Лекция 5

Хеш-функции и хеш-таблицы

Таблицы с прямой адресацией

Ассоциативный массив — абстрактный тип данных (интерфейс к хранилищу данных), позволяющий хранить пары вида «(ключ, значение)» и поддерживающий операции добавления пары, а также поиска и удаления пары по ключу:

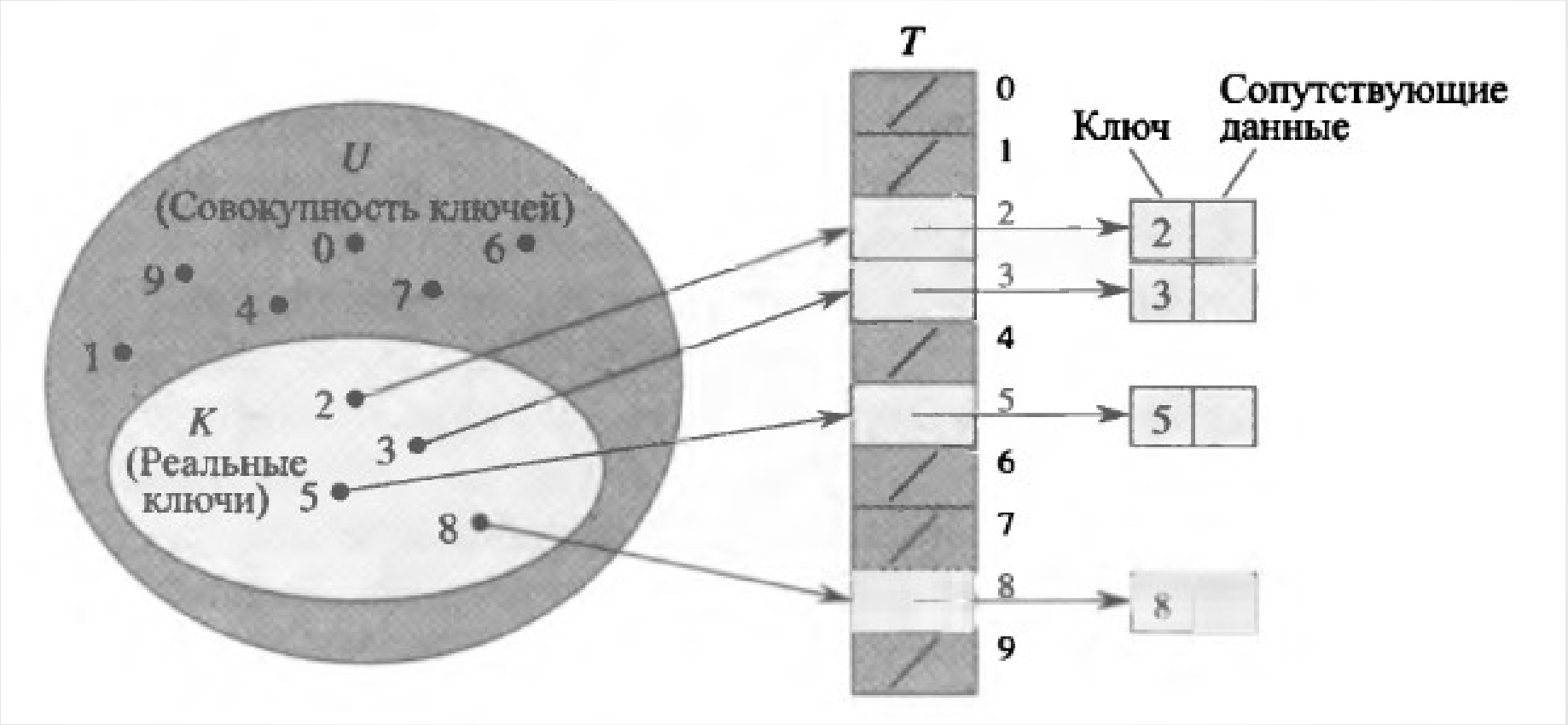
- INSERT(ключ, значение)
- FIND(ключ)
- REMOVE(ключ)

Таблицы с прямой адресацией

Допустим, приложению требуется динамическое множество, каждый элемент которого имеет ключ из совокупности U={0, 1, …, m-1}, где m не слишком велико.

