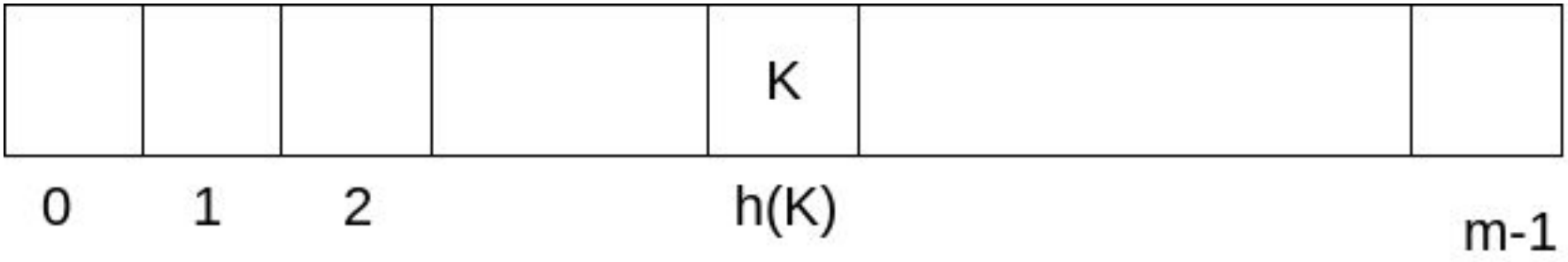
Хэш-таблицы

Берленко Т.А. СПбГЭТУ “ЛЭТИ”, ФКТИ, МОЭВМ

Хэш-таблицы

- **Хэш-таблица** - структура данных, которая позволяет хранить пары (ключ, значение) и осуществлять доступ к элементу по ключу.



Хэш-функция

Хэш-функция

$$H: K \rightarrow \{0, \ldots, m-1\}$$

- K - множество ключей

(например, множество номеров телефонов)

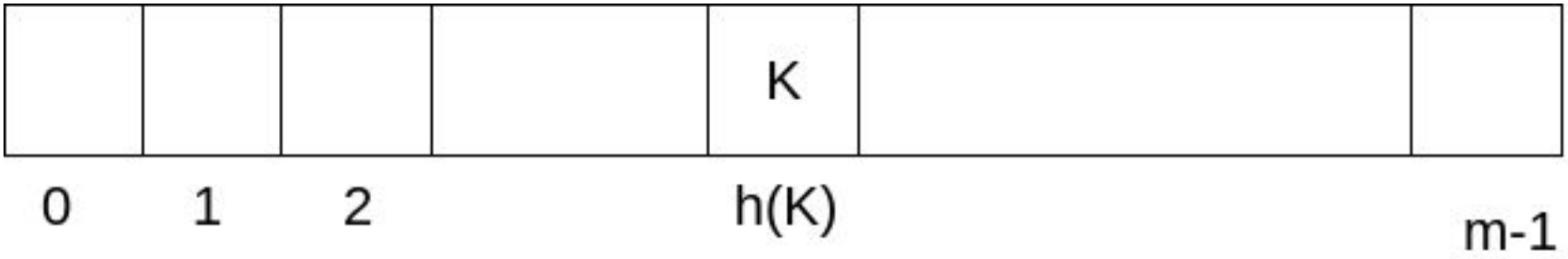
- H(K) - хэш-код ключа K
- m - параметр хэш-функции, количество ячеек хэш-таблицы

Хэш-таблица

α = n / m - коэффициент заполнения хэш-таблицы.

