




Хеш-таблиці

КОЛИЧЕСТВО ЭЛЕМЕНТОВ В КНИГЕ	ПРОСТОЙ ПОИСК	БИНАРНЫЙ ПОИСК	МЭГГИ
	$O(n)$	$O(\log n)$	$O(1)$
100	10 с	1 с	МГНОВЕННО
1000	1.6 мин	1 с	МГНОВЕННО
10000	16.6 мин	2 с	МГНОВЕННО

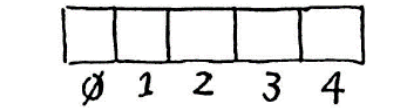
(ЯЙЦА, 2.49)	(МОЛОКО, 1.49)	(ГРУШИ, 0.79)
--------------	----------------	---------------

«НАМАСТЕ» →  → 7

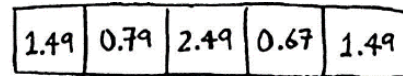
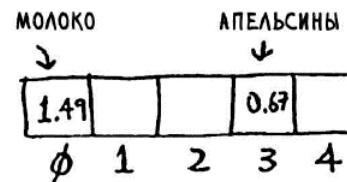
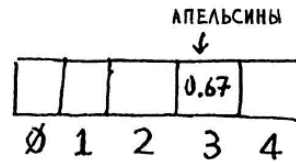
«ПРИВЕТ» →  → 4

«ХЕЛЛО» →  → 2

↑
ХЕШ-ФУНКЦИЯ

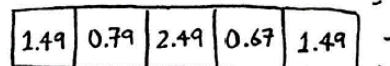


«АПЕЛЬСИНЫ» →  → 3



«АВОКАДО» →  → 4

АВОКАДО = 1.49



c++: `std::unordered_set` / `std::unordered_map`

java: `java.util.HashMap<K,V>`

c#: `System.Collections.Hashtable`, `System.Collections.Dictionary<K, V>`

python: `dict`

php: `array()`

```
>>> book = dict()
```



ПУСТАЯ
ХЕШ-ТАБЛИЦА

```
>>> book["orange"] = 0.67
```

← Апельсины стоят 67 центов

```
>>> book["milk"] = 1.49
```

← Молоко стоит 1 доллар 49 центов

```
>>> book["avocado"] = 1.49
```

```
>>> print book
```

```
{'avocado': 1.49, 'orange': 0.67, 'milk': 1.49}
```

```
>>> print book["avocado"]
```

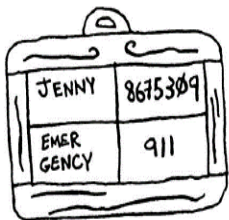
1.49 ← Цена авокадо



ХЕШ-ТАБЛИЦА С ЦЕНАМИ
НА ПРОДУКТЫ

Использование хеш-таблиц для поиска

```
>>> phone_book = dict() или >>> phone_book = {}
>>> phone_book["jenny"] = 8675309
>>> phone_book["emergency"] = 911
>>> print phone_book["jenny"]
8675309
```



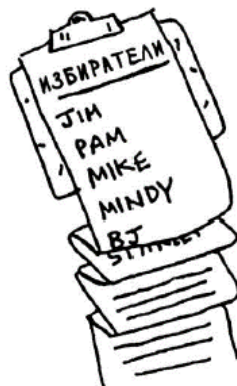
ХЕШ-ТАБЛИЦА
КАК ТЕЛЕФОННАЯ КНИГА

ADIT.10 → 173.255.248.55

GOOGLE.COM → 74.125.239.133

FACEBOOK.COM → 173.252.128.6

SCRIBD.COM → 23.235.47.175



Исключение дубликатов

```
>>> voted = {}
```

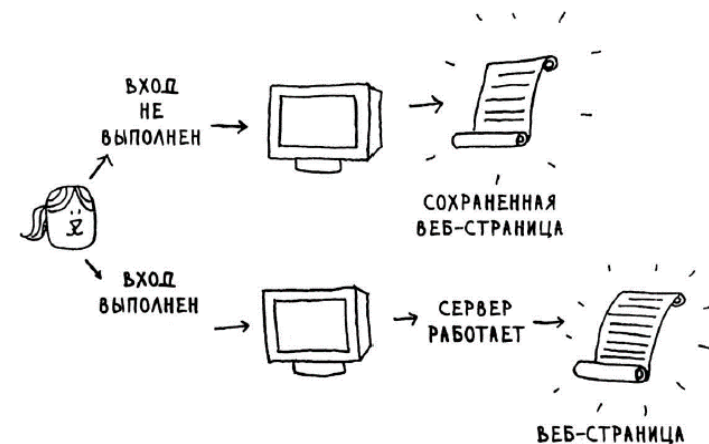
```
>>> value = voted.get("tom")
```

```
voted = {}
def check_voter(name):
    if voted.get(name):
        print "kick them out!"
    else:
        voted[name] = True
        print "let them vote!"
```

