Лекция 11.

Хеширование

Постановка задачи

- ***Хеширование*** - класс методов сортировки и поиска, основанный на арифметическом преобразовании ключа ***K*** и получении некоторой функции, указывающей ***адрес*** в таблице, где хранится запись с ключом ***K***.

Постановка задачи

- Английский глагол “to hash” крошить
- Вычисляем хеш-функцию ключа ***h(K)***
- Берем это значение в качестве адреса в таблице
- Размещаем запись по адресу ***h(K)***
- Различные ключи $K_i \neq K_j$,
- ***h(Ki) = h(Kj) – коллизия***

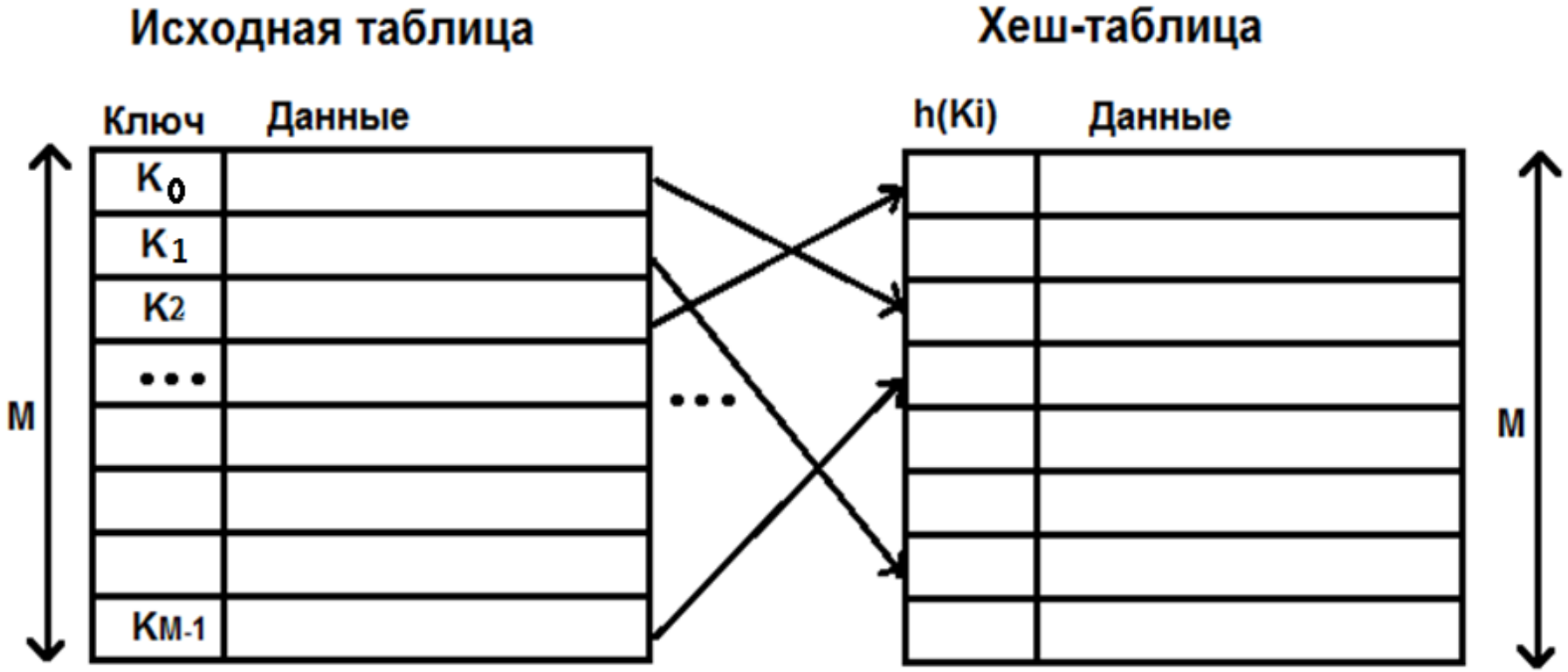
Выбор хеш-функции – непростая задача.

