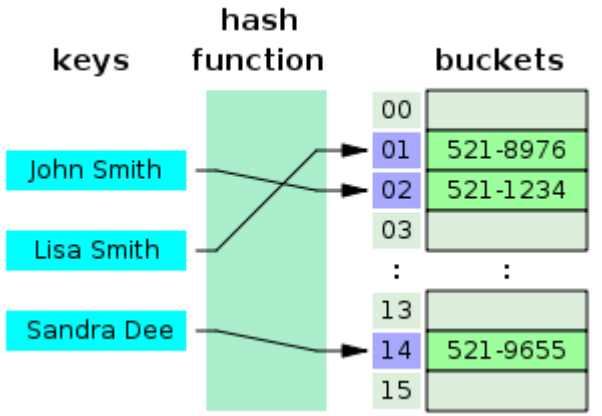
Bài 7.1: Tổng quan về bảng băm

- ✓ Khái niệm và đặc điểm
- ✓ Quá trình băm
- ✓ Xử lý xung đột mã băm
- ✓ Quản lý kích thước bảng băm
- ✓ Ưu nhược điểm

Khái niệm và đặc điểm

- Bảng băm(hash table) là một cấu trúc dữ liệu sử dụng để lưu trữ dữ liệu dưới dạng cặp key-value.
- Bảng băm dựa trên mảng kết hợp trong đó chỉ số của phần tử mảng có được bằng cách sử dụng một hàm băm để băm key trong cặp key-value. Sau đó giá trị value sẽ được lưu vào vị trí vừa có được.
- Giá trị trả về từ hàm băm được gọi là mã băm, giá trị băm hay gọi tắt là băm. Trong khóa học ta sẽ gọi là mã băm.



Khái niệm và đặc điểm

- Nếu lý tưởng, hàm băm sẽ cho ra tương ứng mỗi key một mã băm duy nhất.
- Nhưng hầu hết các bảng băm sử dụng các hàm băm không hoàn hảo do đó có thể cho ra cùng mã băm với hai key khác nhau.
- Lúc này xảy ra xung đột băm và ta phải xử lý nó bằng một cách nào đó.

Khái niệm và đặc điểm

- Trong bảng băm có số chiều phù hợp, chi phí cho việc tìm kiếm phần tử độc lập với số phần tử trong mảng. Thường là O(1).
- Nhiều bảng băm được thiết kế cho phép thêm và xóa cặp key-value với chi phí thời gian là hằng số.
- Trong hầu hết trường hợp, bảng băm hiệu quả hơn nhiều so với cây tìm kiếm hay bất cứ cấu trúc bảng tìm kiếm nào khác.
- Vì lý do này người ta thường sử dụng bảng băm trong rất nhiều ứng dụng máy tính như mảng kết hợp, đánh chỉ số cơ sở dữ liệu, quản lý bộ nhớ đệm, các tập hợp.

Quá trình băm

- Băm là kĩ thuật để xác định một đối tượng cụ thể, duy nhất từ một nhóm các đối tượng tương tự nhau. Sau đây là một số ví dụ:
  - Mỗi công dân được xác định bằng một định danh duy nhất gọi là số chứng minh nhân dân hay căn cước công dân.
  - Mỗi nhân viên trong công ty được phân biệt và xác định duy nhất thông qua mã nhân viên.
- Giả sử ta có các đối tượng và ta muốn gán cho nó một key để cho việc tìm kiếm dễ dàng hơn trong mảng. Với các key là số nguyên nhỏ, ta có thể sử dụng trực tiếp key làm chỉ số. Nhưng khi key trở nên lớn không sử dụng trực tiếp làm chỉ số mảng được nữa, ta phải băm nó nhỏ lại.
- Trong quá trình băm, hàm băm được sử dụng để đưa các key lớn, phức tạp về key nhỏ, đơn giản. Giá trị này sau đó lưu trong bảng băm.

