Хэширование

Что такое хэш-функция?

Хэширование

Что такое хэш-функция?

много объектов определенного типа

Множество целых чисел

Хэширование

Как используем?

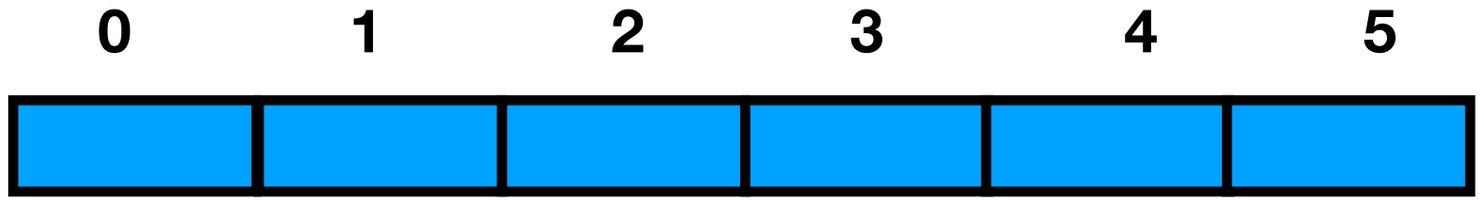
Хэширование

Как используем?

1) Хэш-таблицы, множества
2) Фильтр Блума
3) Алгоритм Рабина-Карпа (следующий семинар)
4) Много где еще

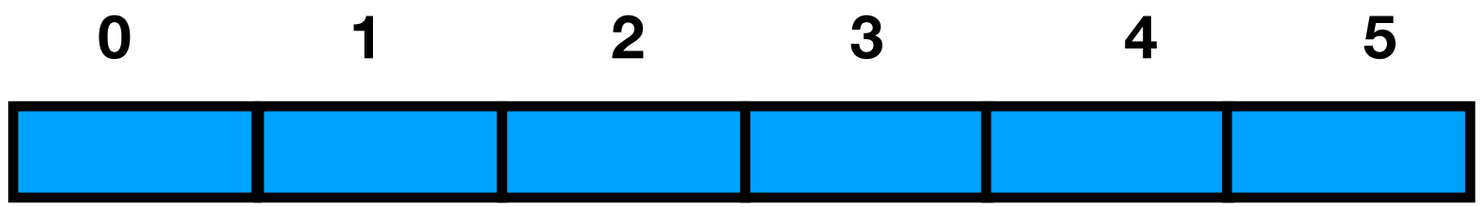
Хэш-таблица (закрытая адресация)

Есть объекты, которые хотим хранить в массиве списков



Хэш-таблица (закрытая адресация)

Есть объекты, которые хотим хранить в массиве списков



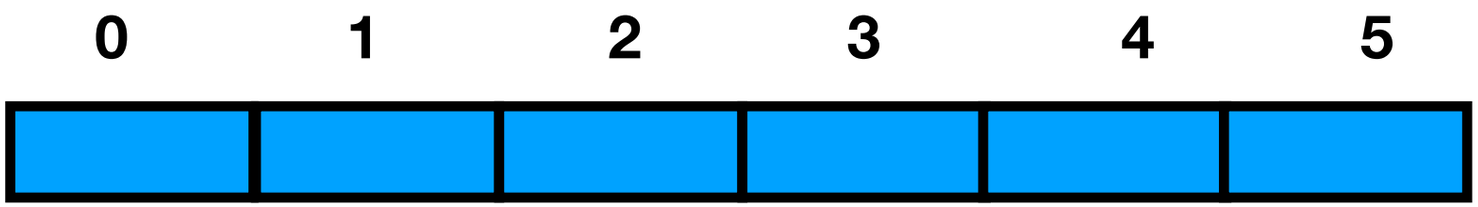
Вычислим для них значения хэш-функции, которую мы взяли

Ключ	Значение	h(x)
“кошка”	72	20
“кот”	26	17
“мейнкун”	44	42
“пушистик”	19	26

Что делать дальше?

Хэш-таблица (закрытая адресация)

Есть объекты, которые хотим хранить в массиве списков



Вычислим для них значения хэш-функции, которую мы взяли

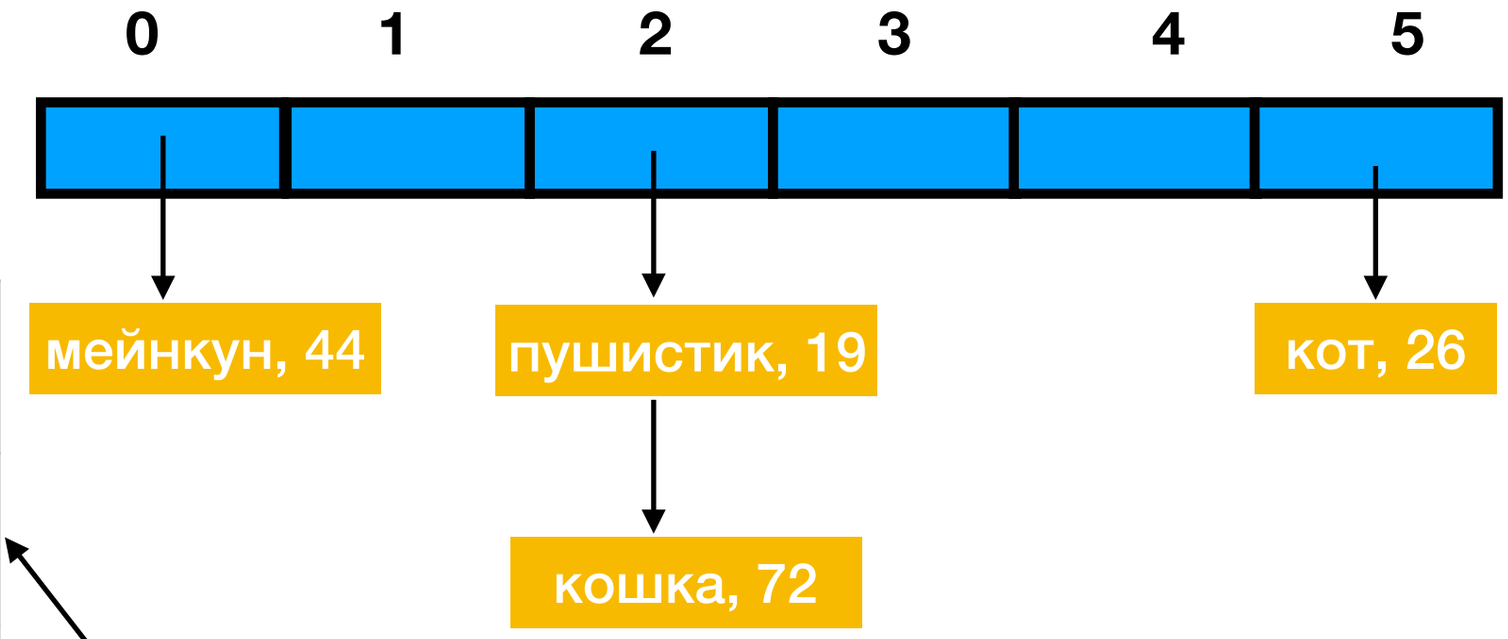
Объект	Значение	h(x)	h(x) mod 6
“кошка”	72	20	2
“кот”	26	17	5
“мейнкун”	44	42	0
“пушистик”	19	26	2

Что делать дальше?

Берем остаток от деления на размер массива

Хэш-таблица (закрытая адресация)

Вставляем объекты в начало связного списка, находящегося в ячейке массива с номером равным остатку.

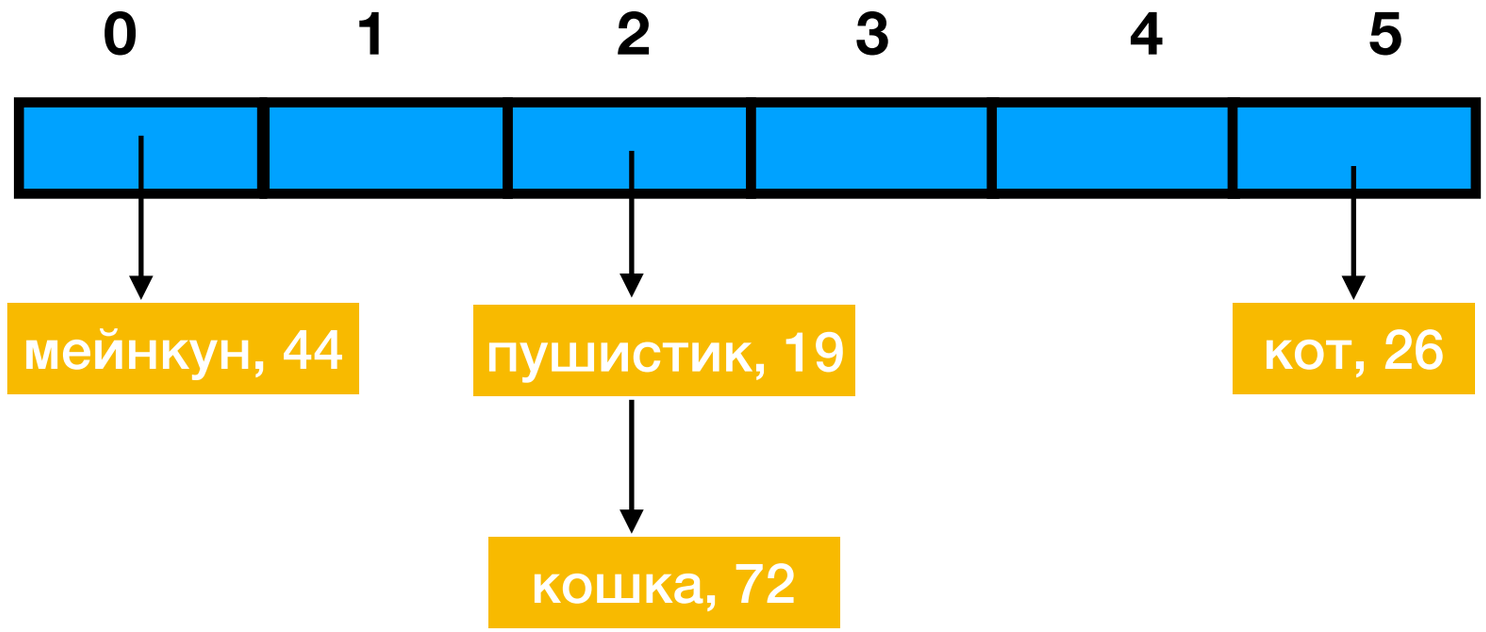


Объект	Значение	h(x)	h(x) mod 6
“кошка”	72	20	2
“кот”	26	17	5
“мейнкун”	44	42	0
“пушистик”	19	26	2

Коллизия, два объекта попадут в одну ячейку

Хэш-таблица (закрытая адресация)

Вставляем объекты в начало связного списка, находящегося в ячейке массива с номером равным остатку.

