Вот неплохой материал с картинками:

https://ruhighload.com/post/%D0%A7%D1%82%D0%BE+%D1%82%D0%B0%D0%BA%D0%BE%D0%B5+%D1%85%D0%B5%D1%88-%D1%82%D0%B0%D0%B1%D0%BB%D0%B8%D1%86%D1%8B+%D0%B8+%D0%BA%D0%B0%D0%BA+%D0%BE%D0%BD%D0%B8+%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%B0%D1%8E%D1%82

**Простое представление хеш-таблиц**

Чтобы разобраться, что такое хеш-таблицы, представьте, что вас попросили создать библиотеку и заполнить ее книгами. Но вы не хотите заполнять шкафы в произвольном порядке.

Первое, что приходит в голову — разместить все книги в алфавитном порядке и записать все в некий справочник. В этом случае не придется искать нужную книгу по всей библиотеке, а только по справочнику.

А можно сделать еще удобнее. Если изначально отталкиваться от названия книги или имени автора, то лучше использовать некий алгоритм хеширования, который обрабатывает входящее значение и выдает номер шкафа и полки для нужной книги.

Зная этот алгоритм хэширования, вы быстро найдете нужную книгу по ее названию.

Учтите, что хеш-функция должна иметь следующие свойства:

* Всегда возвращать один и тот же адрес для одного и того же ключа;
* Не обязательно возвращает разные адреса для разных ключей;
* Использует все адресное пространство с одинаковой вероятностью;
* Быстро вычислять адрес.

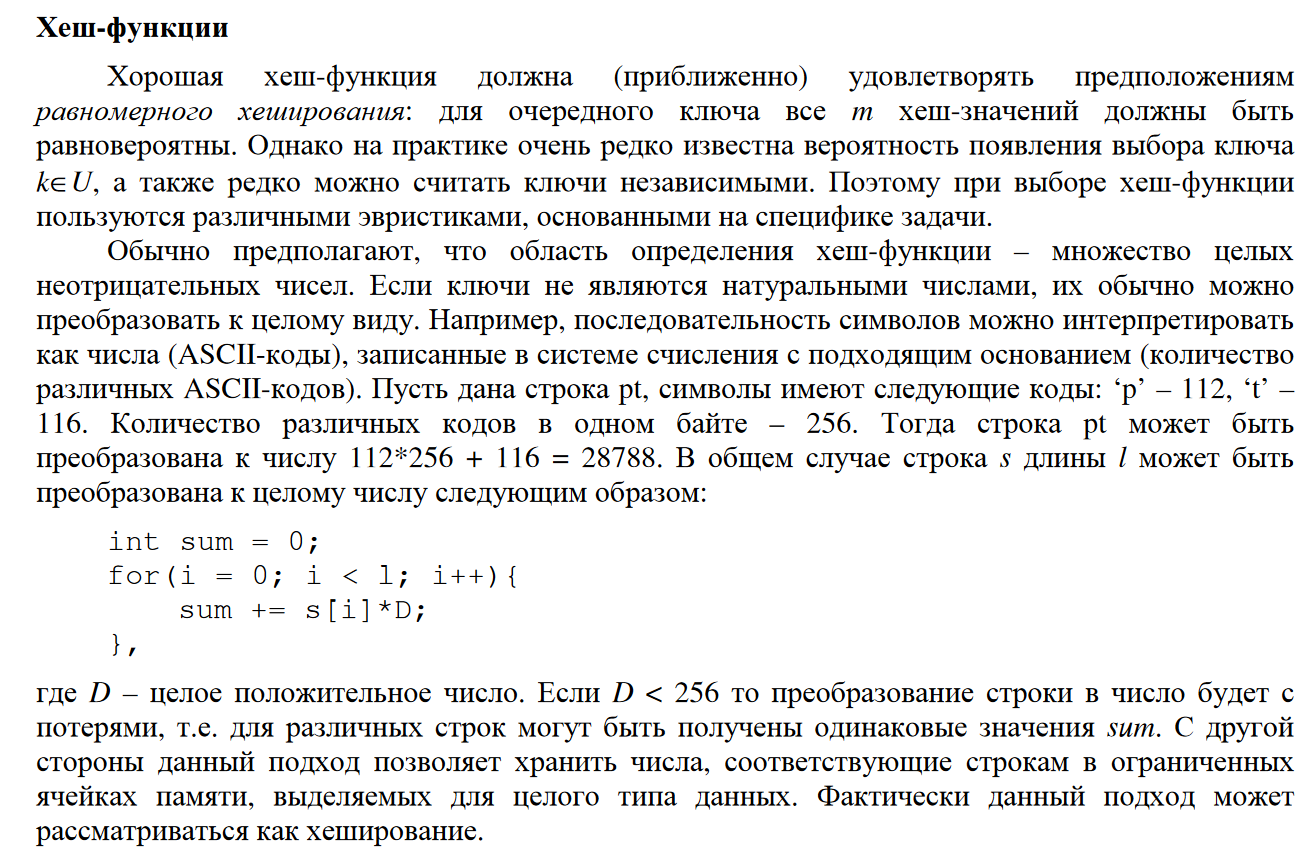
## Борьба с коллизиями (они же столкновения)

В идеальном случае, когда заранее известны все пары ключ-значение, достаточно легко реализовать идеальную хеш-таблицу, в которой время поиска будет постоянным (используется идеальная хеш-функция, которая определяет положения в таблице по целым значениям и без столкновений).

Но в большинстве случаев приходится бороться с коллизиями. Обычно применяются методы цепочек и открытой индексации.

### Метод цепочек

Этот метод часто называют **открытым хешированием**. Его суть проста — элементы с одинаковым хешем попадают в одну ячейку в виде [связного списка](https://ru.wikipedia.org/wiki/%D0%A1%D0%B2%D1%8F%D0%B7%D0%BD%D1%8B%D0%B9_%D1%81%D0%BF%D0%B8%D1%81%D0%BE%D0%BA).



Пожалуй, главное свойство hash-таблиц — все три операции вставка, поиск и удаление в среднем выполняются за время O(1), среднее время поиска по ней также равно O(1) и O(n) в худшем случае.

Разобрать задачу: есть текст. Нужно посчитать, сколько каких букв в тексте (используя принцип хэш-таблиц) (см папку "прога 1 - счет букв")

Задача: есть аппроксимация нормального распределения (распределения Гаусса):

function gaussianRand() {

var rand = 0;

for (var i = 0; i < 6; i += 1) {

rand += Math.random();

}

return rand / 6;

}

(от нуля до единицы).

Построить гистограмму (по сути - можно использовать обычные массивы, главное, пусть суть подсчета поймут, подобные подсчеты часто встречаются на практике.) (см папку "прога 2 - гистограмма")