

Cấu trúc dữ liệu Hash Table

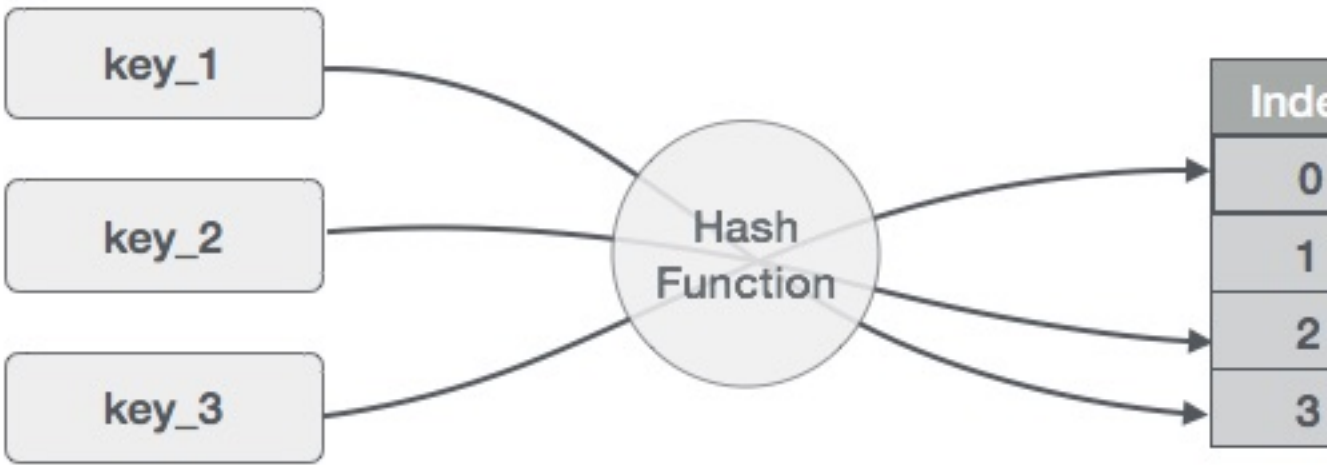
Hash Table là gì?

Cấu trúc dữ liệu Hash Table là một cấu trúc dữ liệu lưu giữ dữ liệu theo cách thức liên hợp. Trong Hash Table, dữ liệu được lưu giữ trong định dạng mảng, trong đó các giá trị dữ liệu có giá trị chỉ mục riêng. Việc truy cập dữ liệu trở nên nhanh hơn nếu chúng ta biết chỉ mục của dữ liệu cần tìm.

Do đó, với loại cấu trúc dữ liệu Hash Table này thì các hoạt động chèn và hoạt động tìm kiếm sẽ diễn ra rất nhanh, bất chấp kích cỡ của dữ liệu là bao nhiêu. Hash Table sử dụng mảng như là một kho lưu giữ trung gian và sử dụng kỹ thuật Hash để tạo chỉ mục tại nơi phần tử được chèn vào.

Kỹ thuật Hashing

Hashing là một kỹ thuật để chuyển đổi một dãy các giá trị khóa (key) vào trong một dãy các giá trị chỉ mục (index) của một mảng. Chúng ta đang sử dụng **toán tử lấy phần dư** để thu được một dãy các giá trị khóa. Giả sử có một HashTable có kích cỡ là 20, và dưới đây là các phần tử cần được lưu giữ. Phần tử trong định dạng (key, value).



- (1,20)
- (2,70)
- (42,80)
- (4,25)
- (12,44)

- (14,32)
- (17,11)
- (13,78)
- (37,98)

Stt	Key	Hash	Chỉ mục mảng
1	1	1 % 20 = 1	1
2	2	2 % 20 = 2	2
3	42	42 % 20 = 2	2
4	4	4 % 20 = 4	4
5	12	12 % 20 = 12	12
6	14	14 % 20 = 14	14
7	17	17 % 20 = 17	17
8	13	13 % 20 = 13	13
9	37	37 % 20 = 17	17

Kỹ thuật Dò tuyến tính (Linear Probing)

Chúng ta thấy rằng có thể xảy ra trường hợp mà kỹ thuật Hashing được sử dụng để tạo chỉ mục đã tồn tại trong mảng. Trong tình huống này, chúng ta cần tìm kiếm vị trí trống kế tiếp trong mảng bằng việc nhìn vào trong ô tiếp theo cho tới khi chúng ta tìm thấy một ô trống. Kỹ thuật này được gọi là **Dò tuyến tính (Linear Probing)**.