- Никакие два элемента не имеют одинаковых ключей
- Для представления такого множества можно использовать массив T[0..m-1], адреса (индексы) которого соответствуют ключам. Такой массив назвается **таблицей с прямой адресацией**.
- Если множество не содержит элемента с ключом k, то T[k]=NULL

Таблицы с прямой адресацией



Реализация динамического множества с использованием таблицы с прямой адресацией T

Таблицы с прямой адресацией

```
DIRECT-ADDRESS-SEARCH(T, k)
1  return T[k]

DIRECT-ADDRESS-INSERT(T, x)
1  T[x.key] = x

DIRECT-ADDRESS-DELETE(T, x)
1  T[x.key] = NIL
```

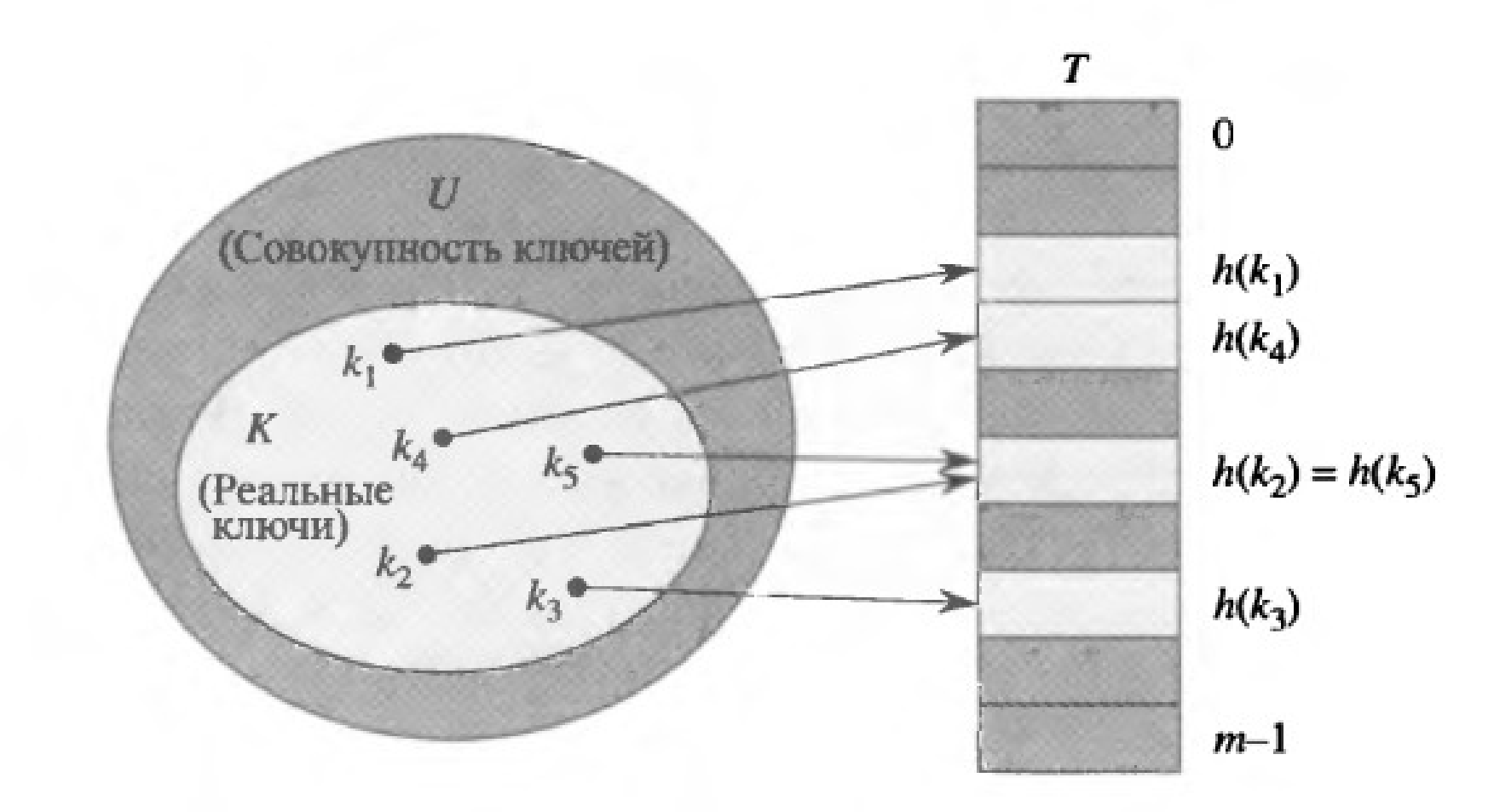
Хеш-таблицы

- Пусть K — множество реально сохранённых ключей
- |K|<<|U|
- В этом случае требования к памяти могут быть снижены до $\Theta(|K|)$
- При этом время поиска элемента в хеш-таблице остается равным O(1) для среднего случая

