# Алгоритмы и структуры данных-2

# Хеширование-1.
# Свойства и анализ хеш-функций

Практическое занятие 01 – **19.01.2024**

2023-2024 учебный год

# План

Вычисление хеша целочисленных ключей. Проблемы распределения получаемых значений

Вычисление хеша по строковым данным

Хеширование с солью – хранение паролей и пара слов о криптографической хеш-функции

# Refresher

# Хеширование **Object → Key → Index**

Двухэтапный процесс вычисления целого индекса для некоторого объекта

# Хеш-функция $h(\text{key})$

Что делает хеш-функцию **хорошей**?

# Вычисляем хеш целочисленного ключа

# Хеширование по остатку

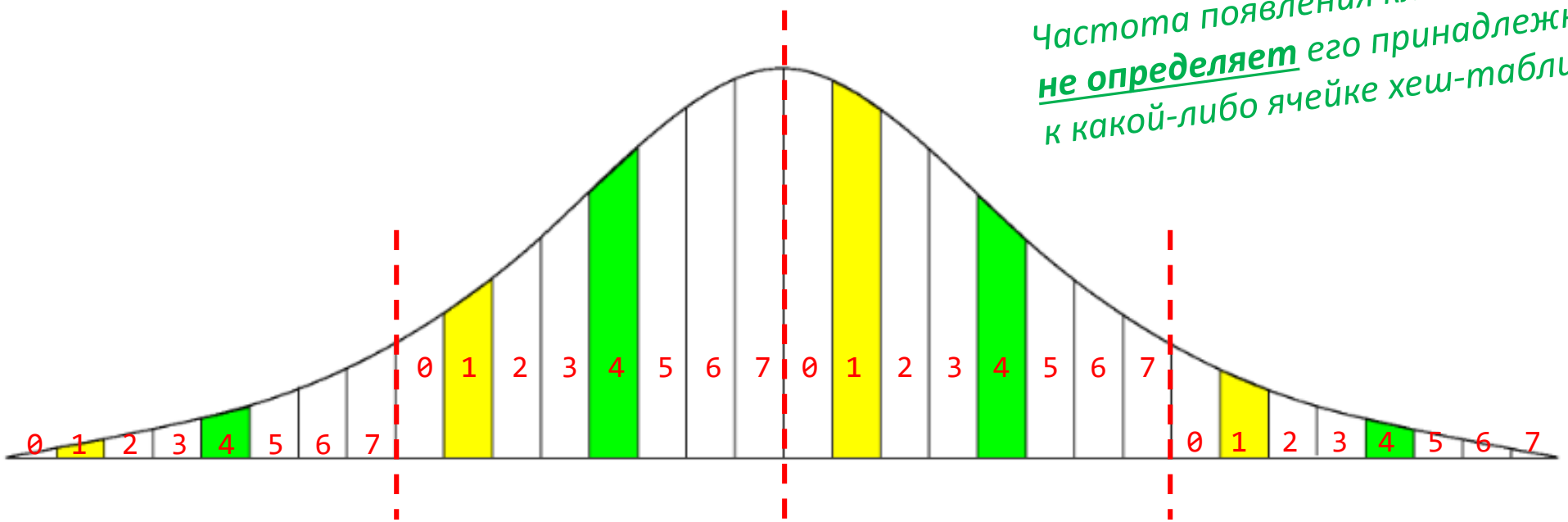
$$h(\text{key}) = \text{key} \bmod M$$

$M$ – размер «хеш-таблицы»

1. Оцените, как влияет значение M на результат вычисления хеша. *Четные/нечетные/простые/...*
2. Какие значения M следует выбирать? Почему?

# Хеширование по остатку

Пусть $M = 8$, а также известно, что ключевые значения имеют нормальное распределение.