Quá trình băm

- Ý tưởng của việc băm là để phân phối các khóa đầu vào lên một bảng. Mỗi phần tử được gán một khóa đã chuyển đổi và sau đó ta có thể truy cập phần tử qua khóa đó với thời gian hằng số(nhanh).
- Quá trình băm được thực hiện qua 2 bước:
  - B1: dùng hàm băm chuyển key thành giá trị băm hashCode. Giá trị này được sử dụng làm chỉ số trong mảng.
  - B2: lưu phần tử vào mảng tại vị trí xác định: index = hashCode%arraySize.
- Để có hàm băm tốt, cần thỏa mãn các yêu cầu:
  - Dễ tính toán: hàm băm không nên trở thành 1 thuật toán.
  - Phân phối đồng đều: cho kết quả đều, không cục bộ, phân cụm.
  - Ít xung đột: xảy ra khi hai băm 2 key cho cùng 1 mã băm. Điều này nên hạn chế tối thiểu.

Quá trình băm

- Hàm băm lý tưởng là hàm băm không tạo ra cặp mã băm trùng nhau nào từ tập key đầu vào.
- Sự phân bổ đồng đều chỉ cần tương thích với kích thước bảng sử dụng trong ứng dụng.

Thống kê khóa(key)

- Một chỉ số quan trọng của bảng băm là load factor.
- Công thức: load factor = n/k;
- Trong đó:
  - N là số key đầu vào.
  - K là số lượng phần tử tối đa của mảng.
- Khi load factor càng lớn thì bảng băm hoạt động càng chậm và có thể bị hỏng. Load factor nhỏ sẽ gây lãng phí bộ nhớ.
- Tính chất thời gian hằng số của bảng băm giả định rằng load factor được giữ dưới một giá trị ràng buộc nào đó.
- Với số lượng phần tử cố định của mảng, thời gian tìm kiếm phần tử tăng lên theo số lượng key đầu vào do đó không đạt được tính chất trên.
- Trong nhiều triển khai, ta gấp đôi kích thước của bảng khi đạt đến giới hạn của load factor sau đó tiến hành băm lại tất cả các key đầu vào.
- Trong Java 10, load factor của HashMap là 0.75.

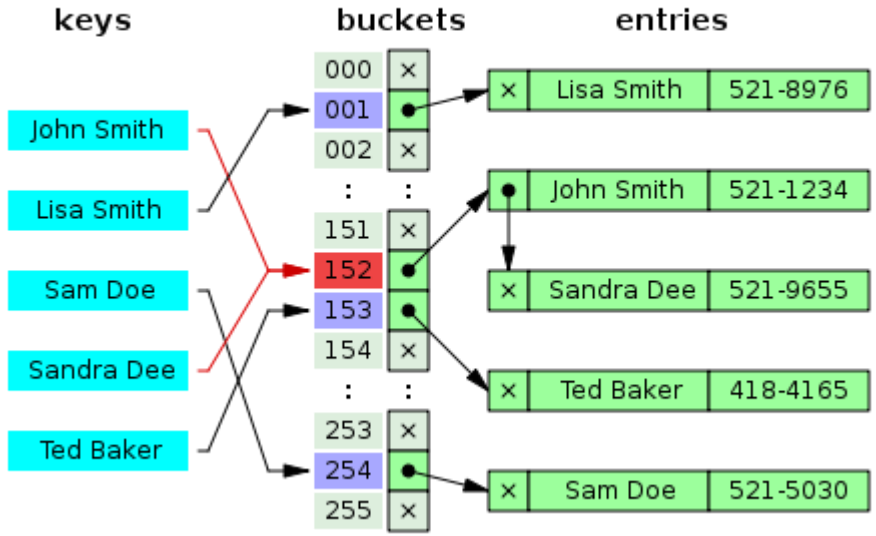
Xử lý xung đột

- Xung đột băm là không thể tránh khỏi khi băm một tập hợp ngẫu nhiên của một lượng lớn các khóa(key).
- Do đó các thiết kế bảng băm đều có phương pháp giải quyết xung đột.
- Sau đây là một số phương pháp phổ biến:
  - Phương pháp kết nối.
  - Phương pháp địa chỉ mở.
  - Phương pháp thăm dò Cuckoo.
- Trong tất cả các phương pháp này, các key hoặc con trỏ trỏ tới chúng được lưu trữ trong bảng băm cùng với giá trị liên kết với nó.
- Mức độ hiệu quả của các phương pháp trên phụ thuộc trực tiếp vào load factor.