Введем хеш-функцию h(k):

$$h : U \to \{0, 1, \ldots, m-1\}$$

Хеш-таблицы

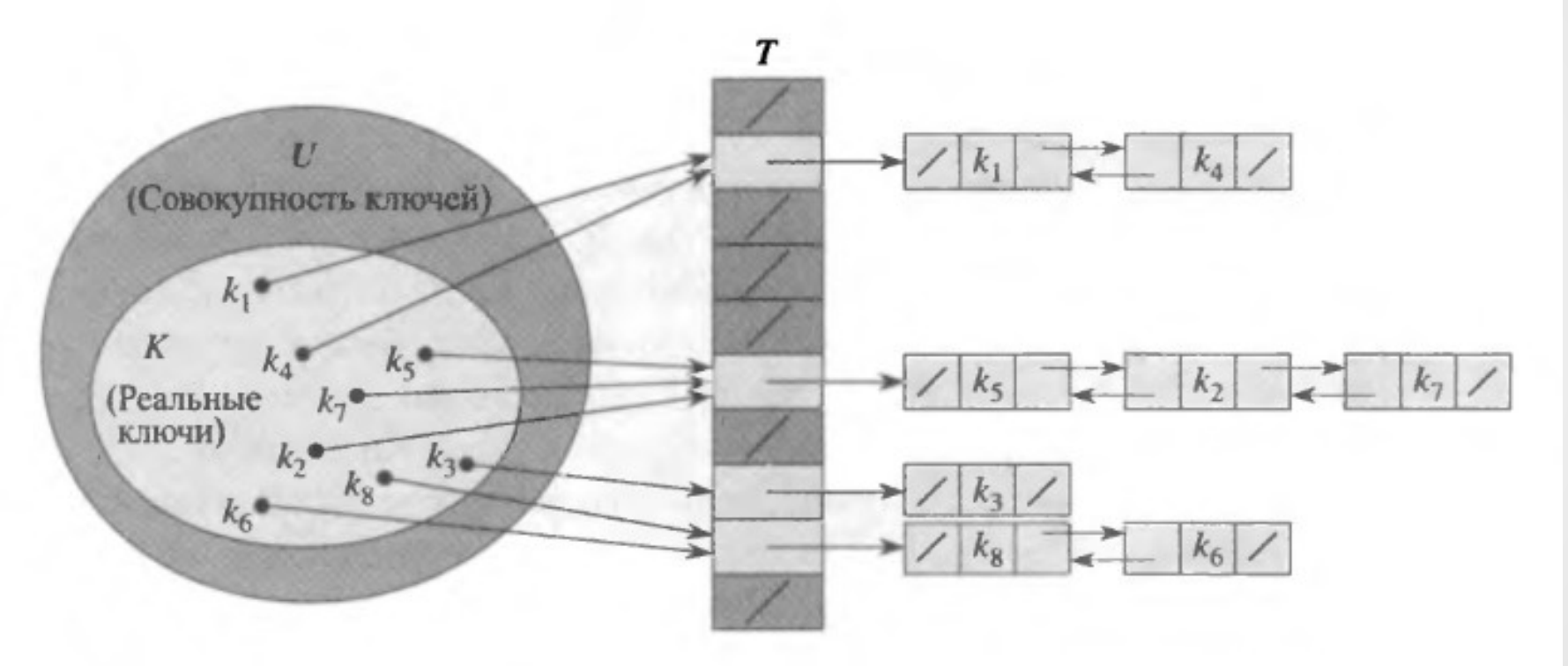


Применение хеш-функции h для отображения ключей в ячейки хеш-таблицы

Хеш-таблицы

- Есть проблема: два ключа могут быть хешированы в одну и ту же ячейку. Такая ситуация называется **коллизией**.
- Можно ли избежать коллизий?
- Вообще говоря, нет, поскольку мы пытаемся сопоставить каждому элементу из множества с большей мощностью элементы из множества с меньшей мощностью.
- Но если коллизии случаются редко, то скорость поиска **в среднем** всё равно будет выше, чем без использования хеш-функций.
- В хороших хеш-функциях коллизии случаются редко.