n - количество ключей

m - количество ячеек в таблице



Коллизии

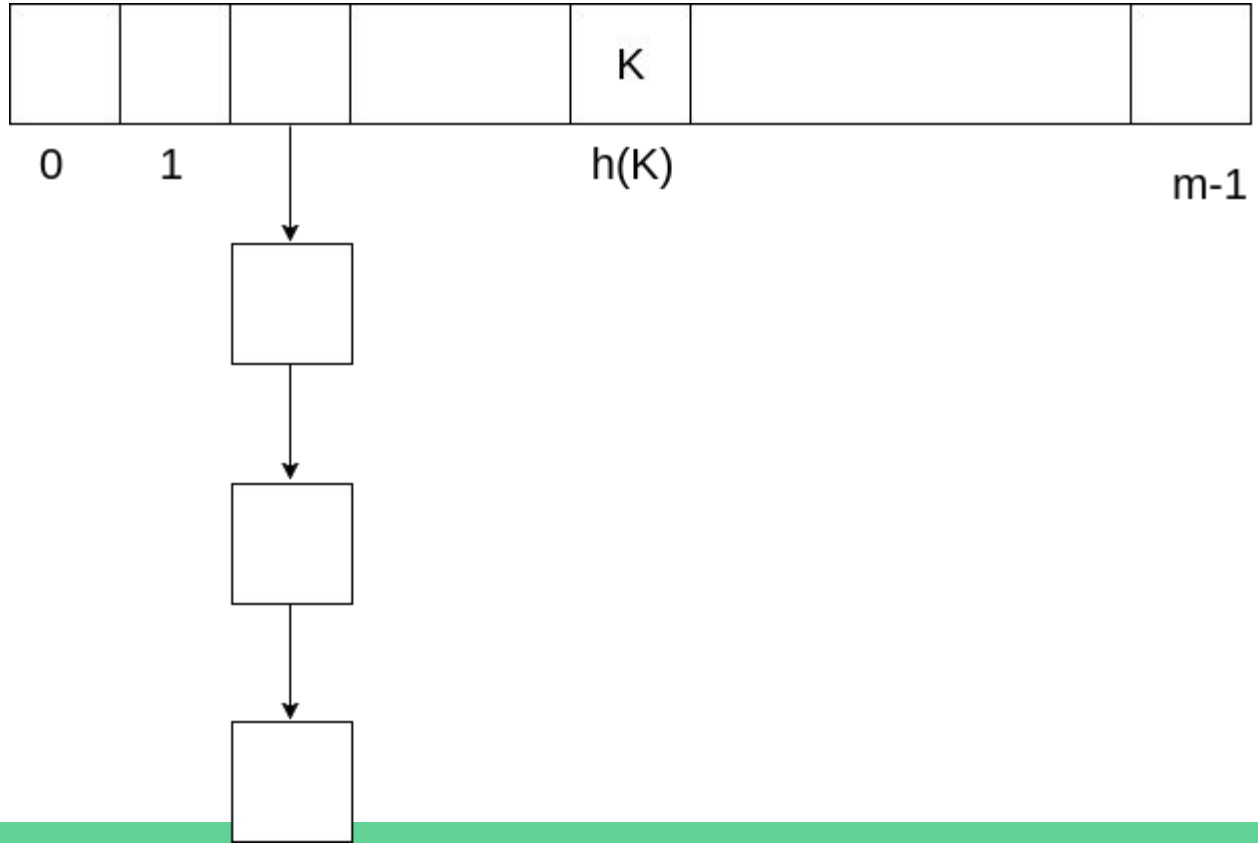
Коллизия - совпадение хэш-значений при разных ключах:

$H(K_1) = H(K_2)$, при $K_1 \neq K_2$

Частые способы разрешения коллизий:

- Цепочки
- Открытая адресация

Методы разрешения коллизий. Цепочки



Методы разрешения коллизий. Открытая адресация

- Работает только в случае $n \leq m$
- При вставке ключа выполняется проверка, свободна ли требуемая ячейка. Если ячейка свободна, выполняется вставка, иначе происходит поиск альтернативного места для вставки ключа.
- Самый простой подход - *последовательное исследование* - последовательный перебор всех ячеек таблицы.

Особенности реализации словарей в Python3.5 и <

- Словари реализованы как хэш-таблицы с открытой адресацией для разрешения коллизий.
- Ключи в словаре должны быть **хэшируемы**.
- При создании словаря создается хэш-таблица с 8 ячейками.
- Словарь увеличивается в два раза при заполнении более чем на ⅔.

python

Хэш-таблицы

	В худшем случае	В среднем
Поиск элемента	O(n)	O(1)
Вставка элемента	O(n)	O(1)
Удаление элемента	O(n)	O(1)

Полезные ссылки

- Хэш-таблицы

  https://ru.wikipedia.org/wiki/Хеш-таблица

- Лекции по алгоритмам и структурам данных

  https://proglib.io/p/data-structure-algorithms/

- Современные словари в Python:

  https://www.youtube.com/watch?v=37S53yFg9wc