Тестируем

```
>>> check_voter("tom")
let them vote!
>>> check_voter("mike")
let them vote!
>>> check_voter("mike")
kick them out!
```

Использование хеш-таблицы как кэша

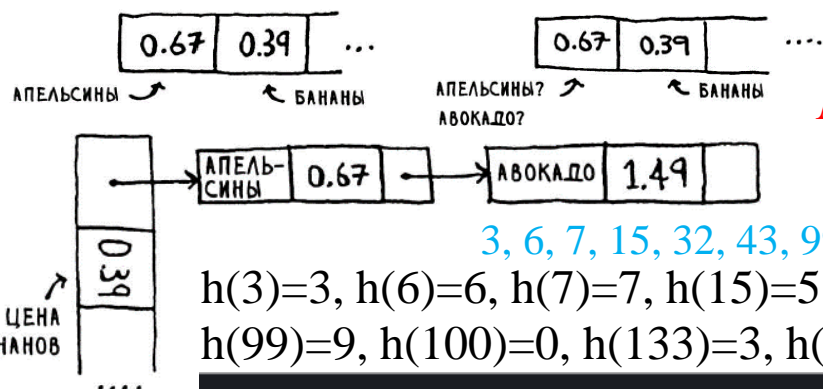
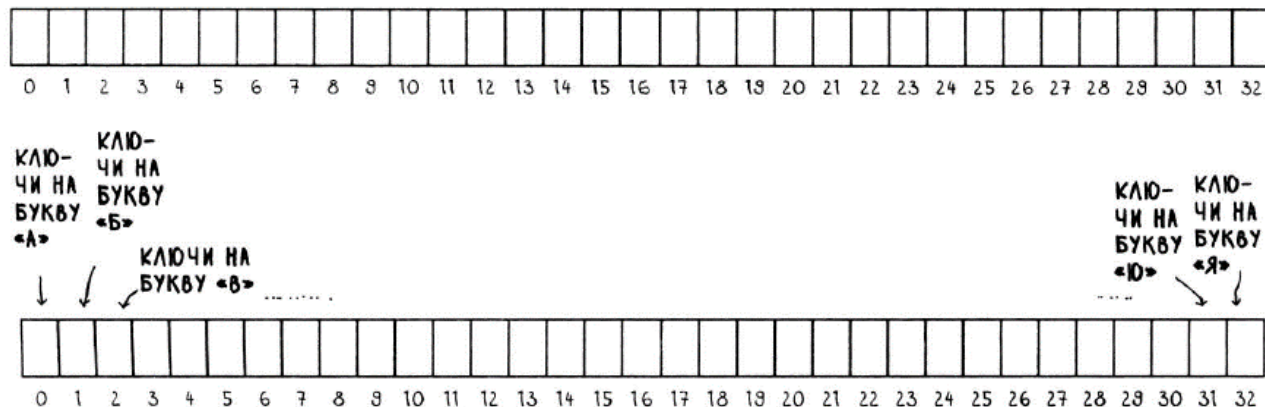


facebook.com/about → ДАННЫЕ СТРАНИЦЫ С ИНФОРМАЦИЕЙ О FACEBOOK

facebook.com → ДАННЫЕ ДОМАШНЕЙ СТРАНИЦЫ

```
cache = {}
def get_page(url):
    if cache.get(url):
        return cache[url]
    else:
        data = get_data_from_server(url)
        cache[url] = data
        return data
```

Коллизии



Метод деления

$$h(k) = k \bmod N$$

3, 6, 7, 15, 32, 43, 99, 100, 133, 158.

$$h(3)=3, h(6)=6, h(7)=7, h(15)=5, h(32)=2, h(42)=2, h(99)=9, h(100)=0, h(133)=3, h(158)=8$$

```
1 #include "stdafx.h"
2 #include <iostream>
3 using namespace std;
4 int HashFunction(int k)
5 {
6     return (k%10);
7 }
8 void main()
9 {
10     setlocale(LC_ALL, "Rus");
11     int key;
12     cout<<"Ключ > "; cin>>key;
13     cout<<"HashFunction("<<key<<"")=<<HashFunction(key)<<endl;
14     system("pause>>void");
15 }
```

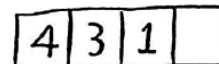
КОЛИЧЕСТВО ЭЛЕМЕНТОВ
В ХЕШ-ТАБЛИЦЕ

ОБЩЕЕ КОЛИЧЕСТВО
ЭЛЕМЕНТОВ

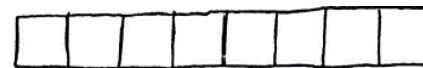
ЗАНЯТЫЕ ЭЛЕМЕНТЫ



КОЭФФИЦИЕНТ
ЗАПОЛНЕНИЯ = $\frac{2}{3}$



КОЭФФИЦИЕНТ
ЗАПОЛНЕНИЯ = $\frac{3}{4}$



КОЭФФИЦИЕНТ ЗАПОЛНЕНИЯ = $\frac{3}{6}$

Метод умножения.

