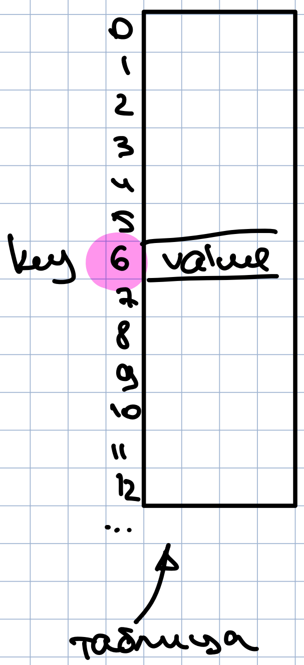
Хеш-таблица

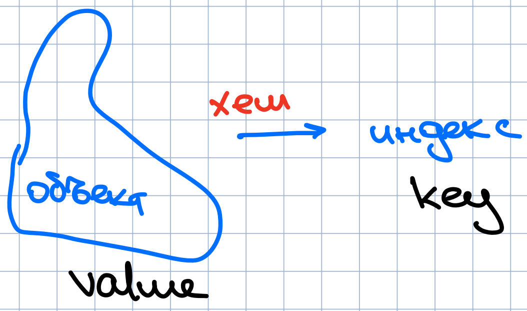
- контейнер (структура данных)

I Мотивация:

	Push	Pop	find
StaticArray	O(N)	O(N)	O(N)
List	O(1)	O(1)	O(N)
BST	O(logN)	O(logN)	O(logN)
Хеш-таблица	O(1)	O(1)	O(1)

II Как же этого добиться?





table[key] = value   O(1)

III Проблемы:

а) Прямая адресация (нет коллизий)

key — value

номер телефона — информация о человеке

table [key] = value

размер table большой

ограничение на key $\in [0, 10^7]$

Много памяти!

б) проблема коллизии

Опр Введем функцию $H : U \to \mathbb{N}$,

объекты

назовем ее хеш-функцией.

Опр Коллизия: $a \in U$ и $b \in U$

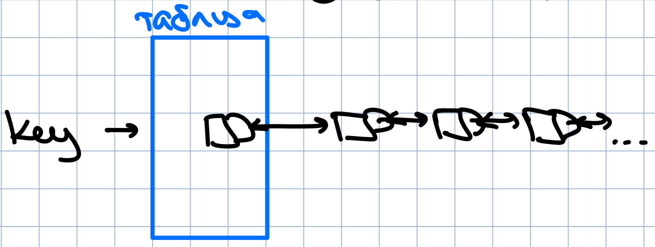
$H(a) = H(b)$ – коллизия.

Метод цепочек

Метод открытой адресации.

IV Метод цепочек

→ $a \in U$ считаем key = $H(a)$



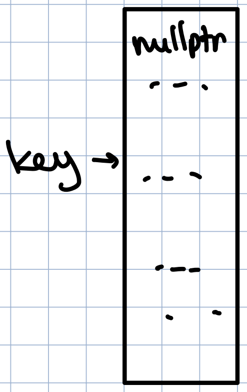
table[key] $\leftrightarrow$ двусвязный список

Операции:

дубли $O(1)$ $O(1) + \overset{\text{find}}{O(k)} = O(k)$	insert:	(table[key]).push_front(value)
$O(k)$	remove:	(table[key]).remove(value)
$O(k)$	find:	(table[key]).find(value)

k - длина цепочки, $k \sim 1$

Хранение цепочек



```
если table[key] = nullptr:
    ↳ table[key] = new List(value)
иначе если table[key].find(value):
    ↳ return
иначе
    ↳ table[key].push(value)
```

Ⓥ Хеш-функция

Ⓥ Хотим считать H(value)

Свойства:

→ меньше коллизий $\left(H(a) \overset{\text{не хотим}}{=} H(b) \right)$

→ хеш $H(a)$ отличается сильно от $H(a+1)$

Ⓥ Вычисление

1) Метод деления (для чисел)

