

# Хеш-таблица. Открытая адресация

Гусев Илья, Булгаков Илья

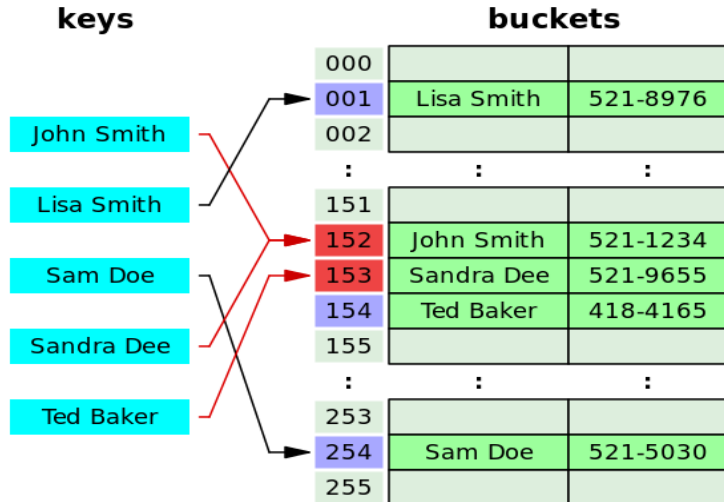
Московский физико-технический институт

Москва, 2018

# Содержание

- Открытая адресация
- Методы пробирования

# Открытая адресация



# Открытая адресация

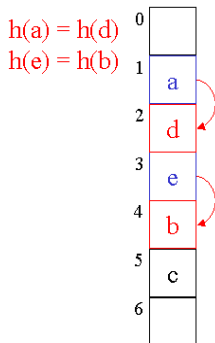
## Closed Hashing *or* Open Addressing

What if we only allow one Key at each entry?

- two objects that hash to the same spot can't both go there
- first one there gets the spot
- next one must *go in another spot*

- Properties

- $\lambda \leq 1$
- performance degrades with difficulty of finding right spot



# Методы пробирования

В случае появления коллизий надо придумать, в каком порядке просматривать ячейки. Есть несколько подходов.

- Линейное пробирование

$$H(k, i) = (\text{Hash}(k) + i) \bmod(m)$$

- Квадратичное пробирование

$$H(k, i) = (\text{Hash}(k) + C1 * i + C2 * i^2) \bmod(m)$$

Например,  $C1 = C2 = 1/2$

- Двойное хеширование

$$H(k, i) = (\text{Hash1}(k) + i * \text{Hash2}(k)) \bmod(m)$$

# Линейное пробирование

## Linear Probing

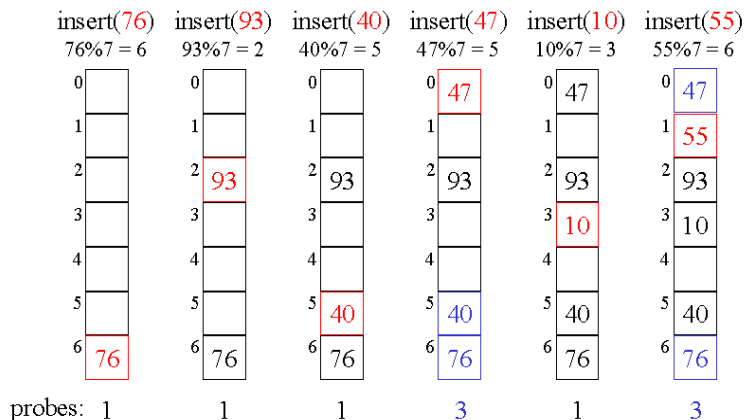
$$f(i) = i$$

- Probe sequence is
  - $h(k) \bmod \text{size}$
  - $h(k) + 1 \bmod \text{size}$
  - $h(k) + 2 \bmod \text{size}$
  - ...
- findEntry using linear probing:
 

```
bool findEntry(const Key & k, Entry *& entry) {
    int probePoint = hash1(k);
    do {
        entry = &table[probePoint];
        probePoint = (probePoint + 1) % size;
    } while (!entry->isEmpty() && entry->key != k);
    return !entry->isEmpty();
}
```