*Частота появления ключевого значения **не определяет** его принадлежность к какой-либо ячейке хеш-таблицы*

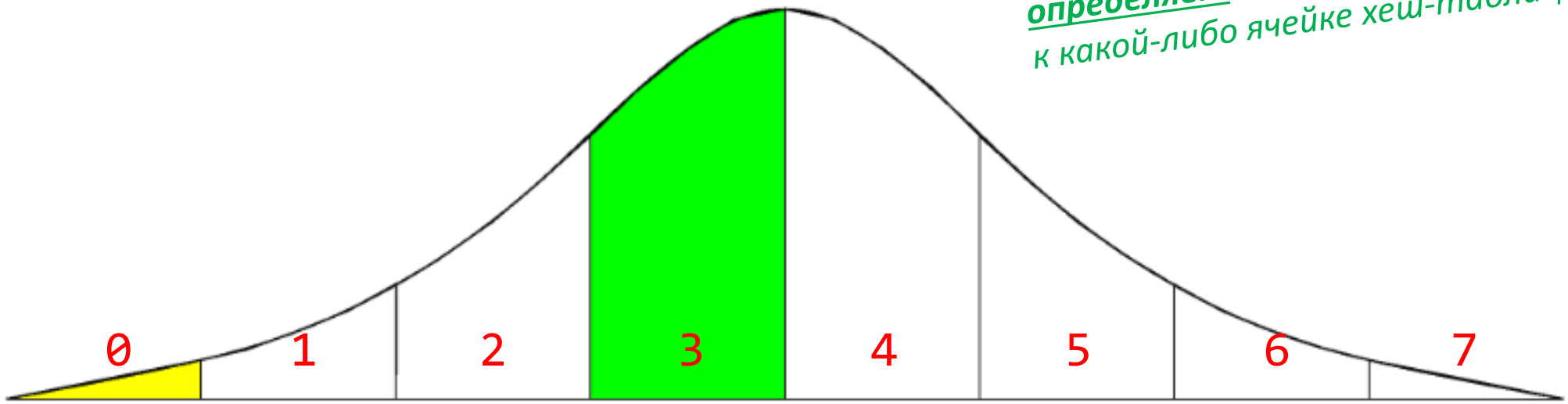
# Binning-хеширование

$$h(\text{key}) = \text{key } \mathbf{div}\ X$$

$X$ – это параметр, который определяет, каким образом выбрать первые цифры (биты), чтобы получить индекс $[0 \ldots M - 1]$ в хеш-таблице размера $M$.

# Binning-хеширование

Пусть размер таблицы $M = 8$, а также известно, что ключевые значения имеют нормальное распределение.




*Частота появления ключевого значения **определяет** его принадлежность к какой-либо ячейке хеш-таблицы*

# Середина квадрата

$$h(\text{key}) = \textbf{midDigits}(key^2)$$

Один из способов генерации *псевдослучайных* чисел, предложенный Дж. Фон Нейманом.

1. Возвести ключевое значение в квадрат.
2. Выбрать необходимое количество разрядов (битов) из середины результата

# Середина квадрата

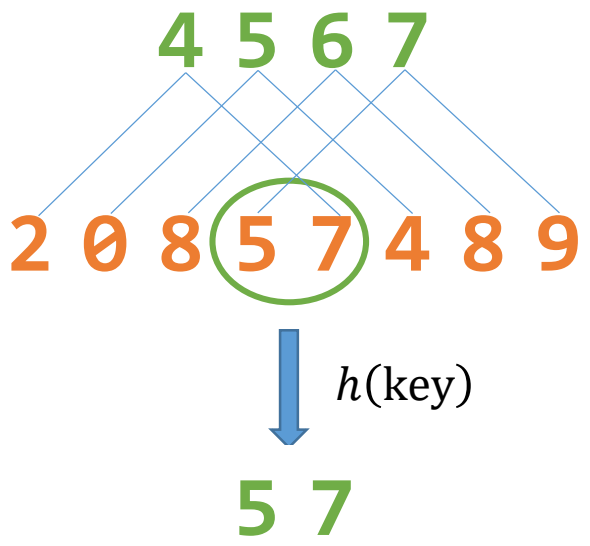
Пусть размер таблицы $M = 100$. Необходимо записать данные об объектах, которые идентифицируются 4-значными ключами.

# Середина квадрата

Пусть размер таблицы $M = 100$. Необходимо записать данные об объектах, которые идентифицируются 4-значными ключами.