Phương pháp kết nối

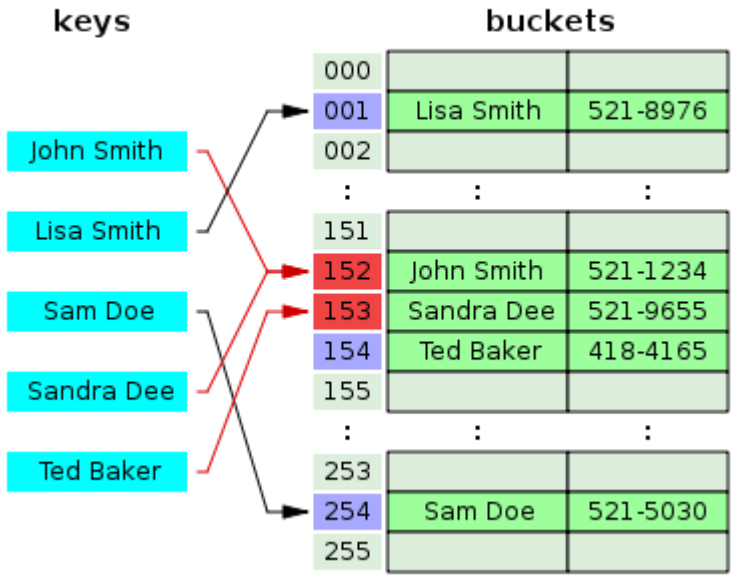
- Trong phương pháp kết nối, mỗi ô của bảng có thể trỏ tới danh sách liên kết, cây nhị phân tìm kiếm tự cân bằng, hoặc mảng.
- Phổ biến nhất trong các loại trên là sử dụng danh sách liên kết tại mỗi ô của bảng vì tính đơn giản.
- Thời gian để thực hiện hành động của bảng băm bằng thời gian tìm kiếm ô trong bảng + thời gian thực hiện thao tác trên danh sách liên kết.
- Nếu việc phân phối khóa trong bảng là hiệu quả, chi phí trung bình của việc tìm kiếm chỉ phụ thuộc vào số lượng khóa được phân bổ trên mỗi ô của bảng, tức load factor.
- Do đó phương pháp này vẫn hiệu quả ngay cả khi số lượng khóa đầu vào lớn hơn số ô có trong bảng băm.

Phương pháp kết nối



- Trường hợp xấu nhất của phương pháp này là khi tất cả các khóa phân bổ vào cùng 1 ô của bảng.
- Lúc này chi phí tìm kiếm của bảng băm là n.
- Nhược điểm của phương pháp này là tốn chi phí bộ nhớ, đặc trưng của danh sách liên kết để lưu trữ con trỏ next.
- Ngoài ra sử dụng danh sách liên kết làm cho hiệu suất cache kém hiệu quả.

Phương pháp địa chỉ mở



- Trong phương pháp này, mọi cặp key-value được lưu trực tiếp vào bảng.
- Khi có một cặp key-value khác cần thêm vào, thuật toán duyệt trong bảng theo một thứ tự thăm dò nhất định bắt đầu từ ô ứng với mã băm của key tới khi gặp 1 ô trống.
- Sau đó cặp key-value chèn vào ô trống vừa tìm được.
- Khi tìm kiếm, thuật toán cũng được duyệt theo thứ tự thăm dò bắt đầu từ ô tương ứng với mã băm.