$$h(k) = \lfloor N * (k * A) \rfloor \quad A = (\sqrt{5}-1)/2 \approx 0,6180339887$$

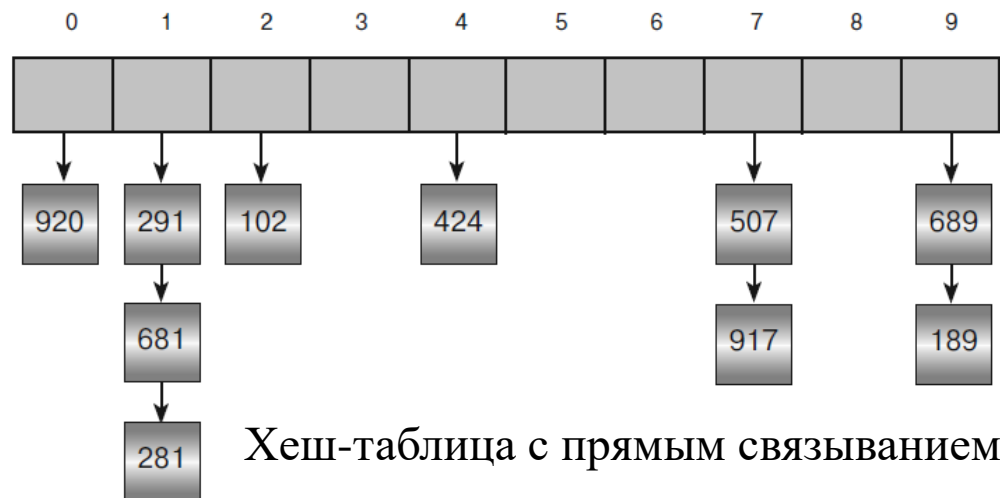
$$h(k) = \lfloor 13 * (25 * 0,618033) \rfloor = \lfloor 13 * 15,450825 \rfloor = \lfloor 13 * 0,450825 \rfloor = \lfloor 5,860725 \rfloor = 5$$

$$h(k) = \lfloor 13 * (44 * 0,618033) \rfloor = \lfloor 13 * 27,193452 \rfloor = \lfloor 13 * 0,193452 \rfloor = \lfloor 2,514876 \rfloor = 2$$

$$h(k) = \lfloor 13 * (97 * 0,618033) \rfloor = \lfloor 13 * 59,949201 \rfloor = \lfloor 13 * 0,949201 \rfloor = \lfloor 12,339613 \rfloor = 12$$

```
1 #include "stdafx.h"
2 #include <iostream>
3 using namespace std;
4 int HashFunction(int k)
5 {
6     int N=13; double A=0.618033;
7     int h=N*fmod(k*A, 1);
8     return h;
9 }
10 void main()
11 {
12     setlocale(LC_ALL, "Rus");
13     int key;
14     cout<<"Ключ > "; cin>>key;
15     cout<<"HashFunction("<<key<<"")=<<HashFunction(key)<<endl;
16     system("pause>>void");
17 }
```

Прямое связывание



Псевдослучайное пробирование

$$K, K + p, K + 2p$$

Двойное хеширование

$$K, K + p, K + 2p, K + 3p$$

$$p_A = F_1(A) \text{ и } p_B = F_2(B)$$

Открытая адресация

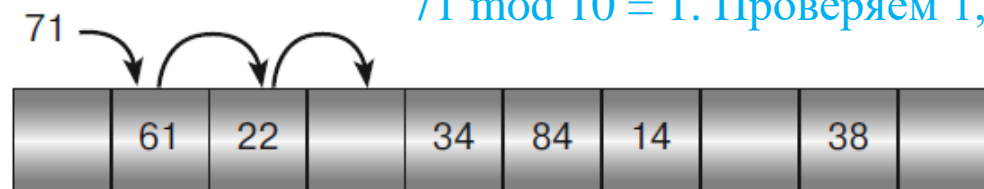
Линейное пробирование

$$K, K + 1, K + 2, K + 3,$$

$$N \bmod 100$$

2197 проверяет ячейки 97, 98, 99, 0, 1, 2

$$71 \bmod 10 = 1. \text{ Проверяем } 1, 2, 3, 4$$



Квадратичное пробирование

$$K, K + 1^2, K + 2^2, K + 3^2$$

71 пробная последовательность 1, $1 + 1^2 = 2$, $1 + 2^2 = 5$, $1 + 3^2 = 10$

93 пробная последовательность 3, $3 + 1^2 = 4$, $3 + 2^2 = 7$

