# Хэш-таблицы. Устройство, сложность операций. Методы разрешения коллизий. Коэффициент загрузки таблицы (load-factor), частота коллизий.

Хэш-таблица – специальная структура данных для хранения пар ключей и их значений. По сути, это ассоциативный массив (массив связных списков), в котором ключ представлен в виде хэш-функции.

Хэш-функция – функция, преобразовывающая входную последовательность данных произвольного размера в выходную последовательность фиксированного размера. Процесс преобразования данных называется хешированием. Результат хеширования – хеш-код(хеш-сумма, хеш).

**Общая информация**

Существуют два основных варианта хеш-таблиц: с цепочками и открытой адресацией. Хеш-таблица содержит некоторый массив H{\displaystyle H}, элементы которого есть пары (хеш-таблица с открытой адресацией) или списки пар (хеш-таблица с цепочками).

Выполнение операции в хеш-таблице начинается с вычисления хеш-функции от ключа. Получающееся хеш-значение {\displaystyle i={\mbox{hash}}(key)}i=hash(key) играет роль индекса в массиве {\displaystyle H}H. Затем выполняемая операция (добавление, удаление или поиск) перенаправляется объекту, который хранится в соответствующей ячейке массива {\displaystyle H[i]}H[i].

Ситуация, когда для различных ключей получается одно и то же хеш-значение, называется коллизией. Такие события не так уж и редки — например, при вставке в хеш-таблицу размером 365 ячеек всего лишь 23 элемента вероятность коллизии уже превысит 50 % (если каждый элемент может равновероятно попасть в любую ячейку). Поэтому механизм разрешения коллизий — важная составляющая любой хеш-таблицы.

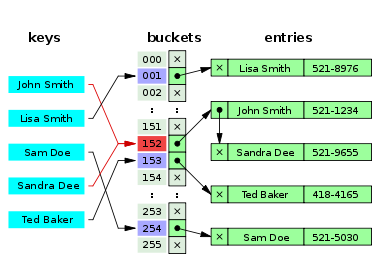
В некоторых специальных случаях удаётся избежать коллизий вообще. Например, если все ключи элементов известны заранее (или очень редко меняются), то для них можно найти некоторую совершенную хеш-функцию, которая распределит их по ячейкам хеш-таблицы без коллизий. Хеш-таблицы, использующие подобные хеш-функции, не нуждаются в механизме разрешения коллизий, и называются хеш-таблицами с *прямой адресацией*.

Число хранимых элементов, делённое на размер массива H{\displaystyle H} (число возможных значений хеш-функции), называется **коэффициентом заполнения хеш-таблицы** (load factor) и является важным параметром, от которого зависит среднее время выполнения операций.(LoadFactor = n / m (где n – кол-во ключей, m – кол-во ячеек)

Важное свойство хеш-таблиц состоит в том, что, при некоторых разумных допущениях, все три операции (поиск, вставка, удаление элементов) в среднем выполняются за время {\displaystyle O(1)}O(1). Но при этом не гарантируется, что время выполнения отдельной операции мало́. Это связано с тем, что при достижении некоторого значения коэффициента заполнения необходимо осуществлять перестройку индекса хеш-таблицы: увеличить значение размера массива {\displaystyle H}H и заново добавить в пустую хеш-таблицу все пары.

**Способы разрешения коллизий:**

**Метод цепочек**

[](https://commons.wikimedia.org/wiki/File:Hash_table_5_0_1_1_1_1_1_LL.svg?uselang=ru)

Каждая ячейка массива *H* является указателем на связный (цепочку) пар ключ-значение, соответствующих одному и тому же хеш-значению ключа. Коллизии просто приводят к тому, что появляются цепочки длиной более одного элемента.

Операции поиска или удаления элемента требуют просмотра всех элементов соответствующей ему цепочки, чтобы найти в ней элемент с заданным ключом. Для добавления элемента нужно добавить элемент в конец или начало соответствующего списка и в случае, если коэффициент заполнения станет слишком велик, увеличить размер массива *H* и перестроить таблицу.