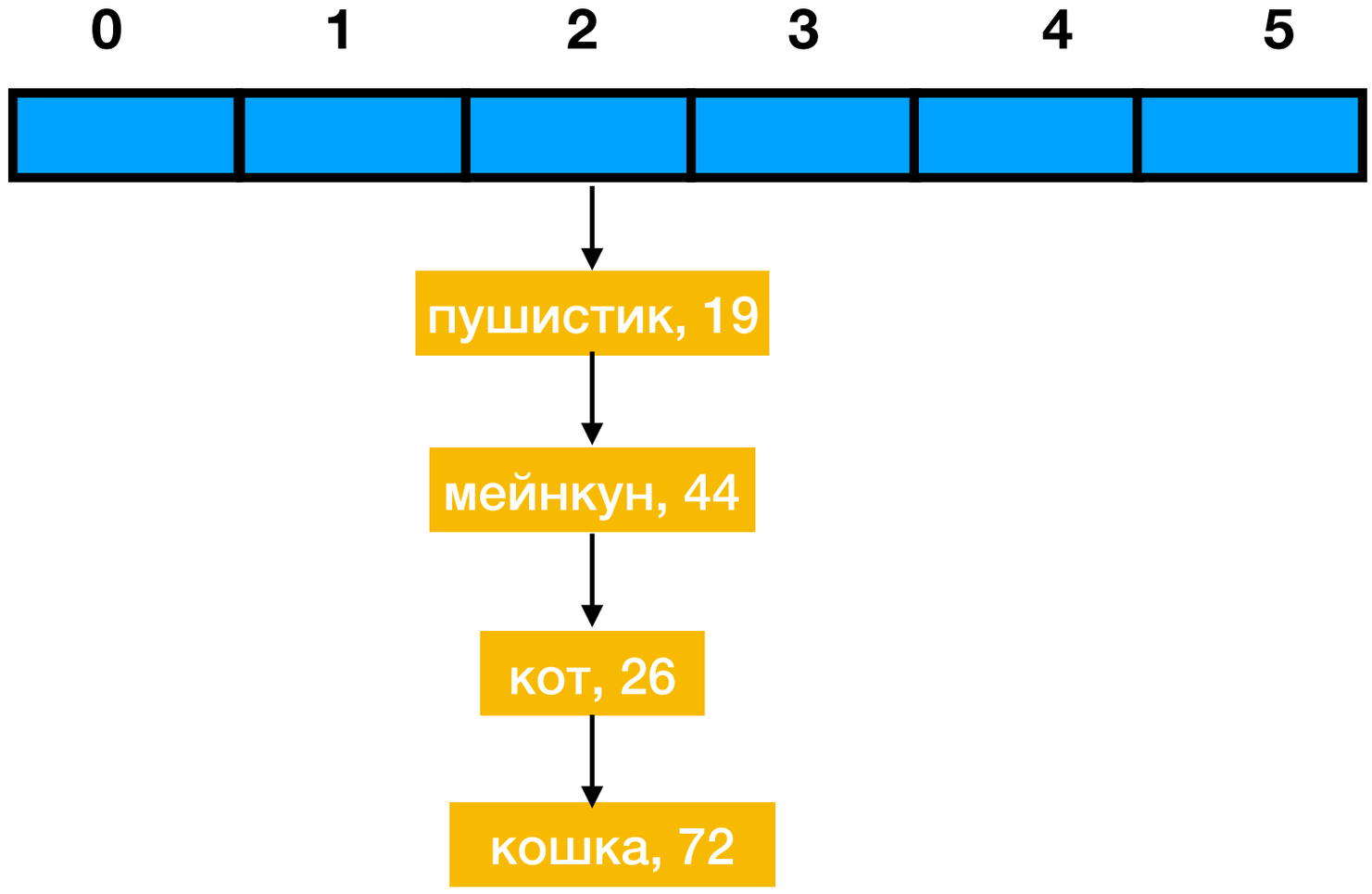


Объект	Значение	h(x)	h(x) mod 6
“кошка”	72	20	2
“кот”	26	17	5
“мейнкун”	44	42	0
“пушистик”	19	26	2

Какие требования к хэш-функции?

Хэш-таблица (закрытая адресация)

Вставляем объекты в начало связного списка, находящегося в ячейке массива с номером равным остатку.



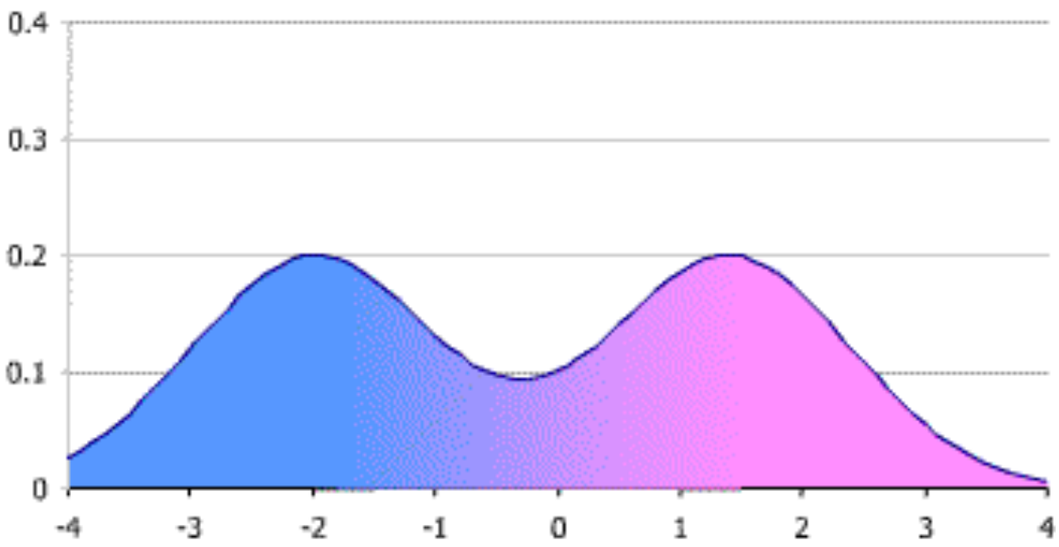
Объект	Значение	h(x)	h(x) mod 6
“кошка”	72	20	2
“кот”	26	20	2
“мейнкун”	44	20	2
“пушистик”	19	20	2

