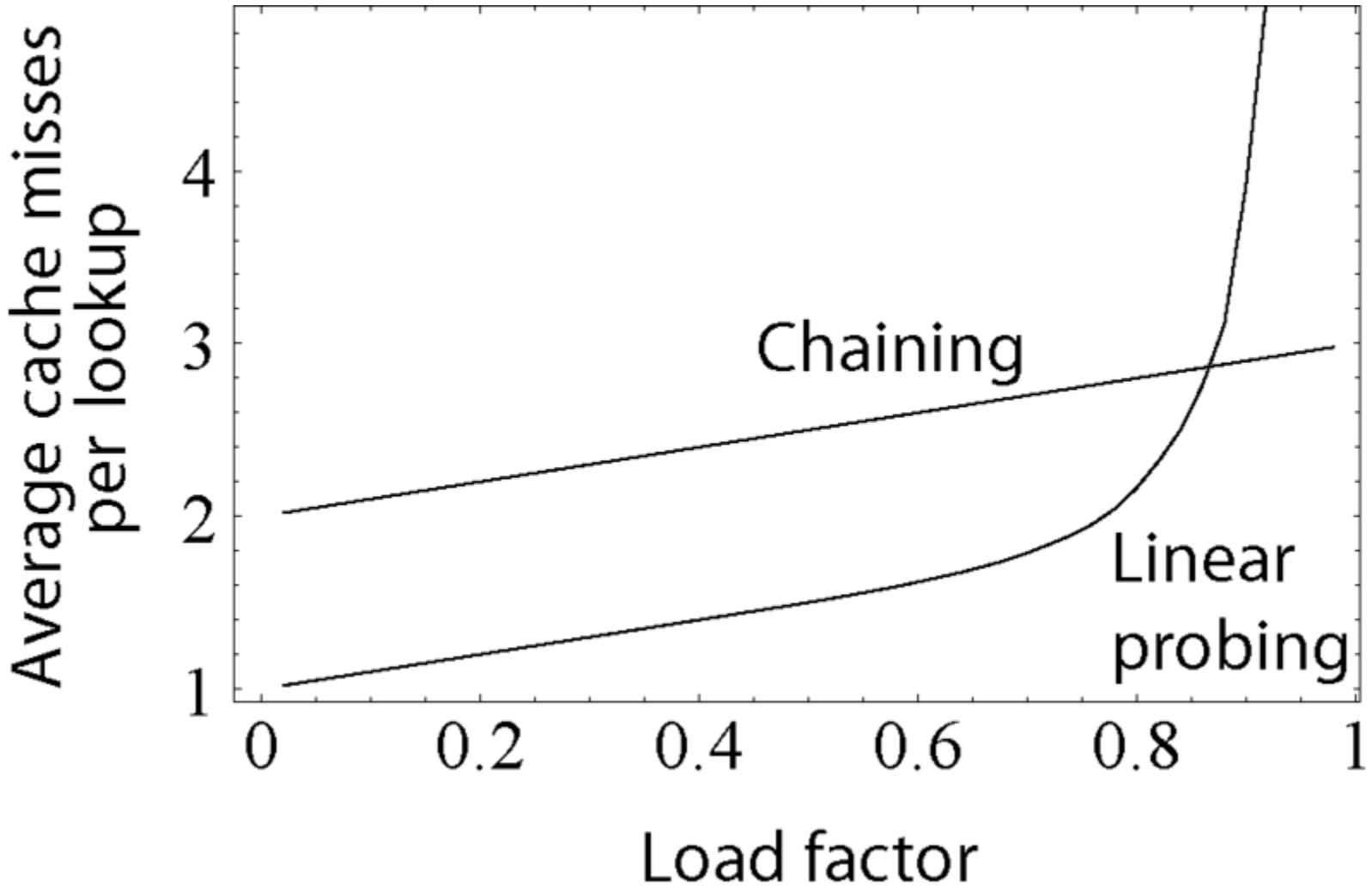


- Ưu điểm
  - số dữ liệu được lưu không phụ thuộc vào cỡ của mảng
- Các phép toán
  - Tìm kiếm?
  - Xen?
  - Loại?

Hiệu quả của phương pháp băm

- Tham số α $\alpha = \frac{N}{SIZE}$
- Băm đ/c mở: mức độ đầy (load factor)
  - α tăng thì khả năng va chạm tăng
  - Khi thiết kế, cần đánh giá max của N để lựa chọn SIZE
  - α không nên vượt quá 2/3
- Băm dây chuyền: độ dài trung bình của một dây chuyền

		Băm đ/c mở, Thăm dò tuyến tính	Băm đ/c mở, Thăm dò bình phương	Băm dây chuyền
Thời gian trung bình	Tìm kiếm thành công	$\frac{1}{2}\left(1+\frac{1}{1-\alpha}\right)$	$\frac{-\ln(1-\alpha)}{\alpha}$	$1+\frac{1}{\alpha}$
	Tìm kiếm thất bại	$\frac{1}{2}\left(1+\frac{1}{(1-\alpha)^2}\right)$	$\frac{1}{1-\alpha}$	α


Average cache misses per lookup
4
3
2
1
Chaining
Linear probing
0
0.2
0.4
0.6
0.8
1
Load factor

Chuẩn bị bài tới

- Đọc chương 10 (Hàng ưu tiên)