Хеш-функция должна удовлетворять двум требованиям:

- равномерное распределение записей по всей таблице, что минимизирует количество коллизий,
- быстрота вычисления.

Постановка задачи

- Пусть ***M*** - размер области памяти размещения записей, т.е. ***0 ≤ h(K) < M***



Выбор хеш-функции

Функция, преобразующая ключ в некоторый адрес, называется **хеш-функцией. Если *h – некоторая хеш-функция, а K – ключ, то h(K) является значением хеш-функции, или хеш-адресом .***

Любая хорошая функция хеширования должна как можно равномернее распределять ключи по всему диапазону значений адресов. Однако всегда существует вероятность того, что найдутся ключи ***K1 ≠ K2, при которых h(K1) = h(K2). Такая ситуация называется коллизией при хешировании.***

Таким образом, при использовании метода хеширования необходимо решить две задачи: **выбрать функцию хеширования и метод разрешения коллизий.**

Каждый ключ может быть цифровым (номер), алфавитным (ФИО) или алфавитно-цифровым (номер группы). Однако всегда есть возможность преобразовать ключи в целые числа, поэтому будем предполагать, что множество ключей состоит из целых величин.

Преобразования ключа в адрес

1. Ключ преобразуется в цифровое представление
2. Ключ в преобразуется в квазислучайное число с использованием алгоритма перемешивания
3. Окончательное вычисление хеш-функции

Метод деления

Наиболее распространенная функция хеширования основывается на **методе деления и имеет вид h(K) = (K mod m),**
где ***K – ключ, mod –*** операция, вычисляющая остаток от деления***, m –*** делитель***.***

Равномерность распределения получаемых адресов во многом зависит от выбранного делителя ***m.*** Следует избегать четных делителей, так как при этом четные и нечетные ключи отображаются соответственно в четные и нечетные адреса. Если множество ключей состоит в основном из четных или в основном из нечетных ключей, будут возникать многочисленные коллизии.

Если **m** является большим простым числом, то количество коллизий невелико. Таким образом, следует выбирать в качестве делителя **m** простое число, наиболее близкое к **n** – диапазону требуемых адресов.

Метод середины квадрата

При хешировании по методу середины квадрата исходный ключ умножается сам на себя (возводится в квадрат). В качестве адреса выбирается столько цифр из середины результата, сколько требует длина адреса.

Пусть дан ключ **234583**. При возведении его в квадрат получаем 753**9582**3889. Если требуется 100 адресов, то адрес будет равен 58, если необходимо 1000 адресов – 582, если 10000 – 9582.

Иногда нужно получить некратное 10 количество адресов, например 736. В этом случае необходимо взять три средние цифры и умножить на коэффициент 0,736. Например, $582 \cdot 0{,}736 = 428$.

Эксперименты с реальными данными показали, что метод середины квадрата дает неплохой результат при условии, что ключи не содержат много левых или правых нулей подряд.

Выделение разрядов

Ключ разбивается на три части, средняя из которых равна длине адреса.

132|456 ~ 13|24|56

Сдвиг разрядов

Все разряды числа, являющегося ключом, разбиваются на две части: старшие и младшие разряды. Обе эти части сдвигаются по направлению друг к другу. Цифры, содержащиеся в перекрывающихся разрядах, суммируются.

132|456 ~ 13|6|56 ~ 1|68|6

Метод свертывания

В методе свертывания ключ разбивается на части, каждая из которых имеет длину, равную длине требуемого адреса. Адрес формируется как сумма этих частей. При этом перенос в старший разряд игнорируется. Экспериментальным путем было показано, что свертывание справа налево дает меньшее число коллизий, чем слева направо.

Пусть дан ключ 3415768898. Для двух-, трех- и четырехцифрового адреса получим следующие значения:

3415768898

98 + 88 + 76 + 15 + 34 = 11; (результат 311)

898 + 768 + 415 + 3 = 084; (результат 2084)

8898 + 1576 + 34 = 0508. (результат 10508)

Метод свертывания используется, как правило, для больших ключей.

Преобразование, подобное преобразованию основания системы счисления

Пусть ключ равен 172 148.