Она не должна давать много коллизий

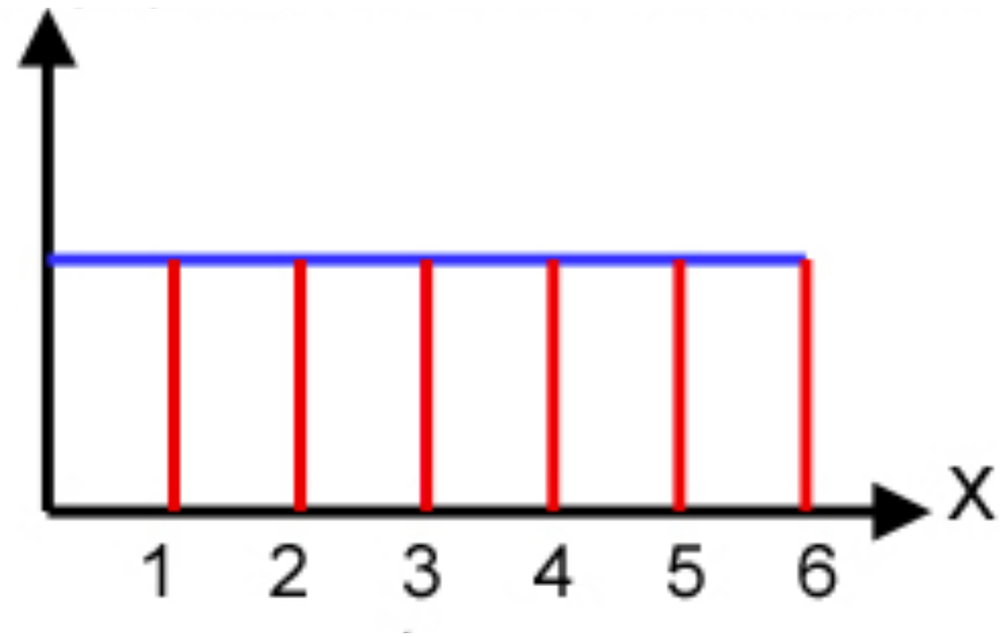
Хорошая хэш-функция

1) **Быстро вычисляется**
2) **Дает мало коллизий**

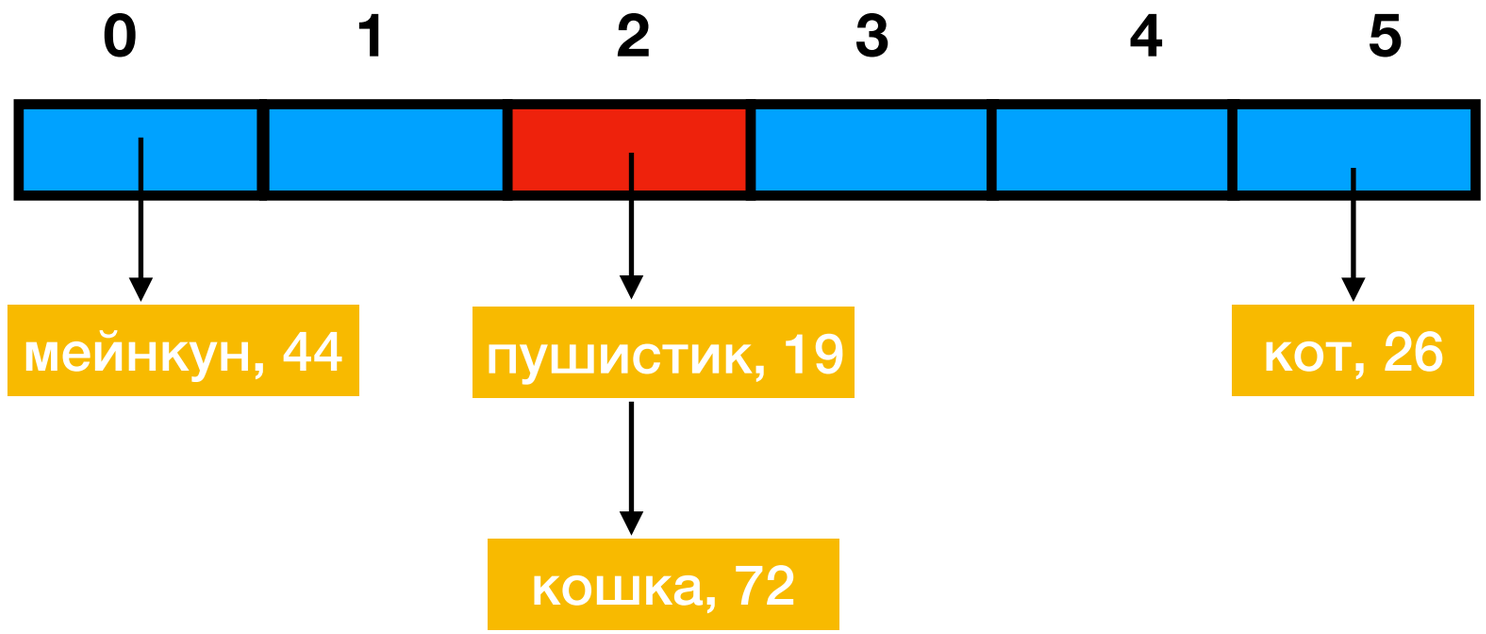
Распределение данных, с которыми мы работаем



Распределение h(x) mod (размер таблицы)



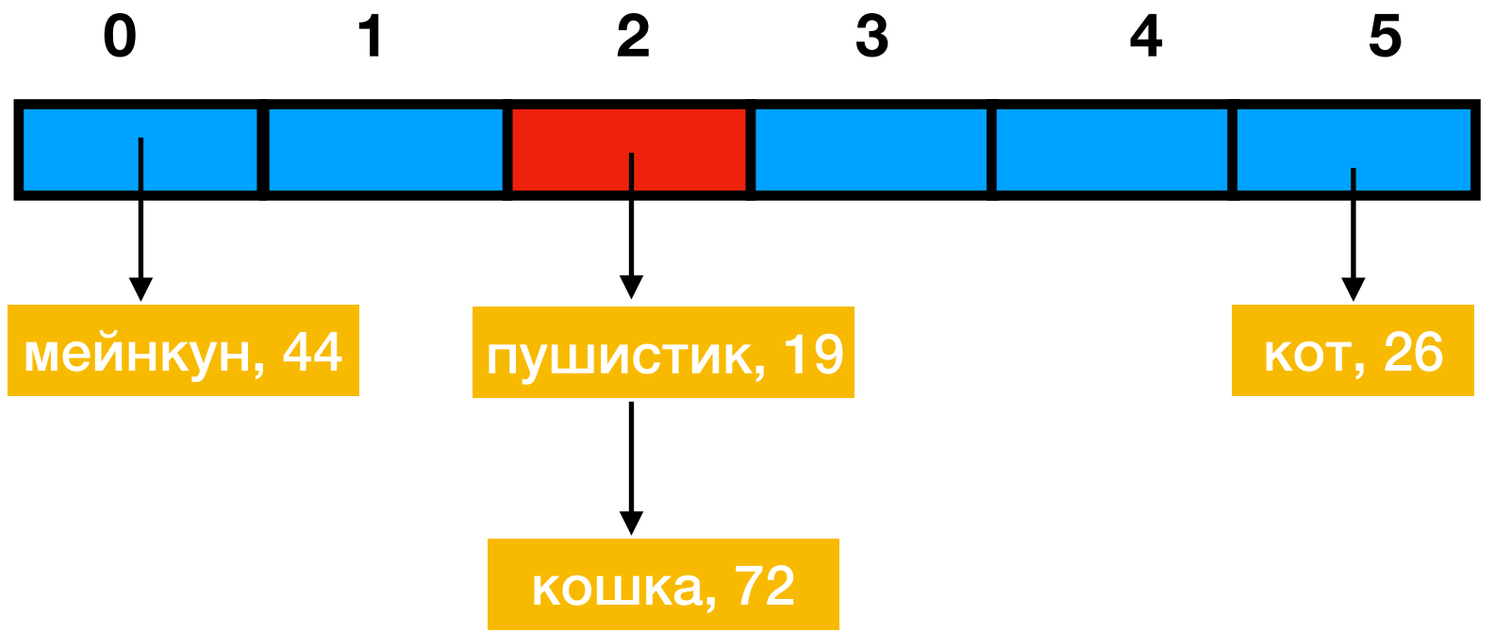
Как искать?



Ищем пушистик:

1) **h(пушистик) % 6= 26 % 6 = 2**
2) **Проходим по односвязному списку, который расположен во второй ячейке, пока не встретим слово пушистик.**
3) **Если встретили, возвращаем значение**
4) **Не встретили - сообщаем, что в таблице пушистика нет**

Как искать?

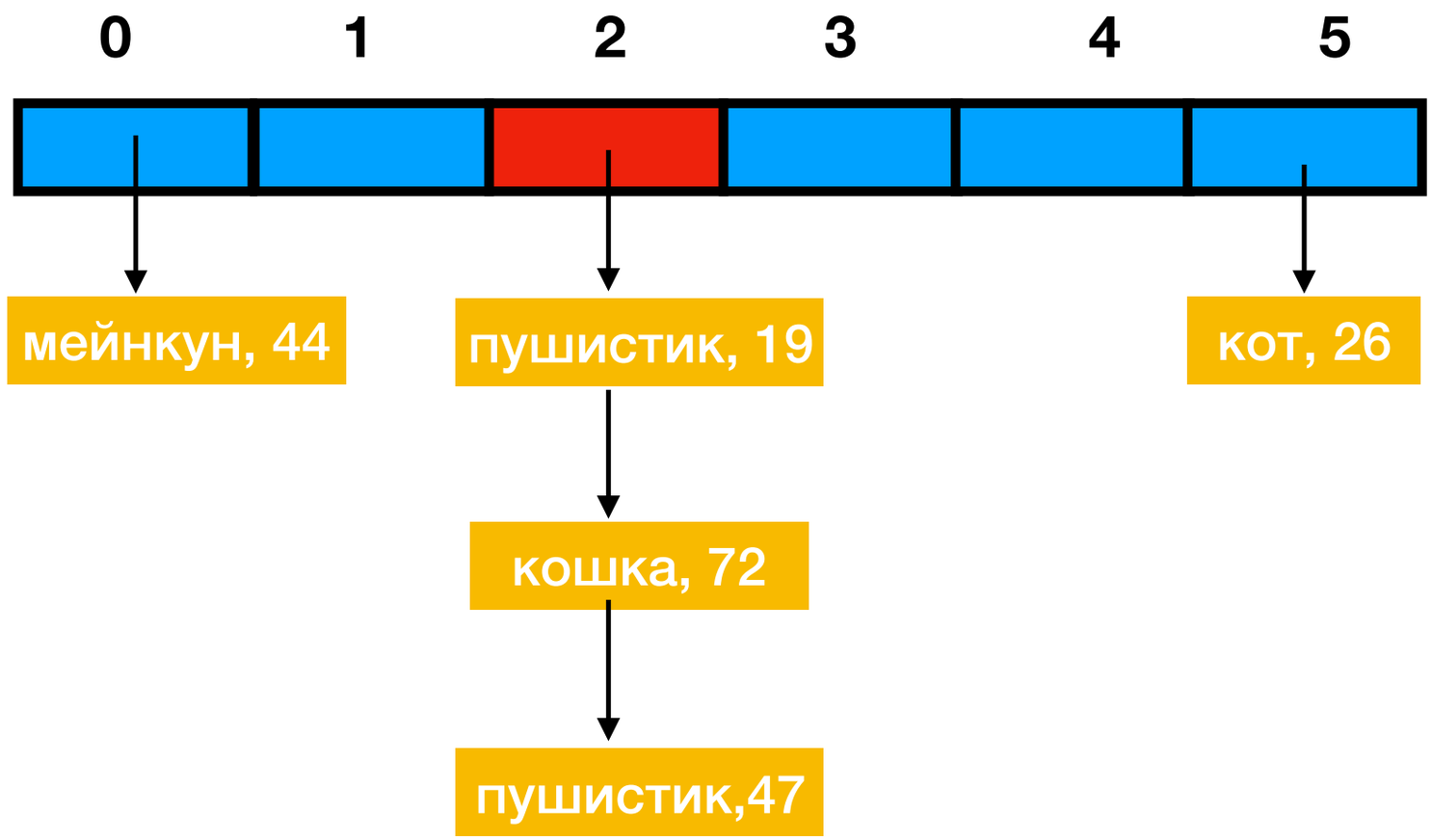


Ищем пушистик:

1) **h(пушистик) % 6= 26 % 6 = 2**
2) **Проходим по односвязному списку, который расположен во второй ячейке, пока не встретим слово пушистик.**
3) **Если встретили, возвращаем значение**
4) **Не встретили - сообщаем, что в таблице пушистика нет**

Есть ли проблема?