Phương pháp địa chỉ mở

- Trong phương pháp này, mọi cặp key-value được lưu trực tiếp vào bảng.
- Khi có một cặp key-value khác cần thêm vào, thuật toán duyệt trong bảng theo một thứ tự thăm dò nhất định bắt đầu từ ô ứng với mã băm của key tới khi gặp 1 ô trống.
- Sau đó cặp key-value chèn vào ô trống vừa tìm được.
- Khi tìm kiếm, thuật toán cũng được duyệt theo thứ tự thăm dò bắt đầu từ ô tương ứng với mã băm.
- Một số thứ tự thăm dò phổ biến:
  - Thăm dò tuyến tính: khoảng cách giữa hai vị trí thăm dò liên tiếp là cố định(thường là 1).
  - Thăm dò bậc 2: khoảng cách giữa hai vị trí thăm dò liên tiếp tăng một lượng được tính bằng đa thức bậc 2.
  - Thăm dò hai lần: khoảng cách giữa hai vị trí thăm dò liên tiếp là một hàm băm khác.

Phương pháp Cuckoo

- Bắt nguồn từ hành vi của loài chim Cuckoo. Cho phép thực hiện các thao tác tìm và xóa trong thời gian hằng số.
- Phương pháp này sử dụng 2 hay nhiều hàm băm. Do đó mỗi cặp key-value có thể nằm ở 2 hoặc nhiều vị trí.
- Để tìm kiếm, hàm băm thứ nhất sẽ được sử dụng. Nếu không tìm thấy cặp key-value nào thì hàm băm tiếp theo sẽ được sử dụng, tiếp tục như vậy.
- Để chèn thêm phần tử vào bảng băm, hàm băm cũng được sử dụng lần lượt. Nếu sau khi băm mà xung đột, hàm băm kế tiếp được sử dụng. Nếu sau khi tất cả các hàm băm đc dùng mà vẫn xung đột, khóa bị xung đột sẽ bị xóa, khóa cũ sẽ bị băm lại bằng một hàm băm khác để chuyển sang vị trí khác.
- Quá trình trên thực hiện đến khi không còn xung đột băm nào xảy ra.
- Phương pháp này cần sử dụng nhiều bộ nhớ.

Các phương pháp khác

- Phương pháp Coalesced: lai giữa phương pháp kết nối và địa chỉ mở.
- Phương pháp Hotscotch: kết hợp phương pháp tuyến tính và Cuckoo.
- Phương pháp Robin Hood: một biến thể của thăm dò 2 lần.
- Phương pháp 2-choice: sử dụng 2 hàm băm.

Quản lý kích thước bảng băm

- Khi việc thêm mới phần tử vào bảng băm đạt ngưỡng của load factor, bảng băm sẽ phải băm lại.
- Hành động này bao gồm mở rộng kích thước bảng băm và đưa các phần tử đã có vào các vị trí mới.
- Để giảm tỉ lệ bộ nhớ bị lãng phí, một số triển khai thu nhỏ kích thước của bảng và tiến hành băm lại khi có phần tử bị xóa.
- Một số cách quản lý kích thước bảng băm:
  - Sao chép tất cả sang bảng mới.
  - Các lựa chọn thay thế cho việc băm lại tất cả cùng lúc: incremental resizing, monotonic keys, linear hashing, hashing for distributed hash tables.

Ưu nhược điểm

Ưu điểm:

- Lợi ích lớn nhất so với các cấu trúc dữ liệu khác là tốc độ.
- Đặc biệt phát huy hiệu quả khi là số đầu vào lớn.
- Nếu có thể dự đoán được số lượng đầu vào thì sẽ không phải cấp phát lại bảng băm.

Nhược điểm:

- Không hiệu quả nếu số phần tử đầu vào nhỏ.
- Không hiệu quả trong xử lý string như kiểm tra đánh vần chẳng hạn. Lúc này sử dụng cây hoặc mảng Judy tốt hơn.
- Chi phí thực hiện một hành động đơn lẻ khá lớn.
- Bảng băm thể hiện vị trí tham chiếu kém, do phân phối ngẫu nhiên trong bộ nhớ.
- Khi thuật toán băm có nhiều xung đột, bảng băm không hiệu quả.

Nội dung tiếp theo

Tạo bảng băm tự định nghĩa