Bài 7.5: Băm lại-rehash

- ✓ Các bước thực hiện
- ✓ Mã giả và triển khai
- ✓ Ví dụ minh họa
- ✓ Bài tập thực hành

Các bước thực hiện

- B1: Lưu lại kích thước cũ, bảng cũ.
- B2: Tạo kích thước mới gấp đôi kích thước cũ.
- B3: Nếu kích thước mới > số phần tử tối đa
  - B3.1: Nếu kích thước cũ bằng kích thước tối đa thì kết thúc.
  - B3.2: Ngược lại, kích thước cũ bằng kích thước cũ.
- B4: Cấp phát mảng với kích thước mới.
- B5: Tăng số lần sửa đổi bảng lên 1.
- B6: tính toán lại ngưỡng.
- B7: Cho bảng băm tham chiếu đến mảng mới.

Các bước thực hiện

- B8: Lặp từ cuối bảng cũ lên đầu, thực hiện
  - B8.1: Lấy từng phần tử old tại vị trí đang xét trong bảng cũ gán cho phần tử e.
  - B8.2: Cập nhật node old: old = old.next.
  - B8.3: Tính toán lại chỉ số của phần tử đó trong mảng mới.
  - B8.4: Cập nhật node next của phần tử e.
  - B8.5: Gán phần tử e vào vị trí mới có được ở bước B8.3.

Mã giả

```
// băm lại bảng băm cũ
rehash() {
    oldCapacity = table.length; // lưu lại kích thước bảng cũ
    oldMap = table; // lưu lại bảng cũ
    // nhân đôi kích thước bảng băm, sử dụng phép dịch trái bít
    newCapacity = (oldCapacity << 1) + 1;
    if (newCapacity - MAX_ARRAY_SIZE > 0) { // nếu kích thước mới quá lớn
        if (oldCapacity == MAX_ARRAY_SIZE) // nếu kích thước cũ bằng khả năng tối đa
            return; // tiếp tục sử dụng mảng cũ, không cần băm lại
        newCapacity = MAX_ARRAY_SIZE; // gán lại kích thước mới = kích thước cũ
    }
    newMap = new Entry<?, ?>[newCapacity]; // cấp phát mảng mới
    modCount++; // tăng số lần sửa đổi bảng băm
    // tính toán lại ngưỡng mới
    threshold = min(newCapacity * loadFactor, MAX_ARRAY_SIZE + 1);
    table = newMap; // cập nhật lại bảng băm
    for (i = oldCapacity - 1; i >= 0; i--) { // duyệt từ cuối mảng cũ
        // cập nhật tất cả các phần tử trong danh sách các phần tử tại vị trí đang xét
        for (old = (Entry<K, V>) oldMap[i]; old != null; ) {
            e = old; // lấy phần tử tại vị trí i
            old = old.next; // chuyển đến phần tử(node) kế tiếp
            index = (e.hash & 0x7FFFFFFF) % newCapacity; // tìm vị trí mới trong newMap
            e.next = (Entry<K, V>) newMap[index]; // phần tử(node) kế tiếp của node mới
            newMap[index] = e; // gắn phần tử vừa băm lại vào vị trí index trong newMap
        }
    }
}
```

Mã thật

```
protected void rehash() {
    int oldCapacity = table.length; // lưu lại kích thước bảng cũ
    Entry<?, ?>[] oldMap = table; // lưu lại bảng cũ
    // nhân đôi kích thước bảng băm, sử dụng phép dịch trái bít
    int newCapacity = (oldCapacity << 1) + 1;
    if (newCapacity - MAX_ARRAY_SIZE > 0) { // nếu kích thước mới quá lớn
        if (oldCapacity == MAX_ARRAY_SIZE) // nếu kích thước cũ bằng khả năng tối đa
            return; // tiếp tục sử dụng mảng cũ, không cần băm lại
        newCapacity = MAX_ARRAY_SIZE; // gán lại kích thước mới = kích thước cũ
    }
    Entry<?, ?>[] newMap = new Entry<?, ?>[newCapacity]; // cấp phát mảng mới
    modCount++; // tăng số lần sửa đổi bảng băm
    // tính toán lại ngưỡng mới
    threshold = (int) Math.min(newCapacity * loadFactor, MAX_ARRAY_SIZE + 1);
    table = newMap; // cập nhật lại bảng băm

    for (int i = oldCapacity - 1; i >= 0; i--) { // duyệt từ cuối mảng cũ
        // cập nhật tất cả các phần tử trong danh sách các phần tử tại vị trí đang xét
        for (Entry<K, V> old = (Entry<K, V>) oldMap[i]; old != null; ) {
            Entry<K, V> e = old; // lấy phần tử tại vị trí i
            old = old.next; // chuyển đến phần tử(node) kế tiếp
            int index = (e.hash & 0x7FFFFFFF) % newCapacity; // tìm vị trí mới trong newMap
            e.next = (Entry<K, V>) newMap[index]; // phần tử(node) kế tiếp của node mới
            newMap[index] = e; // gắn phần tử vừa băm lại vào vị trí index trong newMap
        }
    }
}
```

Nội dung tiếp theo

Lấy danh sách key, value trong bảng băm