Хеш-таблицы. Разрешение коллизий с помощью цепочек

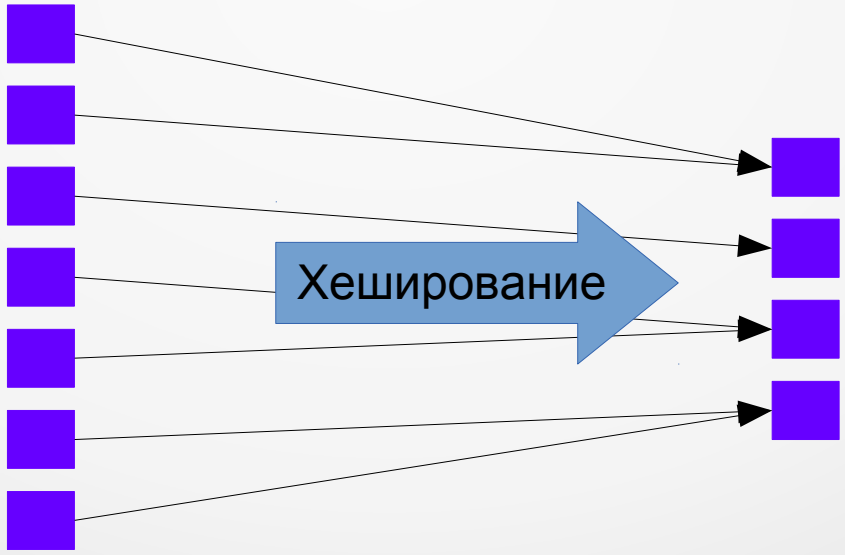


Хеширование

- Хеширование — преобразование массива входных данных произвольной длины в битовую строку фиксированной длины, выполняемое определённым алгоритмом.
- Функция, реализующая алгоритм и выполняющая преобразование, называется «хеш-функцией» или «функцией свёртки».
- Исходные данные называются входным массивом, «ключом» или «сообщением».
- Результат преобразования (выходные данные) называется «хешем», «хеш-кодом» или «хеш-суммой».

Коллизии

Возвращаемые хеш-функцией значения (выходные данные) менее разнообразны, чем значения входного массива (входные данные). Случай, при котором хеш-функция преобразует несколько разных ключей в одинаковые хеш-коды называется «**коллизией**».



Области применения

- ассоциативные массивы (хеш-табицы)
- поиск дубликатов в сериях наборов данных
- построение уникальных идентификаторов для наборов данных
- вычисление контрольных сумм при передаче данных для последующего обнаружения в них ошибок
- хранение паролей в системах защиты в виде хеш-кода
- электронные подписи

«Хеш-функции», основанные на делении

- Хеш-функция может вычислять «хеш» как остаток от деления входных данных на M:

$$h(k) = k \bmod M$$

Почему степень двойки плохо подходит в качестве M?

Обычно выбирают простое M, достаточно удалённое от степеней двойки

«Хеш-функции», основанные на умножении

$$h(k)=\lfloor M\cdot(A\cdot k \bmod 1)\rfloor$$

$A\cdot k \bmod 1$ - получение дробной части произведения A*k

$0<A<1$

В качестве M обычно выбирается 2^w, где w — размер одного машинного слова

Хеширование строк переменной длины

Поэлементное хеширование больших наборов данных (строк)

1. $$h(K) = (h_1(x_1) + h_2(x_2) + \ldots + h_l(x_l)) \mod M$$

2.

```
6  uint8_t pirson(string str)
7  {
8      uint8_t h = 0;
9      for(char& c : str) {
10         uint8_t index = h^c;
11         h := T[index];
12     }
13     return h;
14 }
```

Универсальное хеширование

- Универсальным хешированием называется хеширование, при котором используется не одна конкретная хеш-функция, а происходит выбор хеш-функции из заданного семейства по случайному алгоритму.

Криптографические хеш-функции

Применение:

- Электронная подпись
- Хранение паролей на сервере

Свойства:

- Необратимость
- Стойкость к коллизиям первого рода или восстановлению вторых прообразов: для заданного сообщения M должно быть вычислительно невозможно подобрать другое сообщение N, для которого H(N)=H(M)
- Стойкость к коллизиям второго рода: должно быть вычислительно невозможно подобрать пару сообщений имеющих одинаковый хеш.