Выберем в качестве основания число 11

$1*11^5 + 7*11^4 + 2*11^3 + 1*11^2 + 4*11^1 + 8*11^0 = 266373$.

Младшие разряды: 6373.

Метод умножения

1. Умножение на константу **C = (M-1) / (K_{max} - K_{min})**
   $h(K_i) = (K_i - K_{min}) \cdot C$

2 . Пусть количество адресов равно **m.** Зафиксируем константу A в интервале 0 < A < 1 и положим, что

$$h(K) = \lfloor m \cdot D(K \cdot A) \rfloor,$$

где D(K · A) – дробная часть произведения K · A.

Достоинство метода в том, что качество хеш-функции мало зависит от выбора ***m.*** Метод умножения работает при любом выборе константы A, но некоторые ее значения могут быть лучше других. В литературе предлагается следующее значение константы A:

$$A \approx (\sqrt{5} - 1)/2 \approx 0{,}6180339887...$$

Оценка хеш-функции

1. Ее вычисление должно быть быстрым;
2. Она должна минимизировать число коллизий.

$$P = N / M$$

P – плотность заполнения

N – число записей, находящихся в основной области

M – размер основной области

Методы разрешения коллизий

Метод открытой адресации

Пусть задан ключ **K** и массив **S**, состоящий из **n** элементов. Необходимо разместить ключ **K** в массиве **S**.

d = h(K) – получаемый при использовании некоторой хеш-функции **h** индекс массива **S**, причем **0 ≤ d ≤ n – 1**.

Если элемент массива S_d свободен, то ключ **K** помещается в эту позицию и включение элемента завершается.

Если же элемент S_d уже занят, в массиве просматриваются другие элементы до тех пор, пока не будет найдено свободное место для размещения ключа **K.** Простейший способ поиска свободной позиции состоит в последовательном просмотре элементов массива с индексами **d, d + 1, ..., n – 1, 0, 1, 2, ..., d – 1.**

Если в массиве есть хотя бы один свободный элемент, он будет найден. Иначе, после просмотра всех позиций массива можно сделать заключение о том, что массив переполнен и добавление нового ключа невозможно.

Алгоритм поиска ключа основан на вычислении его хеш-адреса и просмотре той же последовательности элементов. Либо, при удачном поиске, будет найден искомый ключ, либо, при поиске неудачном, будет найден пустой элемент или будут просмотрены все элементы массива. Рассмотрим применение метода открытой адресации на примере.

Пусть задан массив **S** из 11 элементов, хеш-функция **h(K) = (K mod 11),** последовательность ключей **– 88, 21, 96, 86, 11, 22, 5, 29, 19.**
Коллизия возникает при занесении ключей 11, 22 и 19.

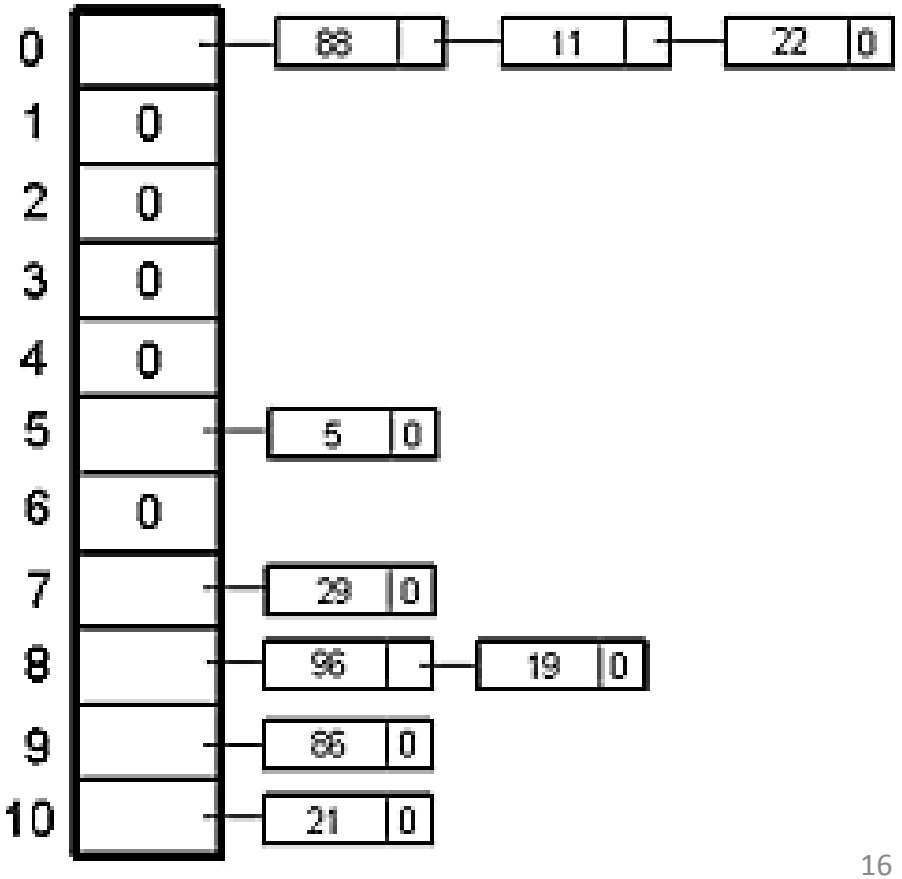
0	1	2	3	4	5	6	7	8	9	10
88	11	22	19		5		29	96	86	21

