Cuckoo

1. Tổng quan

Cukoo hashing được Pagh and Rodler giới thiệu vào năm 2001. Cuckoo hashing là phương pháp băm địa chỉ mở để giải quyết xung đột khóa trong bảng băm, cho phép tra cứu và tính năng xóa trong trường hợp xấu nhất là O (1), và thời gian thêm giá trị trong trường hợp xấu là O(n).

Cuckoo hashing duy trì hai bảng băm T_1 và T_2, với hai hàm băm độc lập h_1 và h_2. Khóa được lưu ở một trong hai vi trí, với vị trí có thể có trong mỗi bảng. Việc tra cứu khóa x được thực hiện bằng cách tìm kiếm một thực thể phù hợp ở một bucket của mỗi bảng băm $T_1[h_1(x)]$ hoặc $T_2[h_2(x)]$. Chức năng xóa được thực hiện đơn giản bằng cách loại bỏ một key cần xóa ra khỏi những vị trí bucket tồn tại, nhưng việc thêm vào thì phức tạp hơn.

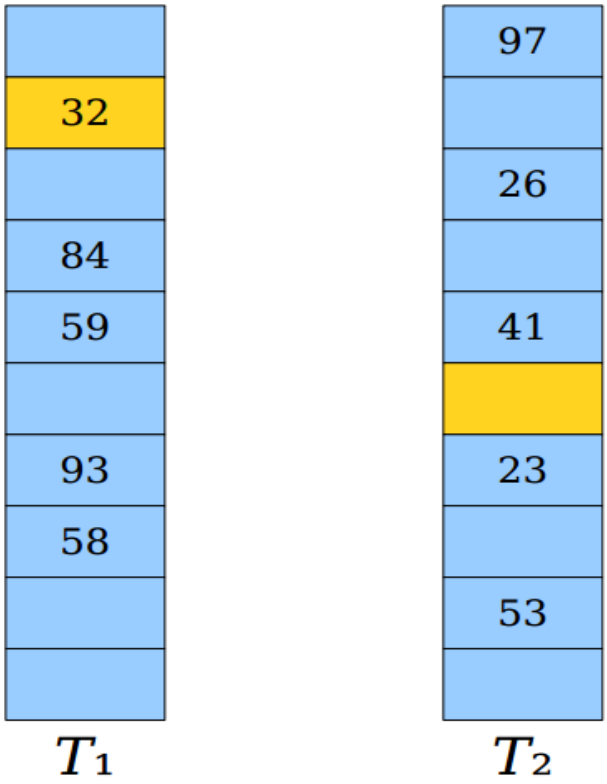
2. Cách thức hoạt động

Cuckoo hashing là cơ chế bảng băm sử dụng hai bảng băm T_1 và T_2, mỗi bảng có r buckets với hai hàm băm độc lập h_1 và h_2 dùng để ánh xạ dữ liệu đến mảng các bucket {0, ..., r-1}. Khóa x có thể được lưu trong một những vị trí T_1 $[h_1(x)]$ và $T_2[h_2(x)]$. Trong cuckoo hashing, việc tra cứu sẽ thực hiện tìm kiếm trong cả hai vị trí T_1 $[h_1(x)]$ và $T_2[h_2(x)]$ và tìm kiếm thành công nếu khóa x được lưu trong một trong vị trí được tìm kiếm trong hai bảng băm T_1 và T_2. Tra cứu trong cuckoo hashing mất thời gian O (1) vì chỉ có hai vị trí đươc kiểm tra. Thuật toán sau mô tả cuckoo hashing thực hiện tra cứu:

```
function lookup(x)
    return T1[h1(x)] = x  ∨  T2[h2(x)] = x
end
```

Ví dụ: Cuckoo hashing thực hiện việc tra cứu x = 32. h_1 và h_2 ánh xạ giá trị x đến các bucket trong bảng T_1 và T_2. Đầu tiên sẽ tìm kiếm vị trí $h_1(x)$ trong bảng T_1.

Nếu x được tìm thấy trong bảng T_1 thì việc tìm kiếm sẽ được dừng, còn nếu không sẽ chuyển sang tìm kiếm trong bảng T_2. Nếu tìm kiếm trên cả hai bảng đều không có x thì kết luận không tìm thấy x.