$key = 4567$:

$$
\begin{array}{rrrrrrrr}
 & & & & 4 & 5 & 6 & 7 \\
 & & & \times & 4 & 5 & 6 & 7 \\
\hline
 & & & 3 & 1 & 9 & 6 & 9 \\
 & & 2 & 7 & 4 & 0 & 2 & \\
 & 2 & 2 & 8 & 3 & 5 & & \\
1 & 8 & 2 & 6 & 8 & & & \\
\hline
2 & 0 & 8 & 5 & 7 & 4 & 8 & 9
\end{array}
$$



# Вычисляем хеш строковых данных

# Простой folding

$$h(\text{str}) = \sum_{c \in str} c \bmod M$$

1. Вычисление хеша строки как суммы кодов символов происходит медленно - $\Theta(n)$.
2. Порядок символов в исходной строке никак не влияет на результат вычислений...

# Простой folding – Вопрос

Как можно охарактеризовать работу такой функции для хеширования строк длиной в $\mathbf{10}$ заглавных латинских букв 'A'..'Z'?

Предположим, что размер таблицы $\boldsymbol{M = 1000}$.

# Простой folding – Вопрос

Как можно охарактеризовать работу такой функции для хеширования строк длиной в $\mathbf{10}$ заглавных латинских букв `'A'..'Z'`?

Предположим, что размер таблицы $\boldsymbol{M = 1000}$.

Код символа `'A'` – `65`, символа `'Z'` – `90`. Строки будут занимать ячейки хеш-таблицы исключительно в диапазоне `[650..900]`.

# Folding по 4 символа

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| **A** | **A** | **A** | **A** | **B** | **B** | **B** | **B** |

$$65 + 65 \cdot 256 + 65 \cdot 256^2 + 65 \cdot 256^3$$

$$1094795585$$

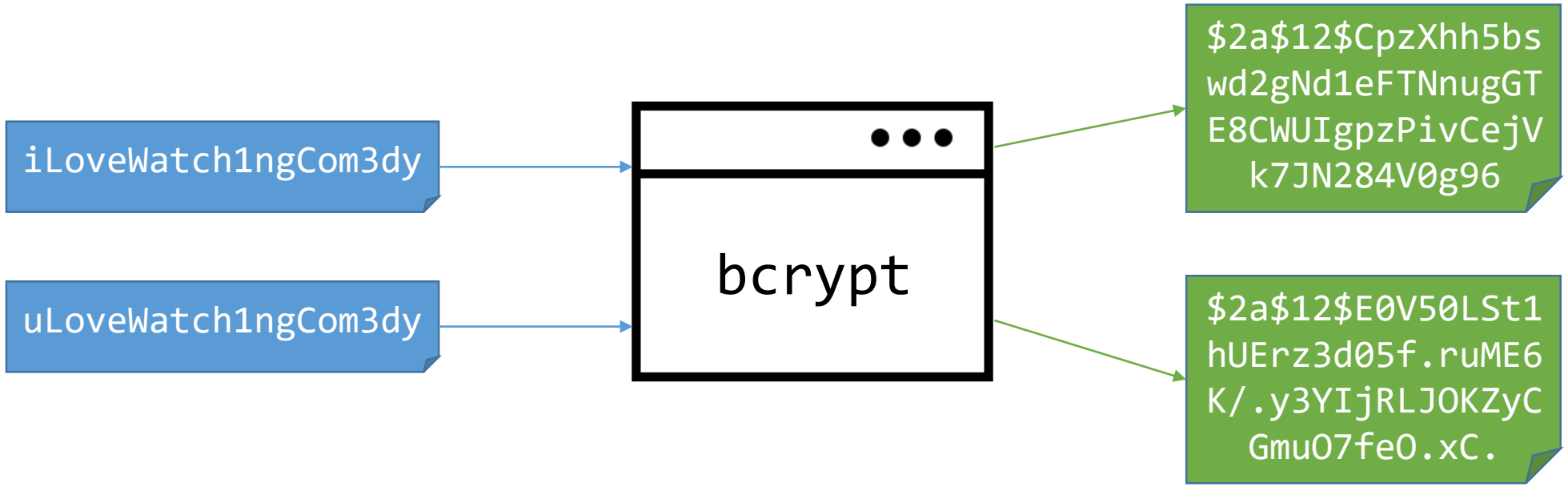
$$66 + 66 \cdot 256 + 66 \cdot 256^2 + 66 \cdot 256^3$$

$$1111638594$$

$$h(\text{AAAABBBB}) = (1094795585 + 1111638594) \bmod 101 = 97$$

# Хеширование с солью

# Хеширование паролей для их хранения




*Криптографическая хеш-функция* дает существенное изменение результата при небольшом изменении входа