Если из массива удаляется ключ $\mathbf{K_i}$, то в соответствующий элемент массива необходимо занести специальный признак удаленного ключа (например, отрицательное число, если все ключи положительные, или наибольшее положительное число и т.п.), т.е. значение, которое не равно ни одному возможному значению ключа. Записи, помеченные как удаленные, при поиске рассматриваются как занятые, а при занесении – как свободные.

Метод цепочек

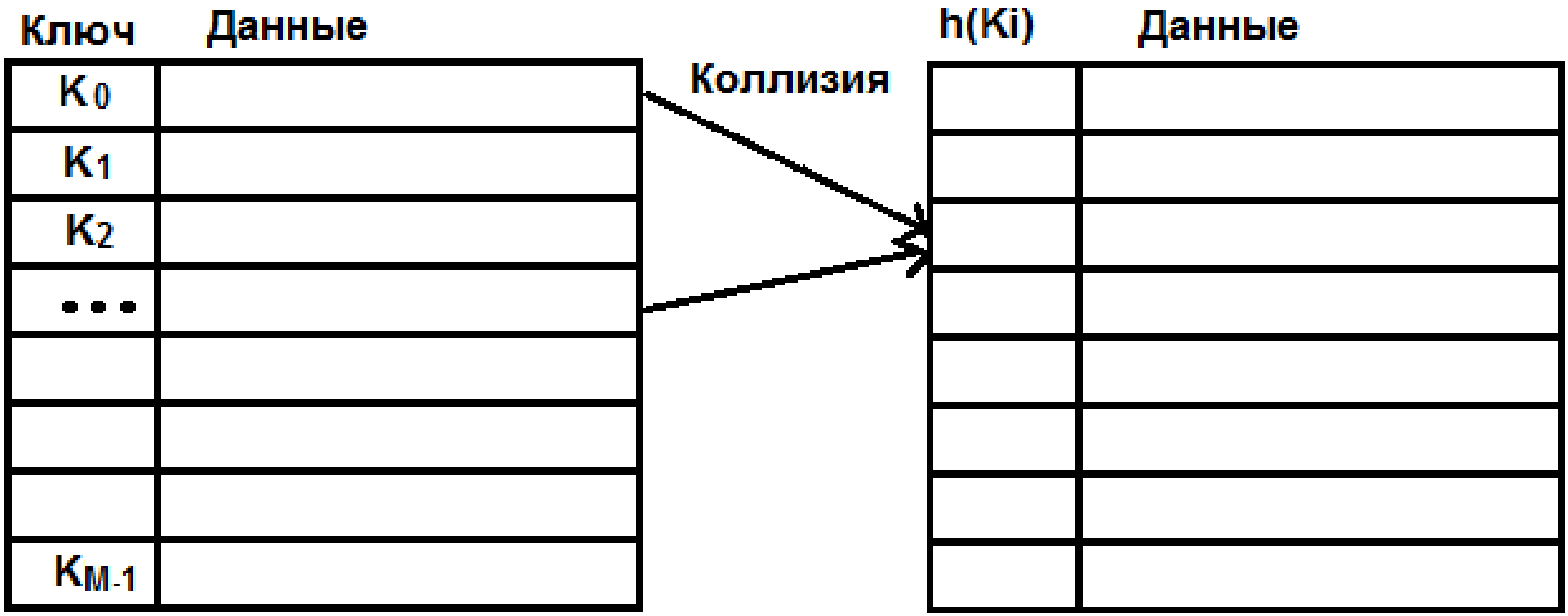
Другой способ разрешения коллизий состоит в том, чтобы поддерживать ***M*** связанных списков, по одному на каждый возможный хеш-адрес. Таким образом, каждый список будет содержать все ключи с одинаковым хеш-адресом. Кроме того, необходимо иметь массив из M указателей на начало каждого из M списков. Метод цепочек лишен большинства недостатков, присущих методу открытой адресации.