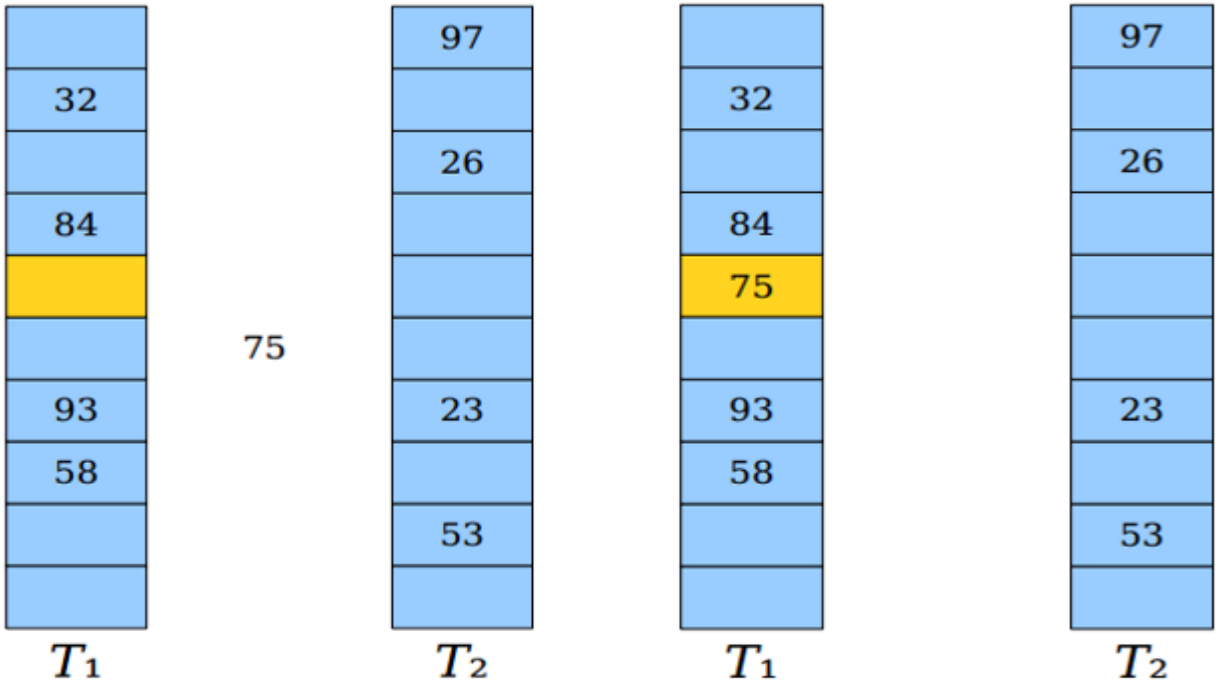


Tương tự với tính năng tra cứu, tính năng xóa cũng là tính năng đơn giản và mất thời gian O (1) bởi vì chỉ hai vị trí được kiểm tra.