# Хеширование паролей для их хранения

Можно выделить два основных способа *взлома* пароля по значению хеш-функции в случае получения несанкционированного доступа к хешам:

1. **Грубая сила** – полный перебор случайных паролей до получения совпадения по значению.
2. **Радужные таблицы** – большой набор предварительно вычисленных хешей

# Хеширование паролей для их хранения

**Соль** – это дополнительная случая строка некоторой длины, которая дописывается к паролю перед тем, как вычислить его хеш

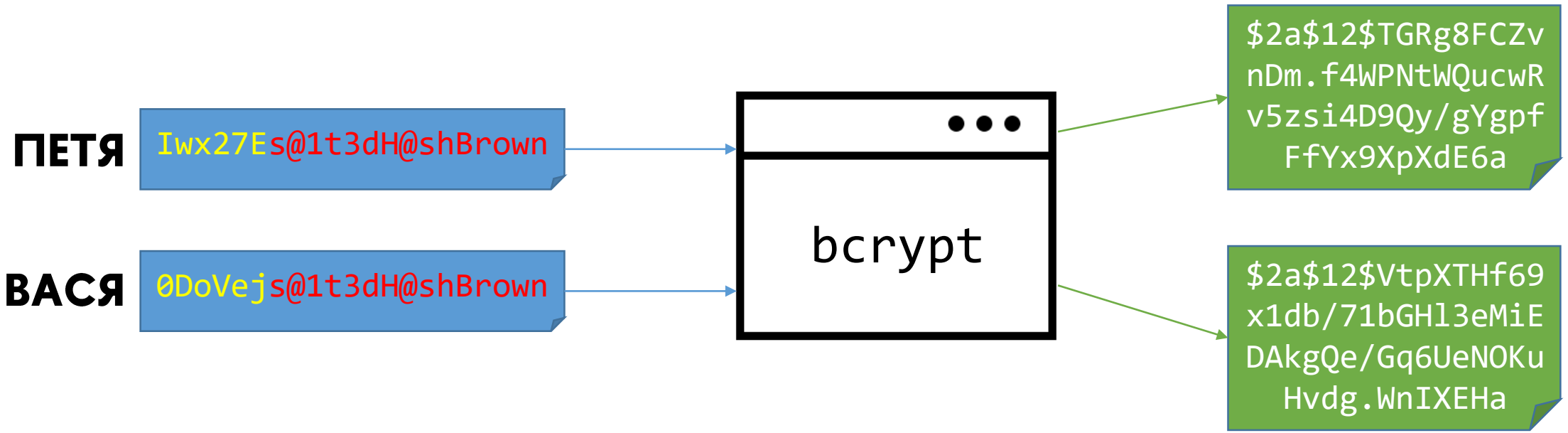
# Хеширование с солью

Предположим, что у **Пети** и **Васи** совпали пароли – `s@1t3dH@shBrown`

Хеш у их паролей также будет одинаковый – `$2a$12$xdWgQ5mhv8rSaUK3qdusTO4XdMFbQi6TD/1VvOZjvGm10RXnhZZa2`

# Хеширование с солью

Добавим разную соль перед тем, как хешировать



# Некоторые известные хеш-функции

*Когда деревья были большими...*

| | |
|---|---|
| Message Digest (MDx) – **MD5**, ... | Долгое время использовались для хранения паролей, однако на данный момент это небезопасно |
| Secure Hash Algorithms (SHA) – **SHA-1**, **SHA-2**, ... | |

# Некоторые известные хеш-функции

| | |
|---|---|
| **Argon2d**, **Argon2i**, **Argon2id** | Семейство алгоритмов, параметризуемых по памяти, длине ключа, длине соли и др. |
| **bcrypt** | Блочное blowfish-шифрование, параметризуемое по количеству итераций |
| **PBKDF1**, **PBKDF2** | Псевдослучайная функция, адаптируемая по количеству итераций и другим параметрам |

https://cheatsheetseries.owasp.org/cheatsheets/Password_Storage_Cheat_Sheet.html