Stt	Key	Hash	Chỉ mục mảng	Sau kỹ thuật Linear Probing, chỉ mục mảng

1	1	1 % 20 = 1	1	1
2	2	2 % 20 = 2	2	2
3	42	42 % 20 = 2	2	3
4	4	4 % 20 = 4	4	4
5	12	12 % 20 = 12	12	12
6	14	14 % 20 = 14	14	14
7	17	17 % 20 = 17	17	17
8	13	13 % 20 = 13	13	13
9	37	37 % 20 = 17	17	18

Các hoạt động cơ bản trên Hash Table

Dưới đây là một số hoạt động cơ bản có thể được thực hiện trên cấu trúc dữ liệu Hash Table.

- **Hoạt động tìm kiếm**: tìm kiếm một phần tử trong cấu trúc dữ liệu HashTable.
- **Hoạt động chèn**: chèn một phần tử vào trong cấu trúc dữ liệu HashTable.
- **Hoạt động xóa**: xóa một phần tử từ cấu trúc dữ liệu HashTable.

Phần tử dữ liệu (DataItem) trong HashTable

Phần tử dữ liệu bao gồm: data và key. Dựa vào key này chúng ta có thể thực hiện các hoạt động tìm kiếm trong cấu trúc dữ liệu HashTable.

```
struct DataItem {    int data;       int key; };
```

Phương thức của cấu trúc dữ liệu Hash Table

Xác định một phương thức để ước lượng Hash Code của key của phần tử dữ liệu.

```
int hashCode(int key){    return key % SIZE; }
```

Hoạt động tìm kiếm trong Hash Table

Mỗi khi một phần tử được tìm kiếm: ước lượng giá trị hash code của key đã truyền vào và sau đó xác định vị trí của phần tử bởi sử dụng giá trị hash code đó giống như là chỉ mục trong mảng. Sử dụng kỹ thuật **Dò tuyến tính (Linear Probing)** để lấy phần tử nếu như không tìm thấy phần tử với giá trị hash code đã ước lượng.

```
struct DataItem *search(int key){                    //lấy giá trị hash     int
hashIndex = hashCode(key);            //di chuyển trong mảng cho tới khi gặp ô
trống     while(hashArray[hashIndex] != NULL){
if(hashArray[hashIndex]->key == key)          return hashArray[hashIndex];
```

Để theo dõi code đầy đủ các hoạt động trong Hash Table, mời bạn click chuột vào chương:**Hash Table trong C**.

Hoạt động chèn trong Hash Table

Mỗi khi một phần tử được chèn: ước lượng giá trị hash code của key đã truyền và xác định vị trí của phần tử bởi sử dụng giá trị hash code đó giống như là chỉ mục trong mảng. Sử dụng **Dò tuyến tính (Linear Probing)** cho vị trí trống nếu phần tử được tìm thấy với giá trị hash code đã ước lượng.

```
void insert(int key,int data){    struct DataItem *item = (struct DataItem*)
malloc(sizeof(struct DataItem));    item->data = data;      item->key = key;
//Lấy giá trị hash    int hashIndex = hashCode(key);     //di chuyển trong mảng
cho tới khi gặp ô trống hoặc bị xóa    while(hashArray[hashIndex] != NULL &&
hashArray[hashIndex]->key != -1){      //tới ô tiếp theo       ++hashIndex;
```

Để theo dõi code đầy đủ các hoạt động trong Hash Table, mời bạn click chuột vào chương:**Hash Table trong C**.

Hoạt động xóa trong Hash Table

Mỗi khi một phần tử cần được xóa: ước lượng giá trị hash code của key đã truyền vào và sau đó xác định vị trí của phần tử bởi sử dụng giá trị hash code đó giống như là chỉ mục trong mảng. Sử dụng **Dò tuyến tính (Linear Probing)** để lấy phần tử nếu như không tìm thấy phần tử với giá trị hash code đã ước lượng. Khi tìm thấy, lưu trữ một phần tử giả tại đây để giữ hiệu suất của hash table.

```
struct DataItem* delete(struct DataItem* item){    int key = item->key;
//lấy giá trị hash     int hashIndex = hashCode(key);     //di chuyển trong
mảng cho tới khi gặp ô trống    while(hashArray[hashIndex] !=NULL){
if(hashArray[hashIndex]->key == key){          struct DataItem* temp =
hashArray[hashIndex];                                //gán một phần tử giả tại
vị trí bị xóa          hashArray[hashIndex] = dummyItem;           return temp;
}                      //tới ô tiếp theo       ++hashIndex;
//bao quanh hash table       hashIndex %= SIZE;    }           return NULL;
}
```

Để theo dõi code đầy đủ các hoạt động trong Hash Table, mời bạn click chuột vào chương:**Hash Table trong C**.