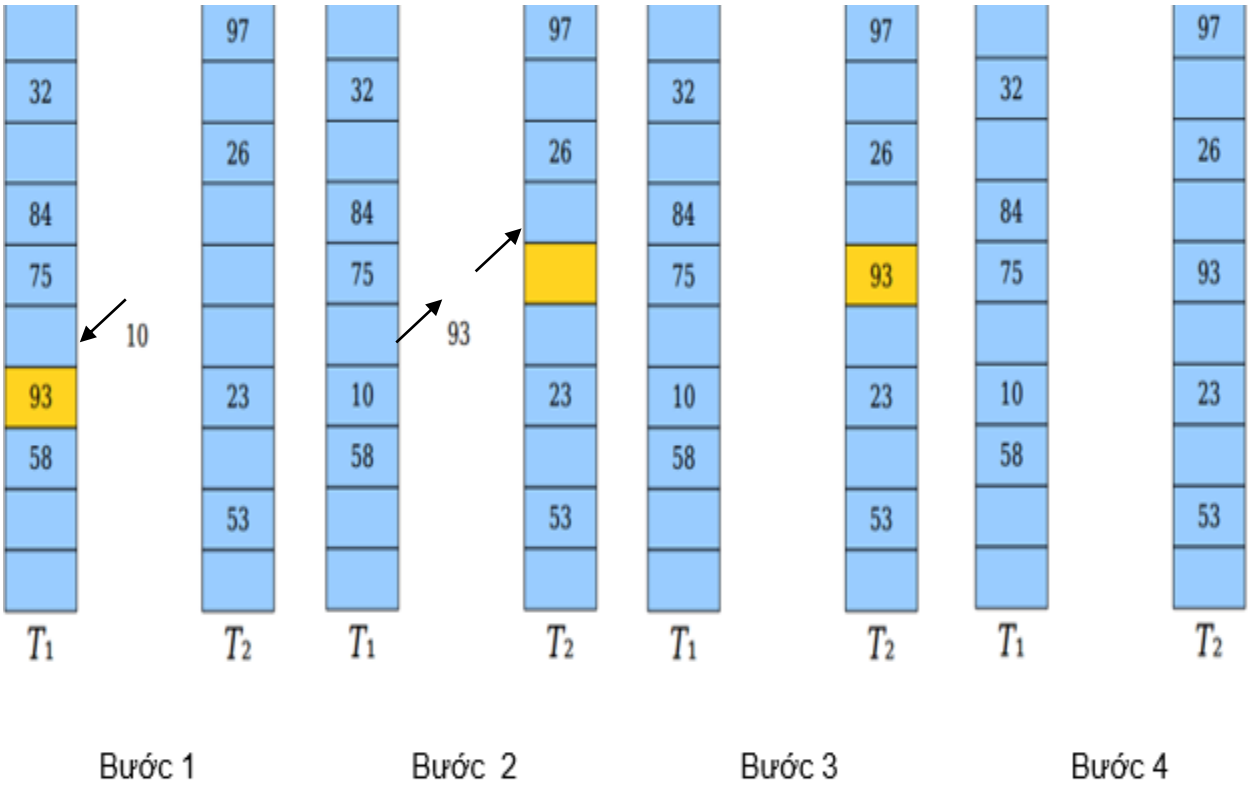
Với chức năng chèn, cuckoo hashing sẽ đẩy khóa khác đã có chỗ sẵn ra và băm lại cho đến khi tất cả các khóa có chỗ của mình. Nếu x được thêm vào, đầu tiên cuckoo hash sẽ kiểm tra xem bucket $h_1(x)$ của bảng T1 đã có giá trị chưa. Nếu không thì x chèn vào $h_1(x)$. Ngược lại, x sẽ được gán vào $T_1[h_1(x)]$, và khóa y trước sẽ được thêm vào T_2. Điều này có thể xảy ra tiến trình lặp, số lần lặp lại được giới hạn bởi giá trị "MaxLoop". Nếu số lần lặp đạt mức tối đa, tất cả khóa sẽ được băm lại với hàm băm mới.

```
procedure insert(x)
  if lookup(x) then return;
  loop MaxLoop times
    if T1[h1(x)] = ⊥ then { T1[h1(x)] ← x; return; }
    x ↔ T1[h1(x)];
    if T2[h2(x)] = ⊥ then { T2[h2(x)] ← x; return; }
    x ↔ T2[h2(x)];
  end loop
  rehash(); insert(x);
end;
```

Ví dụ: Thêm khóa x = 75 vào bảng băm. Đầu tiên sẽ bắt đầu thêm vào $h_1(x)$ của bảng T_1. Nếu như $h_1(x)$ còn trống thì x được đặt ở $h_1(x)$.



Ví dụ 2: Thêm khóa z = 10. Đầu tiên thêm x vào bảng T_1. Nếu $h_1(z)$ còn trống thì z được đặt ở $h_1(z)$. Ngược lại, nếu $h_1(z)$ đã tồn tại khóa y = 93. Thì loại y = 93 ra khỏi vị trí $h_1(z)$, lúc này vị trí $h_1(z)$ sẽ được thay thế từ y thành z và khóa y sẽ được băm lại vào bảng T_2. Lặp lại cho đến khi các giá trị ổn định.



3. Đánh giá

Thiết kế của cuckoo hashing là sử dụng hai hàm băm thay vì chỉ một. Điều này giúp mỗi khóa có thể có hai vị trí để lưu trú. Để sử dụng khả năng của băng băm tối ưu và tốc độ chèn của cuckoo hashing sẽ sử dụng nhiều hơn hai hàm băm thay thế có thể được mong đợi. Việc sử dụng ba hàm băm nâng tải tới 91%.

Cuckoo hashing có một số tính chất:

- Dễ triển khai
- Tra cứu sử dụng hai thăm dò
- Hiệu quả trong trường hợp trung bình