При предположении, что каждый элемент может попасть в любую позицию таблицы *H* с равной вероятностью и независимо от того, куда попал любой другой элемент, среднее время работы операции поиска элемента составляет О(1 + *loadFactor*).

**Открытая адресация**

[](https://commons.wikimedia.org/wiki/File:Hash_table_5_0_1_1_1_1_0_SP.svg?uselang=ru)

Пример хеш-таблицы с открытой адресацией и линейным пробированием, получающейся при вставке элементов в левой колонке сверху вниз.

В массиве *H* хранятся сами пары ключ-значение. Алгоритм вставки элемента проверяет ячейки массива *H* в некотором порядке до тех пор, пока не будет найдена первая свободная ячейка, в которую и будет записан новый элемент. Этот порядок вычисляется на лету, что позволяет сэкономить на памяти для указателей, требующихся в хеш-таблицах с цепочками.

Последовательность, в которой просматриваются ячейки хеш-таблицы, называется последовательностью проб. В общем случае она зависит только от ключа элемента, то есть это последовательность *h*0(*x*), *h*1(*x*), …, *hn*— 1(*x*), где *x* — ключ элемента, а *hi*(*x*) — произвольные функции, сопоставляющие каждому ключу ячейку в хеш-таблице. Первый элемент в последовательности, как правило, равен значению некоторой хеш-функции от ключа, а остальные считаются от него одним из приведённых ниже способов. Для успешной работы алгоритмов поиска последовательность проб должна быть такой, чтобы все ячейки хеш-таблицы оказались просмотренными ровно по одному разу.

Алгоритм поиска просматривает ячейки хеш-таблицы в том же самом порядке, что и при вставке, до тех пор, пока не найдется либо элемент с искомым ключом, либо пока не будет достигнут конец списка. Ошибочно представление, что следует искать до первой свободной ячейки, так как возможно, что элемент из этой ячейки был удален, а искомый элемент находится далее.

Удаление элементов в такой схеме несколько затруднено. Обычно поступают так: заводят булевый флаг для каждой ячейки, помечающий, удален элемент в ней или нет. Тогда удаление элемента состоит в установке этого флага для соответствующей ячейки хеш-таблицы, но при этом необходимо модифицировать процедуру поиска существующего элемента так, чтобы она считала удалённые ячейки занятыми, а процедуру добавления — чтобы она их считала свободными и сбрасывала значение флага при добавлении.

**Последовательности проб**

Ниже приведены некоторые распространенные типы последовательностей проб. Сразу оговорим, что нумерация элементов последовательности проб и ячеек хеш-таблицы ведётся от нуля, а *N* — размер хеш-таблицы (и, как замечено выше, также и длина последовательности проб).

* **Линейное пробирование**: ячейки хеш-таблицы последовательно просматриваются с некоторым фиксированным интервалом *k* между ячейками (обычно *k* = 1), то есть *i*-й элемент последовательности проб — это ячейка с номером (hash(*x*) + *ik*) mod *N*. Для того, чтобы все ячейки оказались просмотренными по одному разу, необходимо, чтобы *k* было взаимно-простым с размером хеш-таблицы.
* **Квадратичное пробирование**: интервал между ячейками с каждым шагом увеличивается на константу. Если размер хеш-таблицы равен степени двойки (*N* = 2*p*), то одним из примеров последовательности, при которой каждый элемент будет просмотрен по одному разу, является: hash(*x*) mod *N*, (hash(*x*) + 1\*1) mod *N*, (hash(*x*) + 2\*2) mod *N*, (hash(*x*) + 3\*3) mod *N*, …
* **Двойное хеширование**: интервал между ячейками фиксирован, как при линейном пробировании, но, в отличие от него, размер интервала вычисляется второй, вспомогательной хеш-функцией, а значит, может быть различным для разных ключей. Значения этой хеш-функции должны быть ненулевыми и взаимно-простыми с размером хеш-таблицы, что проще всего достичь, взяв [простое число](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D1%81%D1%82%D0%BE%D0%B5_%D1%87%D0%B8%D1%81%D0%BB%D0%BE) в качестве размера, и потребовав, чтобы вспомогательная хеш-функция принимала значения от 1 до *N* — 1.