Удаление элемента состоит в исключении узла из связного списка, что никак не влияет на эффективность как алгоритма поиска, так и алгоритма включения. Основным недостатком метода цепочек является то, что для указателей требуется дополнительная память.



Ключ	Симво-льный ключ	Коды символов	Сумма	Окончатель-ное значение хеш-функции K mod N
Clark	ar	97 114	211	1
Pratt	at	97 116	213	3
Kroll	ol	111 108	219	9
Sanchez	nc	110 99	209	9
Lopes	pe	112 101	213	3
Smith	it	105 116	221	1
Jonson	ns	110 115	225	5
Rasin	si	115 105	220	0
Guhu	hu	104 117	221	1
Petech	te	116 101	217	7

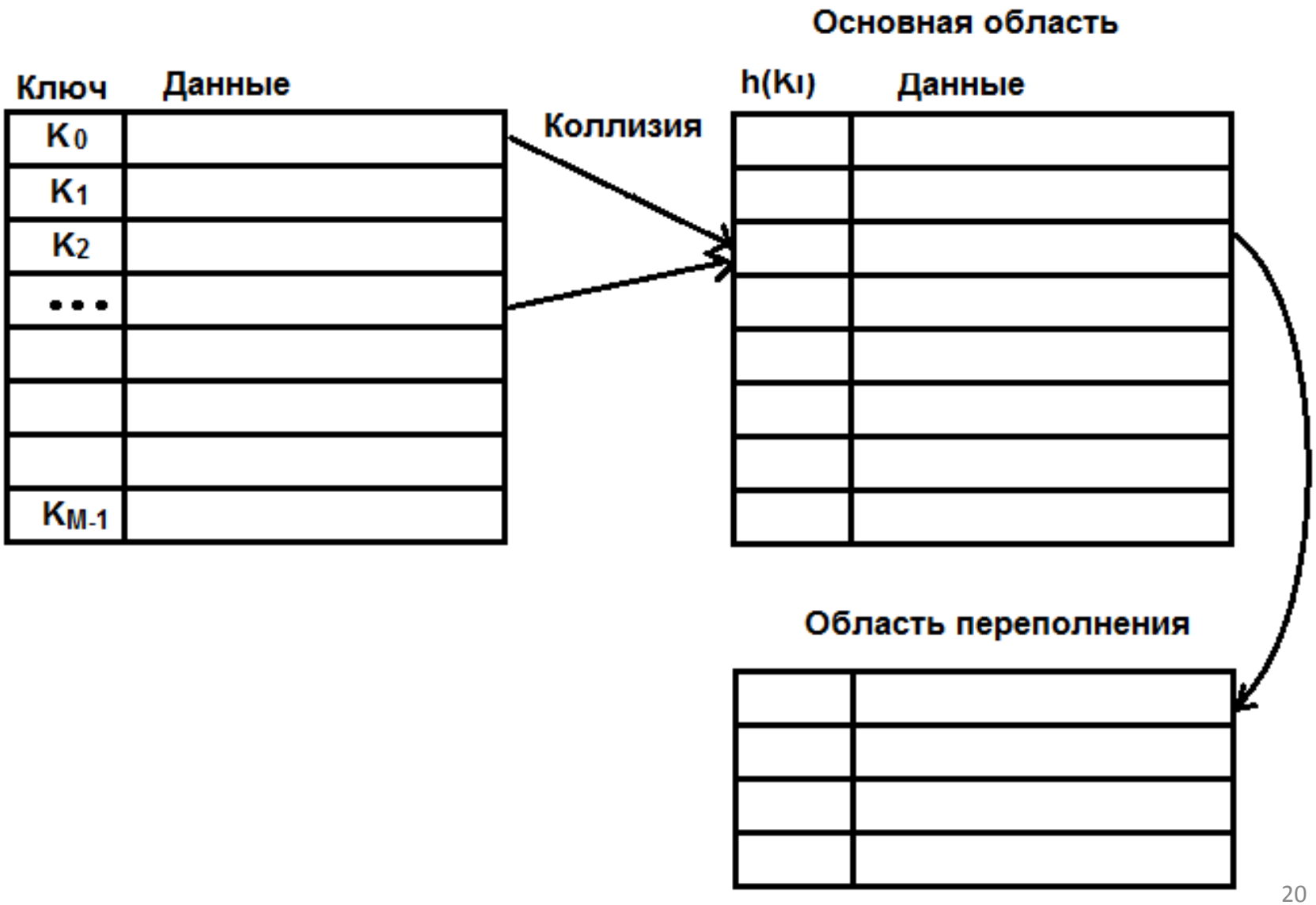
Разрешение коллизий



Два подхода к разрешению коллизий

1. Записи располагаются в отдельной области, называемой областью переполнения
2. Записи располагаются в пределах основной области