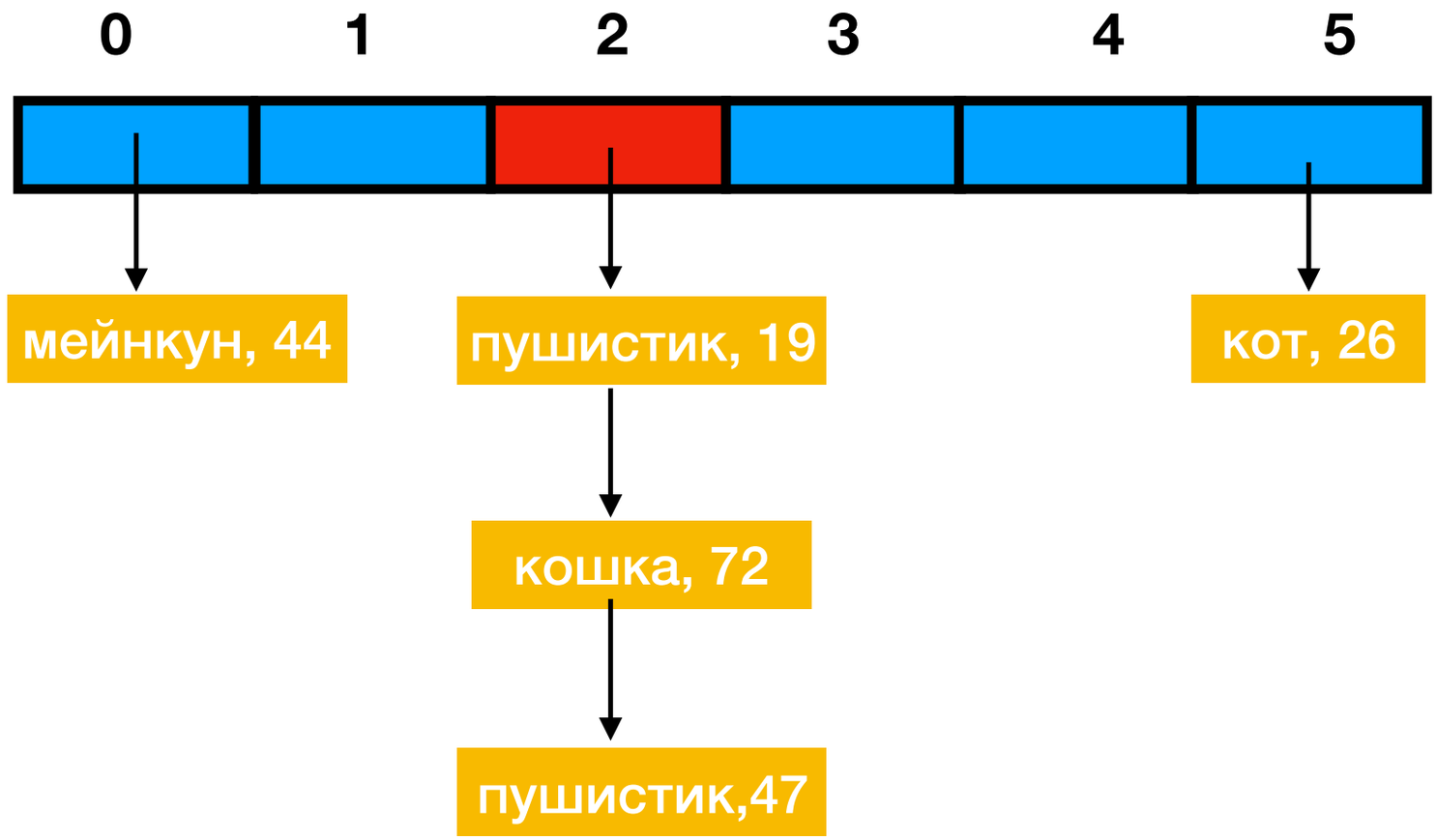
Как искать?



Есть ли проблема?

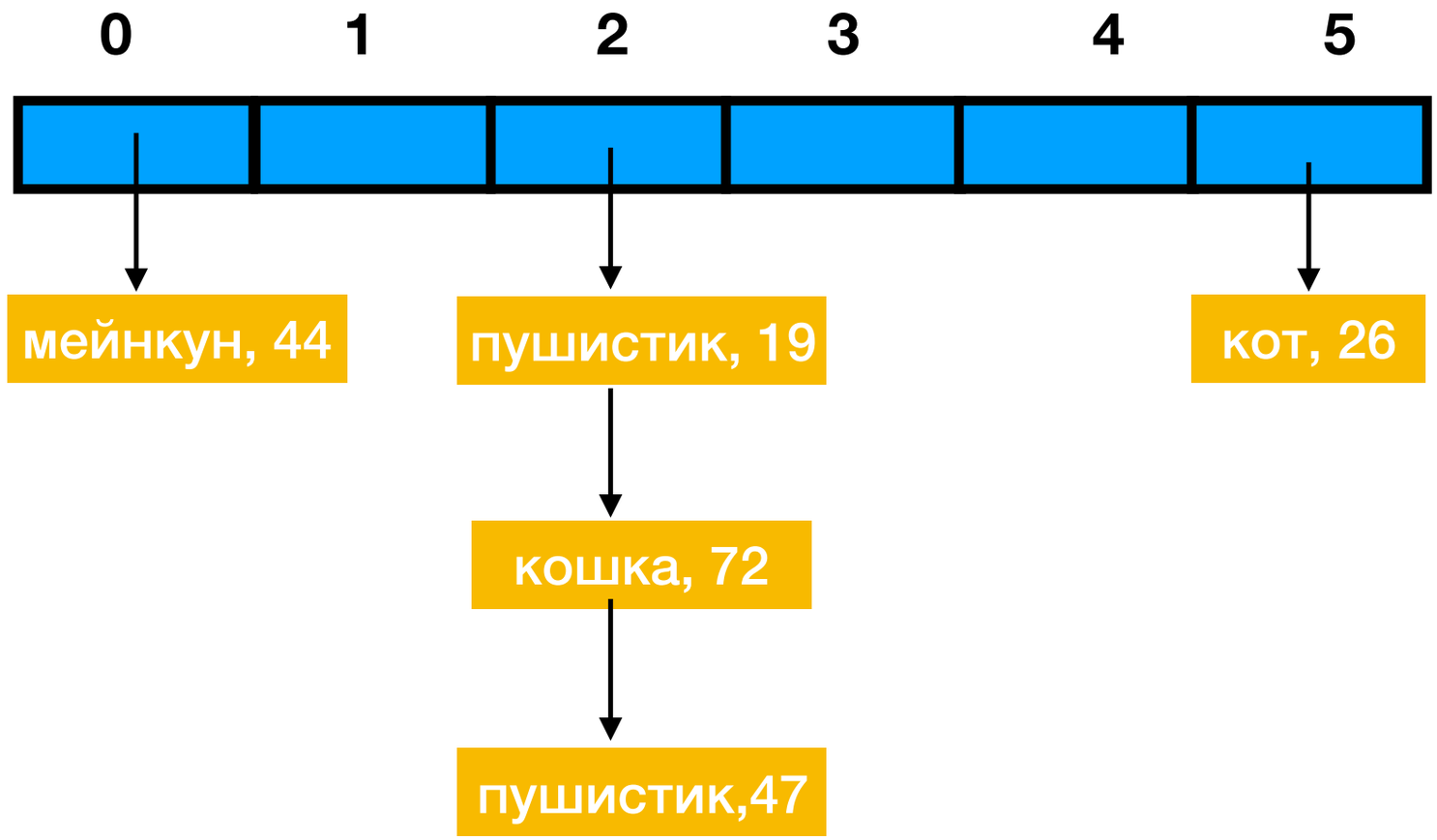
Если слово пушистик вставляли два раза, то найдем то, что вставили вторым. Это не проблема, а то, как обычно и работают хэш-таблицы

Как искать?



Какова в среднем длина списка в таблице размера N, в которую мы вставили M записей?

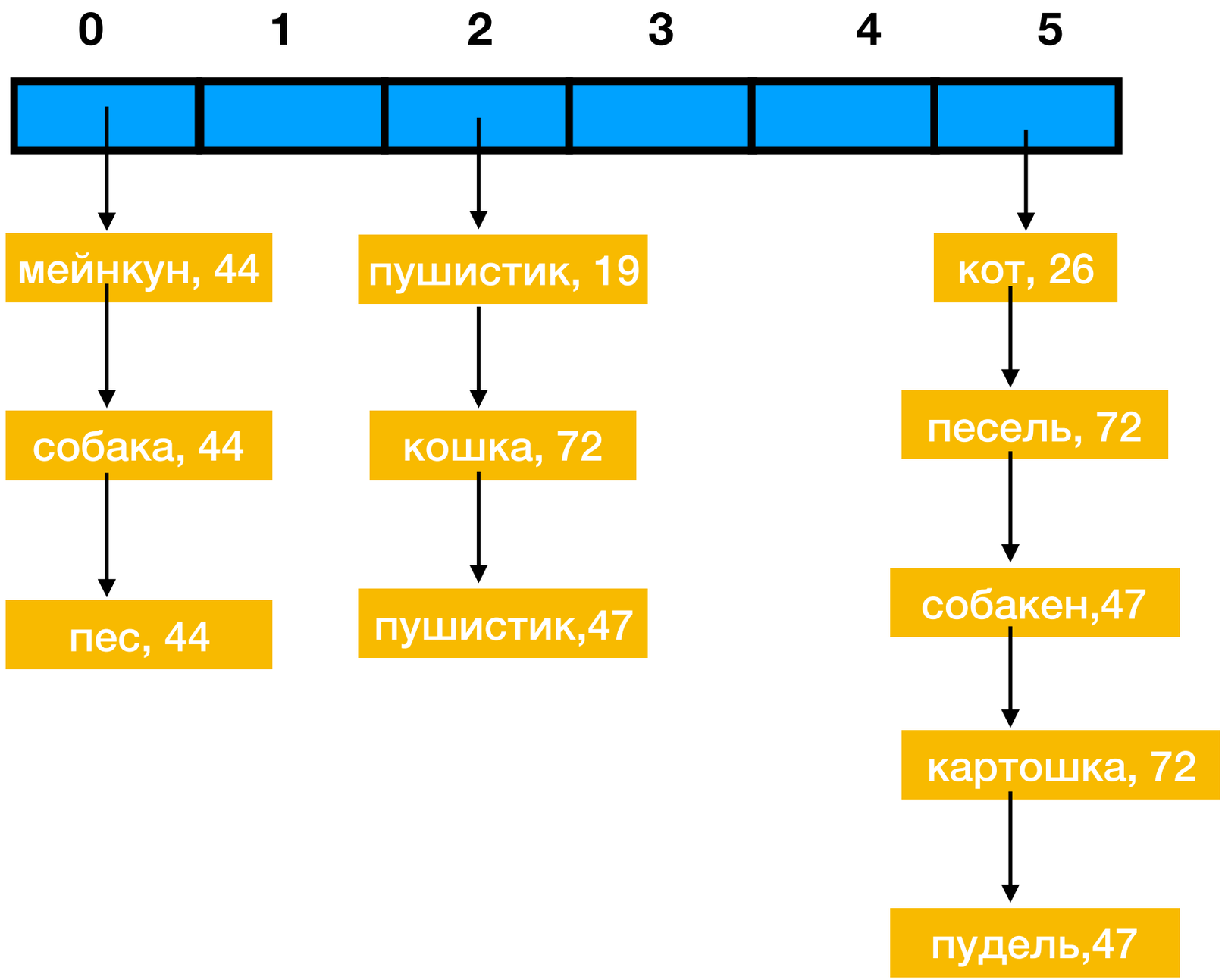
Как искать?