# Линейное пробирование

## Linear Probing Example



# Квадратичное пробирование

## Quadratic Probing

$$f(i) = i^2$$

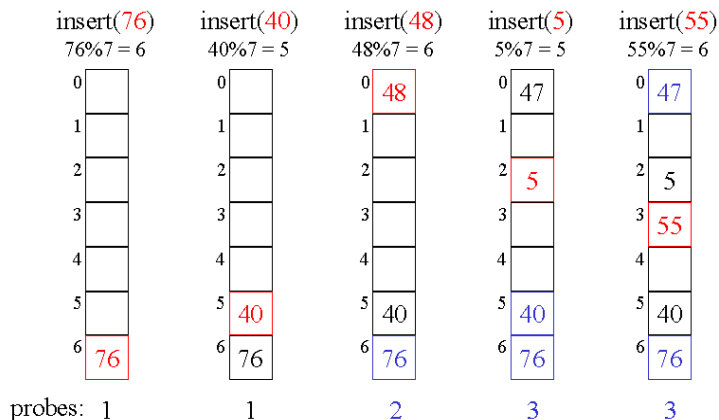
- Probe sequence is
  - $h(k) \bmod \text{size}$
  - $(h(k) + 1) \bmod \text{size}$
  - $(h(k) + 4) \bmod \text{size}$
  - $(h(k) + 9) \bmod \text{size}$
  - ...
- findEntry using quadratic probing:
 

```
bool findEntry(const Key & k, Entry *& entry) {
    int probePoint = hash1(k), numProbes = 0;
    do {
        entry = &table[probePoint];
        numProbes++;
        probePoint = (probePoint + 2*numProbes - 1) % size;
    } while (!entry->isEmpty() && entry->key != key);
    return !entry->isEmpty();
}
```



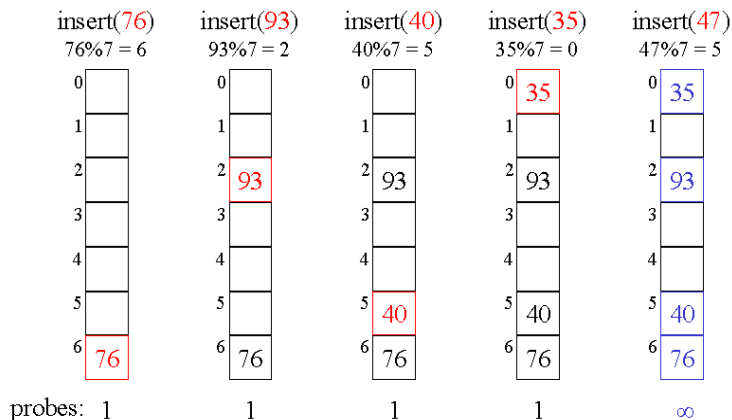
# Квадратичное пробирование

## Quadratic Probing Example ☺



# Квадратичное пробирование

## Quadratic Probing Example ☹️



# Двойное хеширование

## Double Hashing

$$f(i) = i \cdot \text{hash}_2(x)$$

- Probe sequence is
  - $h_1(k) \bmod \text{size}$
  - $(h_1(k) + 1 \cdot h_2(x)) \bmod \text{size}$
  - $(h_1(k) + 2 \cdot h_2(x)) \bmod \text{size}$
  - ...
- Code for finding the next linear probe:
 

```
bool findEntry(const Key & k, Entry *& entry) {
    int probePoint = hash1(k), hashIncr = hash2(k);
    do {
        entry = &table[probePoint];
        probePoint = (probePoint + hashIncr) % size;
    } while (!entry->isEmpty() && entry->key != k);
    return !entry->isEmpty();
}
```

insert(76)   insert(93)   insert(40)   insert(47)   insert(10)   insert(55)  
 $76\%7 = 6$     $93\%7 = 2$     $40\%7 = 5$     $47\%7 = 5$     $10\%7 = 3$     $55\%7 = 6$   
    $5 - (47\%5) = 3$                                   $5 - (55\%5) = 5$

Diagram illustrating the insertion of elements into an array over six steps:

- Step 1: Initial array state (indices 0 to 6). Element 76 is at index 6.
- Step 2: Element 93 is inserted at index 2.
- Step 3: Element 47 is inserted at index 1.
- Step 4: Element 40 is inserted at index 5.
- Step 5: Element 10 is inserted at index 3.
- Step 6: Final array state. Elements are 76, 47, 93, 10, 40, 55, and 76 at indices 0 through 6 respectively.

2