число $a \in \mathbb{N}$

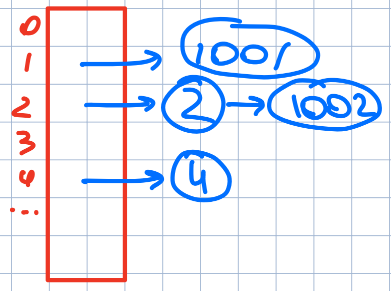
size ← размер хеш-таблицы

$H(a) = a \% \text{ size}$

Пример: таблица размера size

table [size+1] - не может быть

num → table[$\underset{\uparrow \atop H(num)}{num \% size}$] = num



size = 1000

1001 % 1000 => 1

1002 % 1000 => 2

2) Метод Умножения (чисел)

$$h(k) = \lfloor m * \underbrace{\{k * A\}}_{дробь} \rfloor, где \quad A = \frac{\sqrt{5}-1}{2}$$

m = size_table

$h(k) \in [0, \dots size - 1]$

константа Кормена

3) Полиномиальное хеширование строки

$$\underset{\uparrow \atop строка}{h(s)} = \left(\sum_{i=0}^{|s|-1} \left[\left(\underbrace{char(s[i])}_{\uparrow \atop \text{номер символа } s[i] \text{ в таблице ASCII}} * x^i \right) \% p \right] \right) \% size$$

x - число
p - число
НОД$(x,p) = 1$
желательно простые

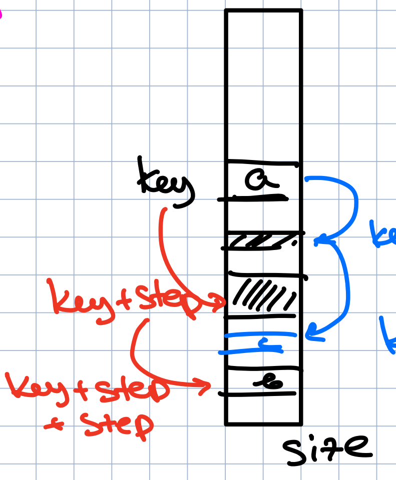
(VI) Метод открытой адресации (двух указателей)

$a \in U \rightarrow h_1(a) = key$

$h_2(a) = step$

$НОД(h_1, h_2, size) = 1$

хотим



$b, \quad h_1(b) = h_1(a)$

$c, \quad h_1(c) = h_1(a) = h_1(b)$

$h_2(c) \neq h_2(b)$

почти наверное

$i == size$ ↳ resize

$позиция = (key + step * i) \% size$

минимальное i (перебором)

Ⓥ Следить за заполнением таблицы

если заполнение 75% (50%)

↳ resize()

- увеличение таблицы в 2 раза
- пересчитываем хеш.

Методы хеш-таблицы

- find (a)

$h_1(a) \; ; \; h_2(a)$

```
i = 0 ... size-1:
    если table[ $(h_1(a) + i * h_2(a)) \% size$ ] == a ?  and table[....] не deleted
        ↳ return true
    если table[pos].state = empty
        ↳ return false
return false
```

• remove (a):

состояние ячейки → empty (nullptr)
состояние ячейки → full (есть значение)
состояние ячейки → deleted (удалено значение)

```
for i = 0 ... size - 1
    pos = (h_1(a) + h_2(a) * i) % size
    if table[pos] = a:
        ↳ table[pos].state = deleted
          return
    if table[pos].state = empty
        ↳ return
```

Node → value, state

• resize()

```
    table ← размер * 2

    for i = 0 ... old_size:
        if old_table[i].state == full:
            table.insert(old_table[i])
```

• rehash() — // — resize

только нет увеличения размера

(если deleted - элементов 50% от full - элементов)

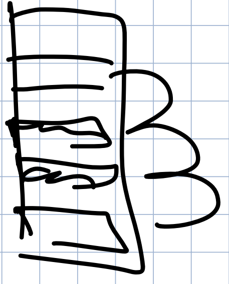
• insert (a)

! можем ! ставить вместо delete - элементов

если counter > size * 0.75 ↳ resize()

если del_counter > 0.5 * counter ↳ rehash()

```
for i = 0 ... size - 1 :
    pos = ...
    if table[pos].state = empty
        ↳ table[pos] = value
          table.[pos].state = full
```



[illegible]

```
            counter++; //← кол-во элементов в табле
else if     table[pos].state = deleted:
        if    find(value) → false
                  table[pos].state = full
                  table[pos] = value
                  del_counter --; //→ освободили deleted-элемент.
```