Какова в среднем длина списка в таблице размера N, в которую мы вставили M записей?

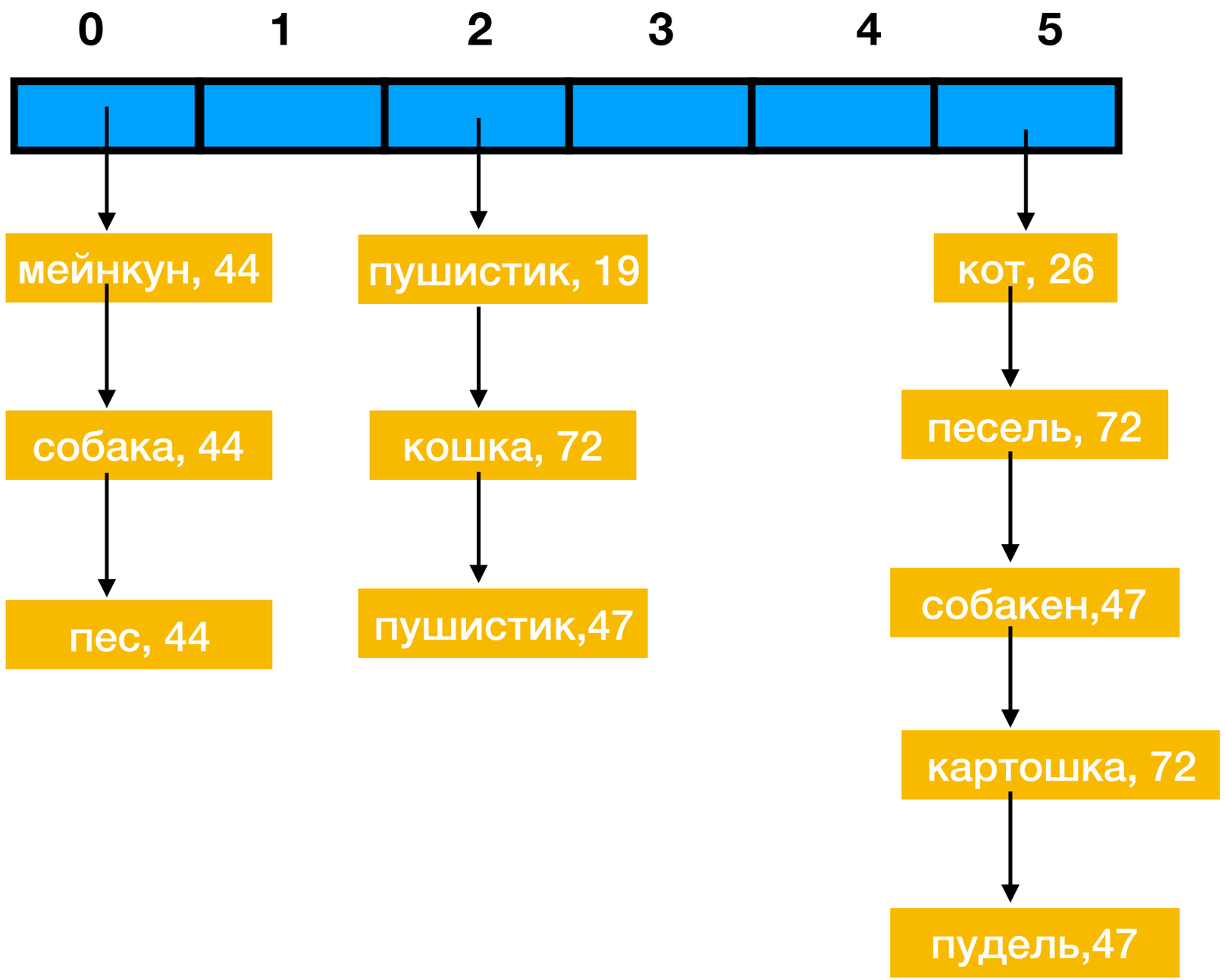
M/N

Как искать?



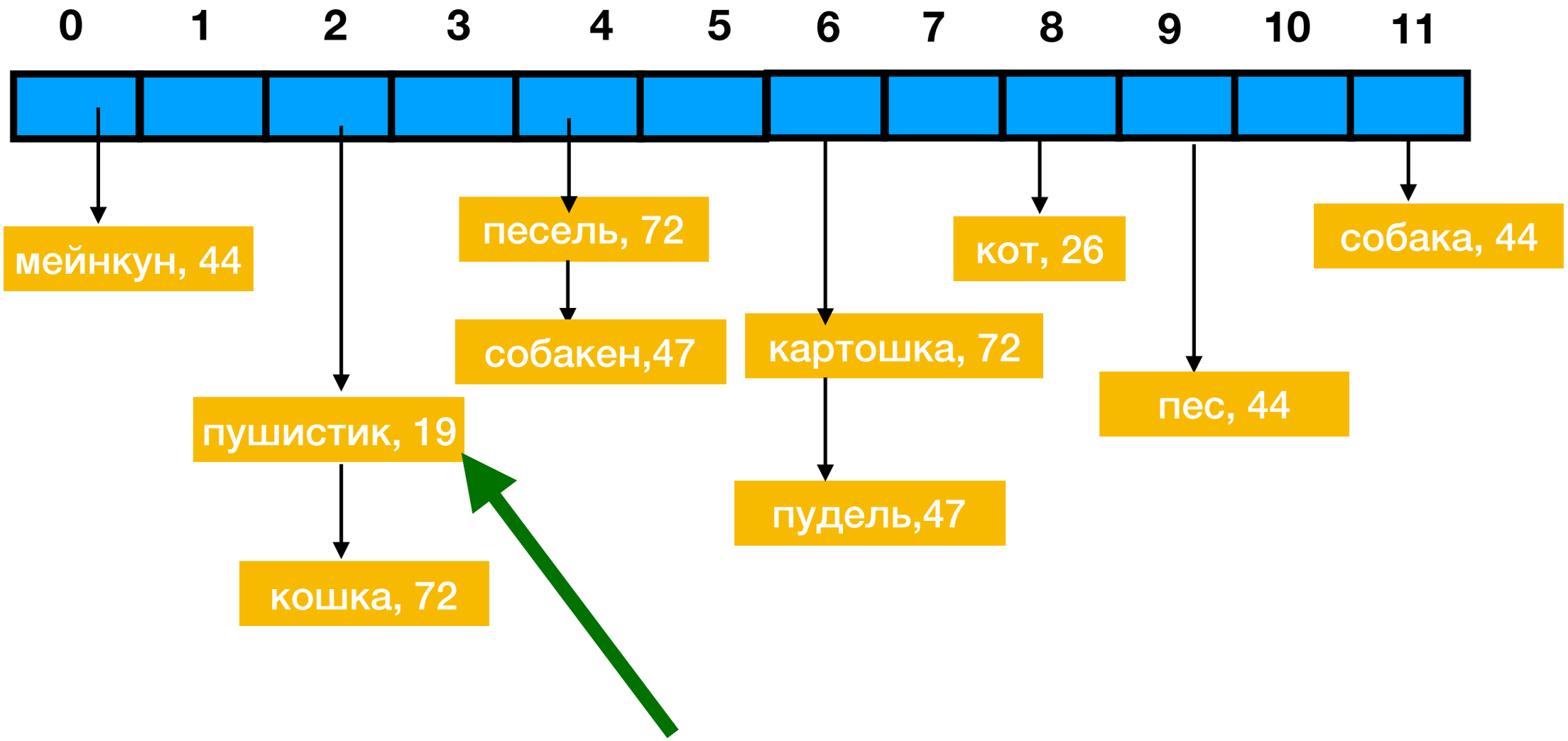
Что делать, когда записей становится очень много, а таблица маленькая? Или даже один из списков стал очень длинным?

Как искать?



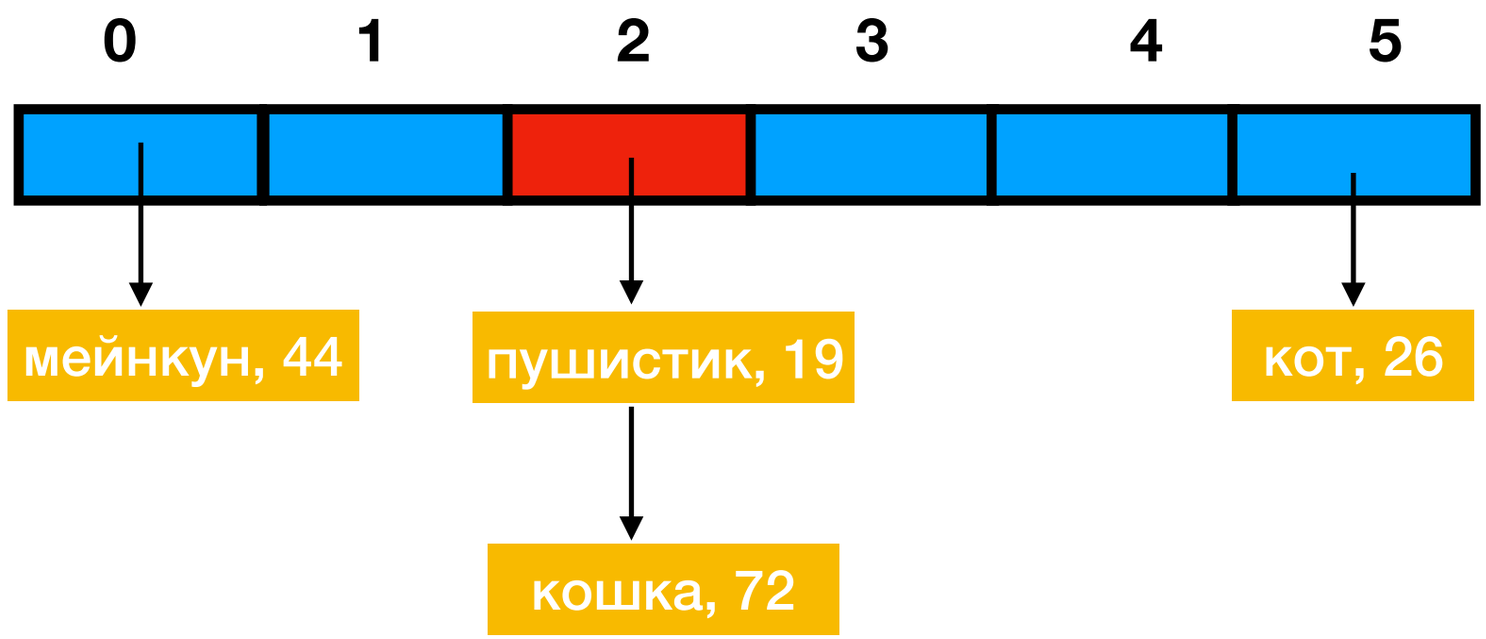
Делаем таблицу бОльшего размера (в 2 раза, например) и переставляем ключи туда.