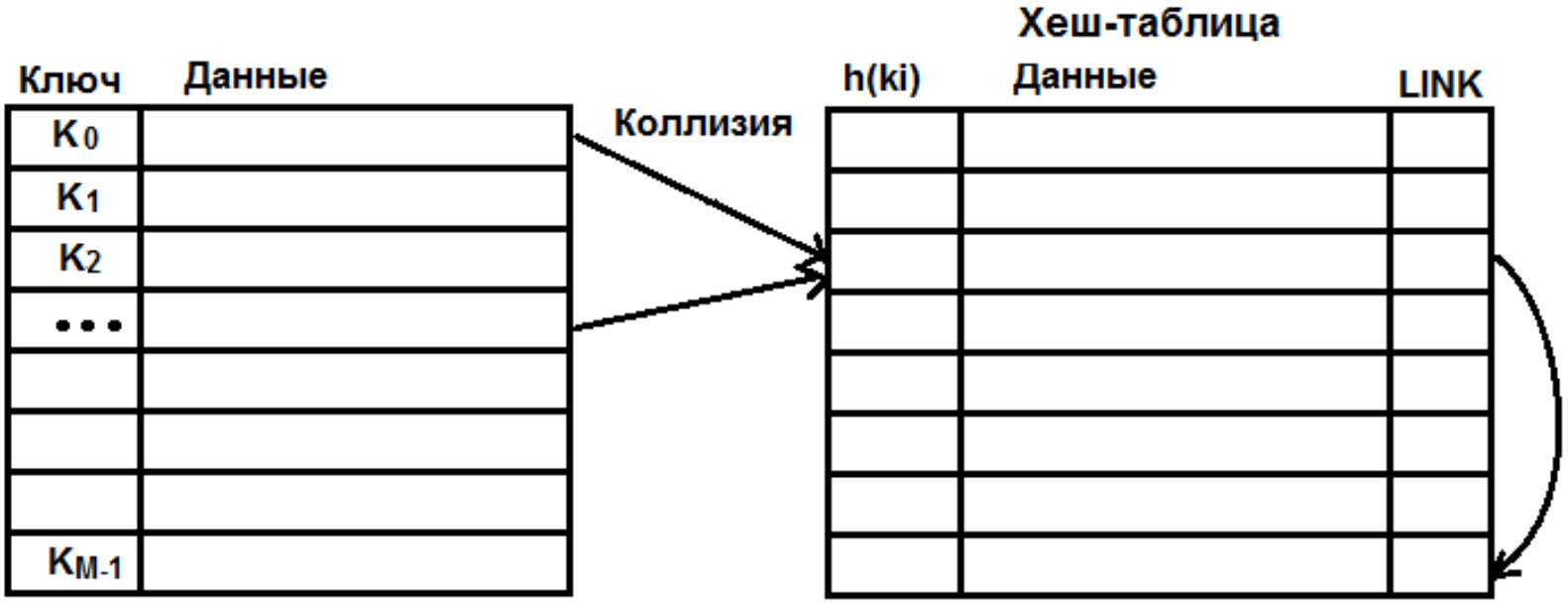
Использование области переполнения



Поиск в хеш-таблице

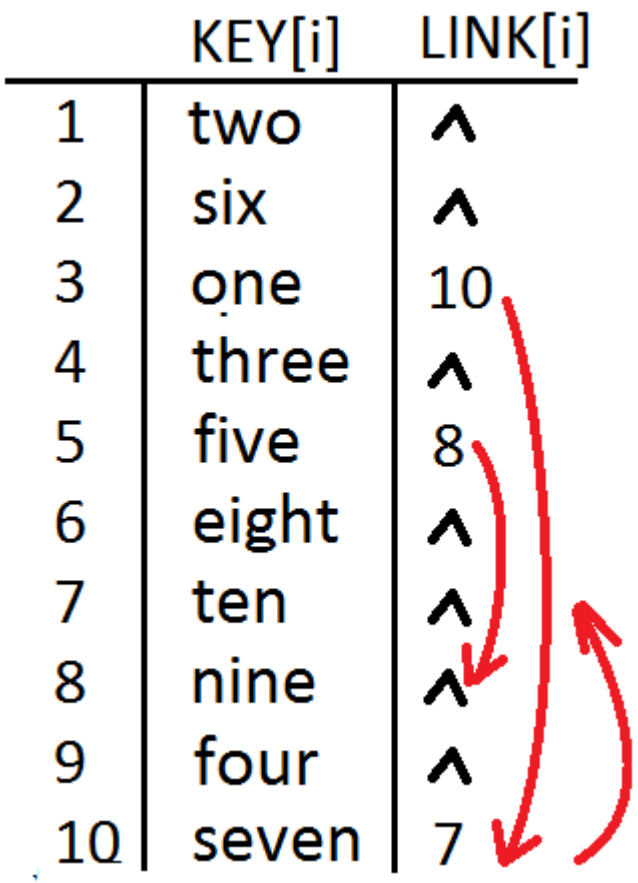
1. Ввести искомый ключ
2. Преобразовать ключ в адрес с помощью хеш-функции. Адрес – это индекс в массиве
3. Поиск в основной области по хеш-функции. Если ключ найден то УДАЧА
4. Если ключ не найден то последовательный поиск в области переполнения
5. Если ключ найден то УДАЧА иначе НЕУДАЧА

Метод цепочек



Метод цепочек. Пример

K:	one	two	three	four	five	six	seven	eight	nine	ten
h(K):	3	1	4	9	5	2	3	6	5	10



	KEY[i]	LINK[i]
1	two	^
2	six	^
3	one	10
4	three	^
5	five	8
6	eight	^
7	ten	^
8	nine	^
9	four	^
10	seven	7

Метод цепочек. Поиск

1. Ввести искомый ключ
2. Преобразовать ключ в адрес с помощью хеш-функции.
3. Идем по этому адресу.
4. Если ключ найден то УДАЧА иначе делаем шаг по связи
5. Если не конец цепочки то на шаг 3 иначе НЕУДАЧА

Открытая рассеянная адресация

- Просмотр начинается от текущей позиции ключа ***К снизу вверх***.
- Если найден свободный участок, ключ вставляется на этот участок.
- Если участок не найден, поиск продолжается с конца таблицы ***вверх***, пока не будет найден свободный участок.

Открытая рассеянная адресация