Как искать?



При этом мы можем избавиться от дубликатов старых ключей

Как удалять?



Удаляем пушистик:

1) **h(пушистик) % 6= 26 % 6 = 2**
2) **Проходим по ВСЕМУ односвязному списку, который расположен во второй ячейке, и удалям ВСЕ вхождения пушистика**

Какое время работы?

Операция	Средний случай / амортизированный	Худший случай
Вставка ключ-значение	1	O(M) + O(N) (если мы увеличиваем размер таблицы)
Поиск ключ-значение	O(L) = O(M/N)	O(M)
Удаление ключа	O(L) = O(M/N)	O(M)

Можно ли получить объект с такими же операциями, но с logN на все операции?

Какое время работы?

Операция	Средний случай / амортизированный	Худший случай
Вставка ключ-значение	1	O(M) + O(N) (если мы увеличиваем размер таблицы)
Поиск ключ-значение	O(L) = O(M/N)	O(M)
Удаление ключа	O(L) = O(M/N)	O(M)

Можно ли получить объект с такими же операциями, но с logN на все операции?

Да, можно использовать красно-черное дерево для хранения пар ключ-значение

Пишем код поиска ключа в таблицу

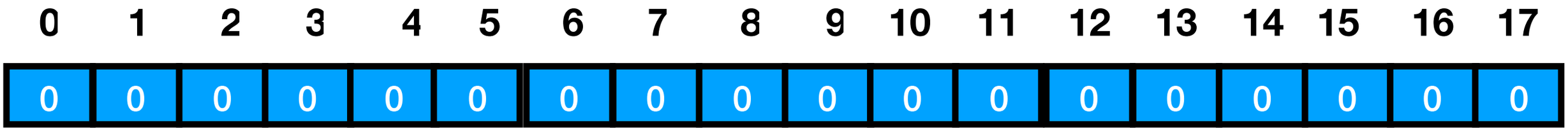
Пишем код поиска ключа в таблицу

```
fn find(hm: HashMap, key: Key) -> Union<Value, None>{
    index = hm.hash_func(key) % hm.length();
    head = hm.table[index];
    cur_elem = head;
    while cur_elem != None{
        if cur_elem.key == key{
            return value
        }
        cur_elem = cur_elem.next;
    }
    return None;
}
```

Фильтр Блума

1) Представим себе ситуацию, когда бОльшая часть вашей структуры, хранящей данные по ключу, лежит на жестком диске;
2) В этом случае искать в этой структуре что-то по ключу - долго - чтение с жесткого диска - дорого
3) Хотим научиться быстро понимать, лежит у нас какой-то ключ на жестком диске или нет.

Фильтр Блума



Делаем набор хэш функций. Самый “тупой способ”:

$$h_0 = h(x)$$

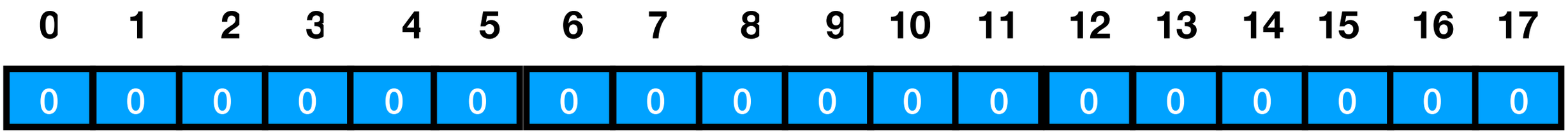
$$h_1 = h(h_0(x))$$

$$h_2 = h(h_1(x))$$

$$\dots$$

$$h_k = h(h_{k-1}(x))$$

Фильтр Блума



Теперь нам приходит слово котяра.
Считаем для него значения каждой хэш-функции, пусть у нас их три

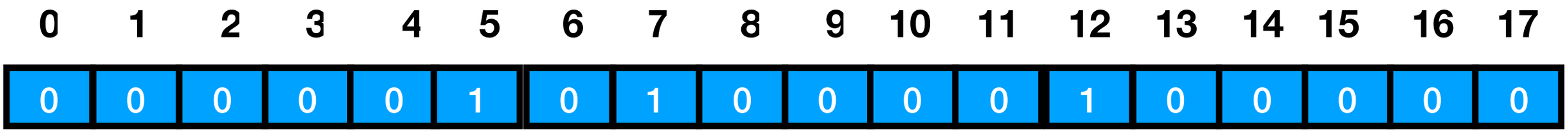
$$h_0(x) = 5$$

$$h_1(x) = 7$$

$$h_2(x) = 12$$

Ставим в таблице 5, 7 и 12 значения 1

Фильтр Блума



Теперь нам приходит слово собакен.
Считаем для него значения каждой хэш-функции, пусть у нас их три

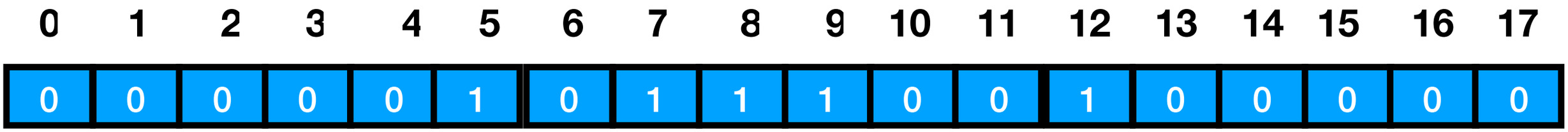
$$h_0(x) = 8$$

$$h_1(x) = 12$$

$$h_2(x) = 9$$

Ставим в таблице 8, 12 и 9 значения 1

Фильтр Блума



Хотим узнать, если ли слово котяра в фильтре?

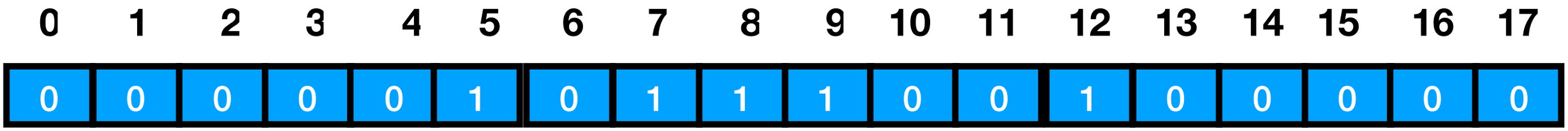
$h_0(x) = 5$

$h_1(x) = 7$

$h_2(x) = 12$

На 5, 7 и 12 позициях стоят 1 - значит, есть

Фильтр Блума



Хотим узнать, если ли слово кошечка в фильтре?

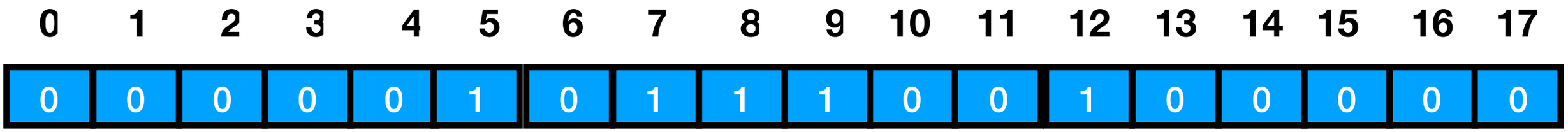
$h_0(x) = 5$

$h_1(x) = 0$

$h_2(x) = 12$

На 0 позиции нет - значит, нет

Фильтр Блума



Хотим узнать, если ли слово сапог в фильтре?

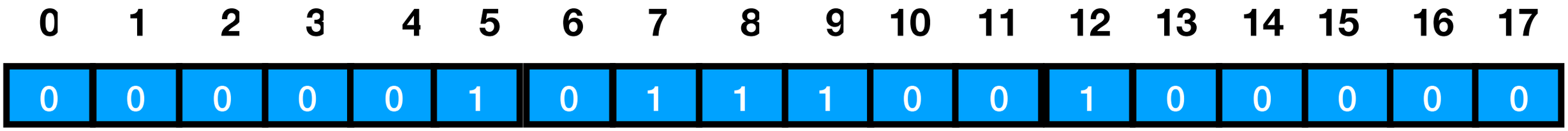
$$h_0(x) = 5$$

$$h_1(x) = 8$$

$$h_2(x) = 12$$

На позициях 5, 8 и 12 стоят 1. Значит, есть?

Фильтр Блума



Хотим узнать, если ли слово сапог в фильтре?

$$h_0(x) = 5$$

$$h_1(x) = 8$$

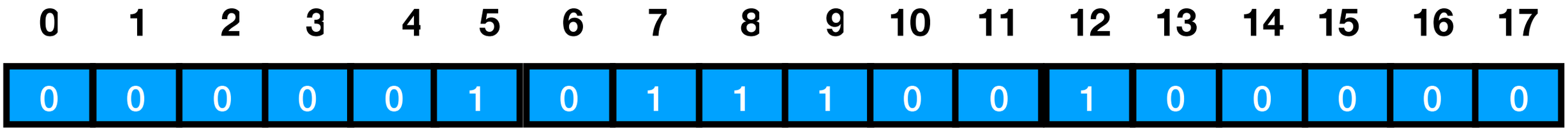
$$h_2(x) = 12$$

На позициях 5, 8 и 12 стоят 1. Значит, есть?

Фильтр Блума может давать ложно-положительные результаты. Когда говорится, что объект есть, а на самом деле его нет.

Может ли он давать ложно-отрицательные результаты?

Фильтр Блума



Хотим узнать, если ли слово сапог в фильтре?

$$h_0(x) = 5$$

$$h_1(x) = 8$$

$$h_2(x) = 12$$

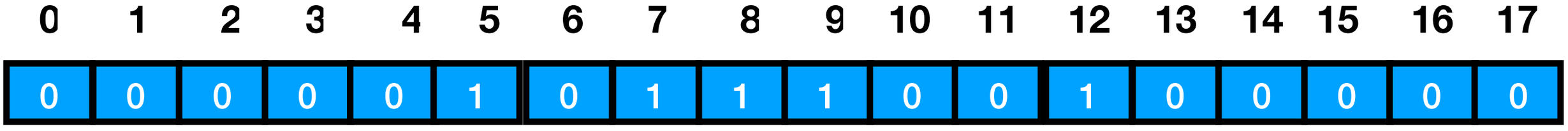
На позициях 5, 8 и 12 стоят 1. Значит, есть?

Фильтр Блума может давать ложно-положительные результаты. Когда говорится, что объект есть, а на самом деле его нет.

Может ли он давать ложно-отрицательные результаты?

Нет. Мы никогда не стираем поставленные 1

Фильтр Блума



Как использовать?

1. **Над нашей большой и хранящейся частично на жестком диске структурой делаем фильтр Блума. Его можно хранить в виде битов, занимает крайне мало места в оперативной памяти.**
2. **Когда ищем что-то в нашей структуре, сначала спрашиваем фильтр Блума, а есть ли оно там?**
3. **Если он ответил, что такого ключа нет, то его точно нет. Иначе - смотрим в структуру**

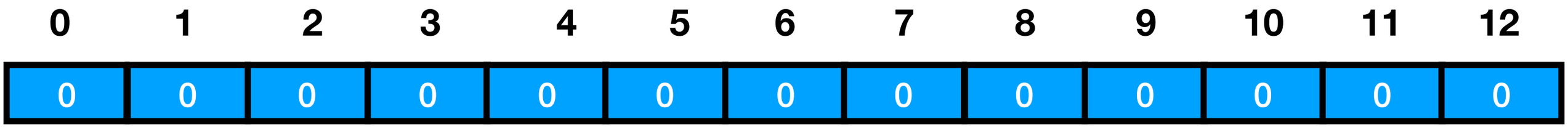
Хэш-таблица с открытой адресацией (вариант приблизительно Python)

https://stackoverflow.com/questions/327311/how-are-pythons-built-in-dictionaries-implemented

У нас есть таблица, где мы просто храним записи ключ-значения. В начале в ней занят только 0 слот, там храним NULL

Номер	Hash	Key	Value
0	NULL	NULL	NULL
1			
2			
3			
4			
5			
6			
7			

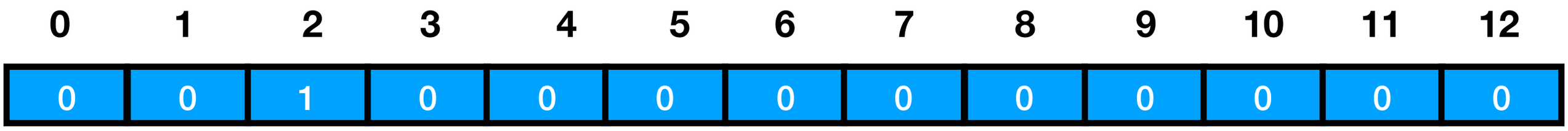
У нас есть просто массив целых неотрицательных чисел, сначала все 0 (по сути это наша хэш-таблица)



Приходит пара (пушистик, 22)
h(x) = 15
Вставляем эту информацию в таблицу

Номер	Hash	Key	Value
0	NULL	NULL	NULL
1	15	пушистик	22
2			
3			
4			
5			
6			
7			

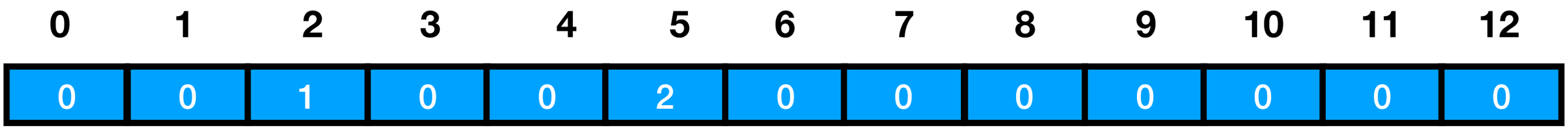
15 % 13 = 2 -> пробуем в ячейку 2 записать номер строки таблицы.
Удачно



Приходит пара (котейка, 48)
h(x) = 5
Вставляем эту информацию в таблицу

Номер	Hash	Key	Value
0	NULL	NULL	NULL
1	15	пушистик	22
2	5	котейка	48
3			
4			
5			
6			
7			

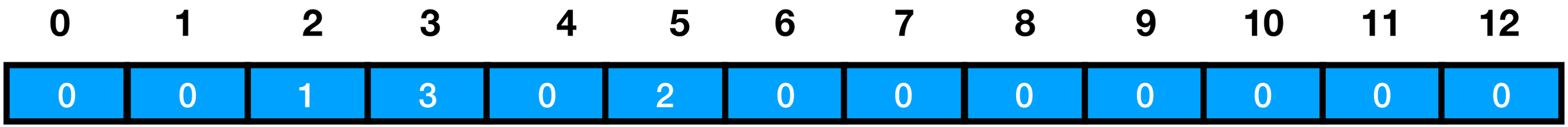
5 % 13 = 5 -> пробуем в ячейку 5 записать номер строки таблицы.
Удачно



Приходит пара (собакен, 24)
h(x) = 2
Вставляем эту информацию в таблицу

Номер	Hash	Key	Value
0	NULL	NULL	NULL
1	15	пушистик	22
2	5	котейка	48
3	2	собакен	24
4			
5			
6			
7			

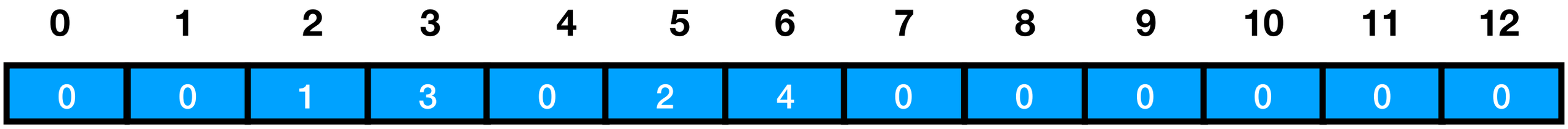
2 % 13 = 2 -> пробуем в ячейку 2 записать номер строки таблицы.
Уже занято
Пробуем записать в ячейку 2 + 1 = 3. Удачно



Приходит пара (песель, 42)
h(x) = 31
Вставляем эту информацию в таблицу

Номер	Hash	Key	Value
0	NULL	NULL	NULL
1	15	пушистик	22
2	5	котейка	48
3	2	собакен	24
4	31	песель	42
5			
6			
7			

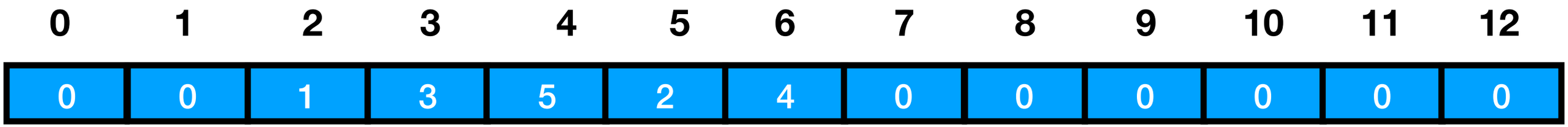
31 % 13 = 5 -> пробуем в ячейку 5 записать номер строки таблицы.
Уже занято
Пробуем записать в ячейку 5 + 1 = 6. Удачно



Приходит пара (кот, 13)
h(x) = 16
Вставляем эту информацию в таблицу

Номер	Hash	Key	Value
0	NULL	NULL	NULL
1	15	пушистик	22
2	5	котейка	48
3	2	собакен	24
4	31	песель	42
5	16	кот	13
6			
7			

16 % 13 = 3 -> пробуем в ячейку 3 записать номер строки таблицы.
Уже занято
Пробуем записать в ячейку 3 + 1 = 4. Удачно



Приходит пара (кошка, 22)
h(x) = 9
Вставляем эту информацию в таблицу

Номер	Hash	Key	Value
0	NULL	NULL	NULL
1	15	пушистик	22
2	5	котейка	48
3	2	собакен	24
4	31	песель	42
5	16	кот	16
6	9	кошка	22
7			

16 % 13 = 9 -> пробуем в ячейку 9 записать номер строки таблицы.
Удачно

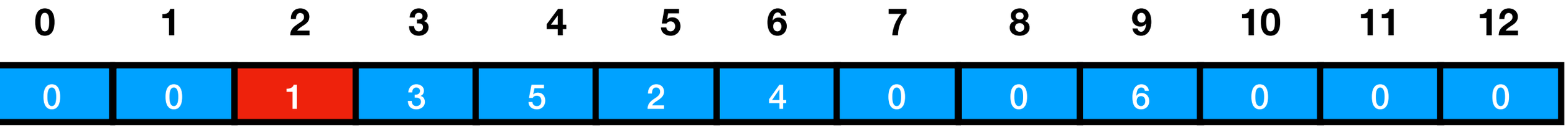
0	1	2	3	4	5	6	7	8	9	10	11	12
0	0	1	3	5	2	4	0	0	6	0	0	0

Как искать?

Ищем пушистика

h(x) = 15

15 % 13 = 2

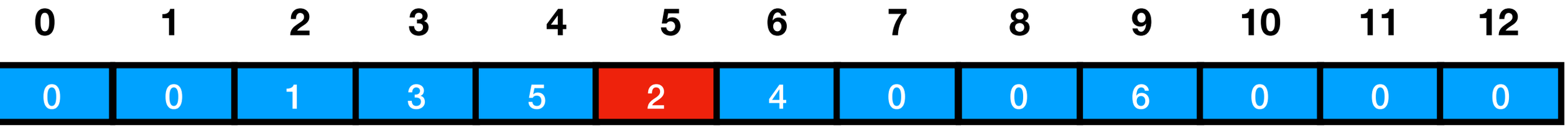


Номер	Hash	Key	Value
0	NULL	NULL	NULL
1	**15**	**пушистик**	**22**
2	5	котейка	48
3	2	собакен	24
4	31	песель	42
5	16	кот	16
6	9	кошка	22
7			

Как искать?

Ищем песеля
h(x) = 31
31 % 13 = 5

Хэш в строке 2 не равен 31, значит, не совпадут и строки



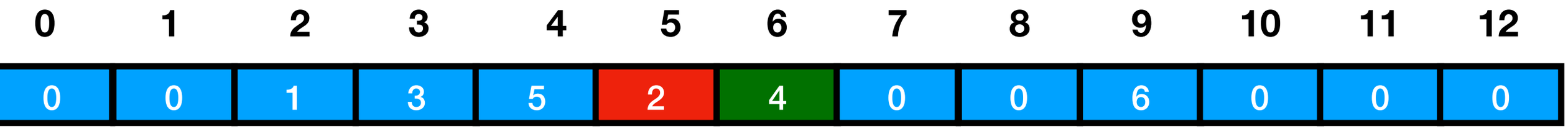
Номер	Hash	Key	Value
0	NULL	NULL	NULL
1	15	пушистик	22
2	5	котейка	48
3	2	собакен	24
4	31	песель	42
5	16	кот	16
6	9	кошка	22
7			

Как искать?

Ищем песеля
h(x) = 31
31 % 13 = 5

Пробуем 6 ячейку массива - она указывает на 4ю строку

Хэши совпадают, сравниваем сами строки - нашли

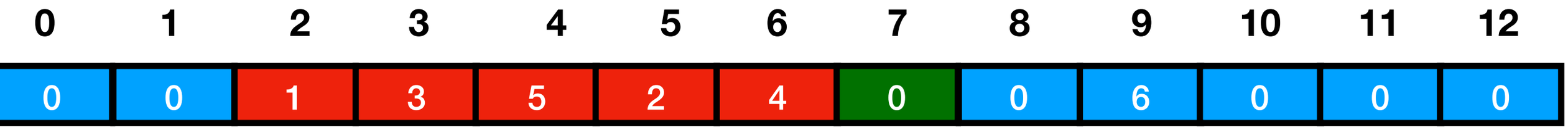


Номер	Hash	Key	Value
0	NULL	NULL	NULL
1	15	пушистик	22
2	5	котейка	48
3	2	собакен	24
4	**31**	**песель**	**42**
5	16	кот	16
6	9	кошка	22
7			

Как искать?

Ищем барабан
h(x) = 28
28 % 13 = 2

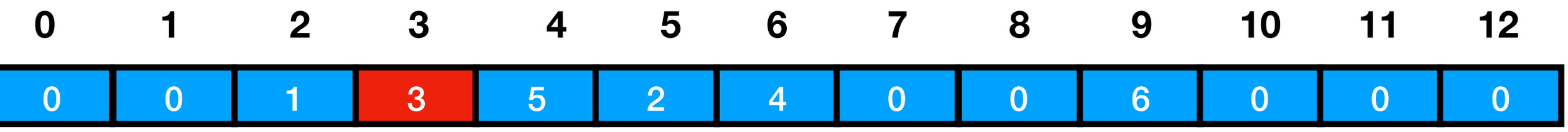
Опробуем все ячейки в массиве с 2 по 7. На 7й наткнулись на NULL - значения в массиве нет



Номер	Hash	Key	Value
0	NULL	NULL	NULL
1	15	пушистик	22
2	5	котейка	48
3	2	собакен	24
4	31	песель	42
5	16	кот	16
6	9	кошка	22
7			

Как удалять?

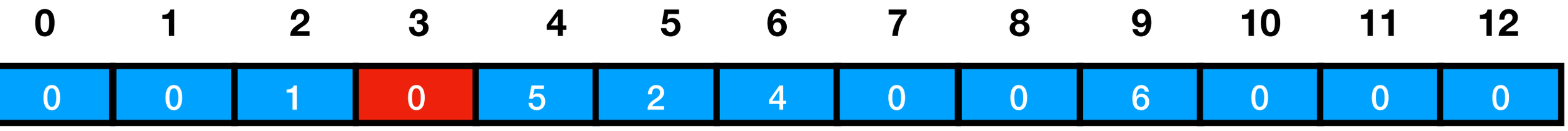
Удаляем собакена. Сначала находим (умеем), потом в ячейке 3 пишем 0?



Номер	Hash	Key	Value
0	NULL	NULL	NULL
1	15	пушистик	22
2	5	котейка	48
3	2	собакен	24
4	31	песель	42
5	16	кот	16
6	9	кошка	22
7			

Как удалять?

Удаляем собакена. Сначала находим (умеем), потом в ячейке 3 пишем 0?

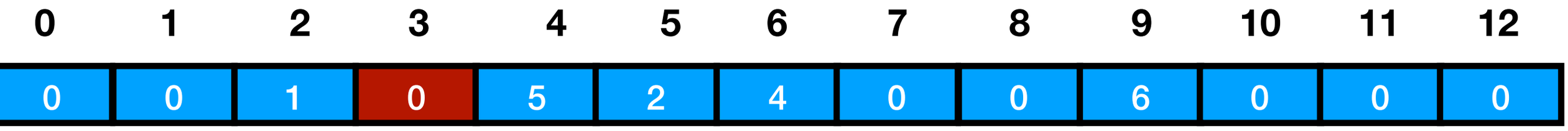


Номер	Hash	Key	Value
0	NULL	NULL	NULL
1	15	пушистик	22
2	5	котейка	48
3	NULL	NULL	NULL
4	31	песель	42
5	16	кот	16
6	9	кошка	22
7			

Как удалять?

Удаляем собакена. Сначала находим (умеем), потом в ячейке 3 пишем 0?

Нельзя! Иначе мы не найдем кота, когда будем искать

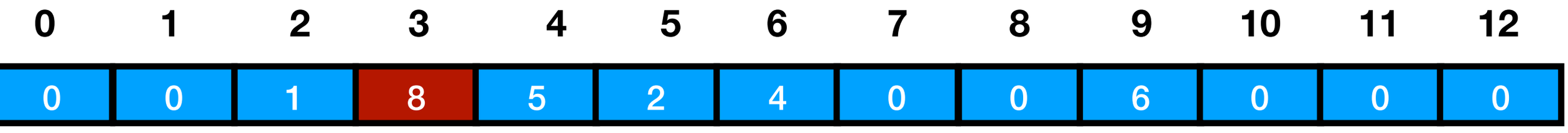


Номер	Hash	Key	Value
0	NULL	NULL	NULL
1	15	пушистик	22
2	5	котейка	48
3	NULL	NULL	NULL
4	31	песель	42
5	16	кот	16
6	9	кошка	22
7			

Как удалять?

Удаляем собакена.

Пишем в ячейке 3, к примеру, размер таблицы с записями (8). Этим мы показываем, что ячейка удалена. Когда-нибудь сможем удалить по-настоящему?

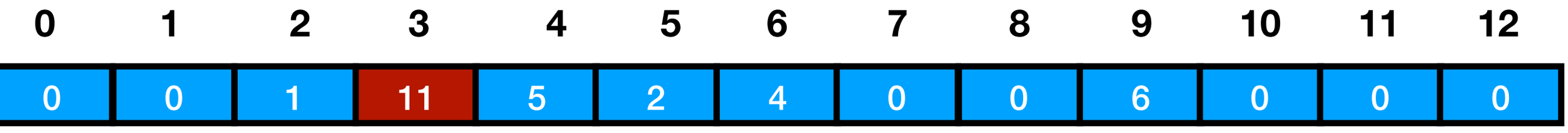


Номер	Hash	Key	Value
0	NULL	NULL	NULL
1	15	пушистик	22
2	5	котейка	48
3	NULL	NULL	NULL
4	31	песель	42
5	16	кот	16
6	9	кошка	22
7			

Как удалять?

Удаляем собакена.

Пишем в ячейке 3, к примеру, 11 = размер таблицы с записями (8) + номер строки (3). Этим мы показываем, что ячейка удалена. Удаляем по-настоящему если операцией добавления наткнулись на эту ячейку. Или когда увеличиваем размер хэш-таблицы



Номер	Hash	Key	Value
0	NULL	NULL	NULL
1	15	пушистик	22
2	5	котейка	48
3	NULL	NULL	NULL
4	31	песель	42
5	16	кот	16
6	9	кошка	22
7			

Время работы вставки

Точно понятно, что зависит от заполненности хэш-таблицы (части с массивом). Пусть размер таблицы - N. Пусть заполнено доля равная alpha. Сколько в среднем будем искать, куда запись новое вставляемое значение?

- вероятность, что случайно взятая ячейка непустая

α^2 **- вероятность, что две ячейки подряд непустые**

...

Число испытаний до первой удачи (p = 1 - alpha) - геометрическое распределение

$$\frac{1}{1-\alpha}$$ **- столько в среднем попробуем ячеек**

Время работы вставки

$$\frac{1}{1-\alpha}$$

- столько в среднем попробуем ячеек

Даже при заполненности таблицы наполовину - всего 2 ячейки. На 90% - 10.

Правда, мы предполагали независимость, но будем считать, что у нас очень хорошая хэш-функция

Аналогично для других операций

$$\frac{1}{1-\alpha} = O(\alpha)$$

Время работы

Операция	Среднее время	Худшее
Вставка	O(1 + alpha)	O(alpha * N) или O(N), если пришлось увеличивать размер таблицы
Поиск	O(1 + alpha)	O(alpha * N)
Удаление	O(1 + alpha)	O(alpha * N)

Преимущества

1. Потребляет меньше памяти, чем метод цепочек
2. Лежит в памяти линейно - хорошо взаимодействует с кэшами
3. Позволяет хранить порядок вставки ключей

Недостатки

1. Плохо работает с удалением