NATIONAL RESEARCH UNIVERSITY

АЛГОРИТМЫ И СТРУКТУРЫ ДАННЫХ

Хеширование (часть 1 из 2)

Рамон Антонио Родригес Залепинос
arodriges@hse.ru

Структура модуля 1

	№	Дата	Тема лекции	№	Домашние задания
Модуль № 1	1	08 сен	Введение	A1	Установить Visual Studio с поддержкой C# и C++ на все рабочие машины (стационарные, переносные, т.п.). Найти книги по структурам данных. Посетить конференции.
	2	15 сен	Асимптотика	A2	Тестовая задача (C#): знакомство с процессом
	3	22 сен	Базовые СД	1a	Задание 1a на C#
	4	29 сен	Базовые СД 2	1b	Задание 1b на C++, **перевод 1a на C++**
	5	06 окт	Bitmaps	2a	Задание 2a: Контрольное Домашнее Задание (на C#) – CW
	6	13 окт	Хэш таблицы		

Мы в точности следуем
нашему календарному (понедельному) плану

Требование к студентам на лекции:
слушайте внимательно!

Хеширование (hashing)

- Очень часто используется на практике
- В обыденных сценариях чаще, чем balanced trees
- Основные причины
  - Простота реализации
  - Скорость работы (меньше случайных обращений к оперативной в памяти; части таблицы кэшируются)
  - Легкость распараллеливания (*Q*: *зачем?*)

Яркие примеры использования:

- ✓ Хэш-таблицы (1953 г.)
- ✓ Фильтр Блума (1970 г.)
- ✓ Кукушкин фильтр (2014 г.)

} Современные СУБД, распределенные системы

«Постановка задачи»

Входные данные

- Объекты вида (**key**, **payload**)
- **key** – ключ, **payload** – полезная нагрузка (опустим и в дальнейшем будем рассматривать только **key**)

Цель – предоставить эффективный «контейнер» объектов: **вставка, поиск, удаление по ключу (key)**

Структура данных

- множество/словарь

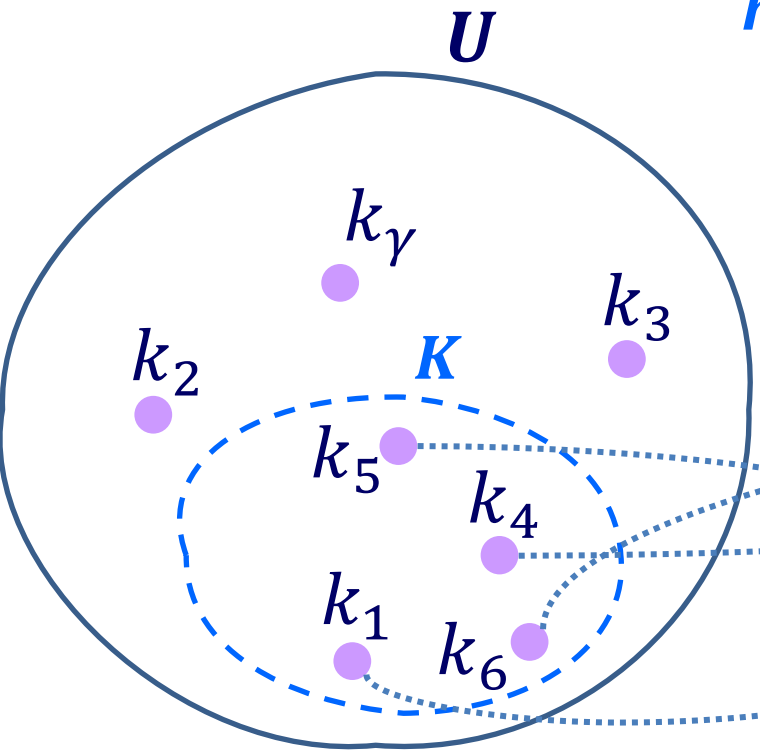
Поддерживаемые операции

- Поиск (search/find/…)
- Вставка (insert/put/…)
- Удаление (remove/delete/…)

$O(1)$ в среднем, на практике трудно outperform хеш-таблицы

Хэш-таблица: идея

Входные данные



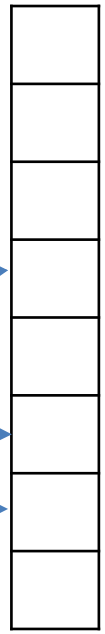
Хэш-функция

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

$T(h) = O(1)$

$h(k)$ – **хэш-значение ключа** $k \in U$

Хэш-таблица, размер $m \approx \Theta(|K|)$



$h(k_4) = h(k_6)$

Коллизия

$h(k_5)$

$h(k_1)$

2 classical способа решения

Предположение $|K| \ll |U|$

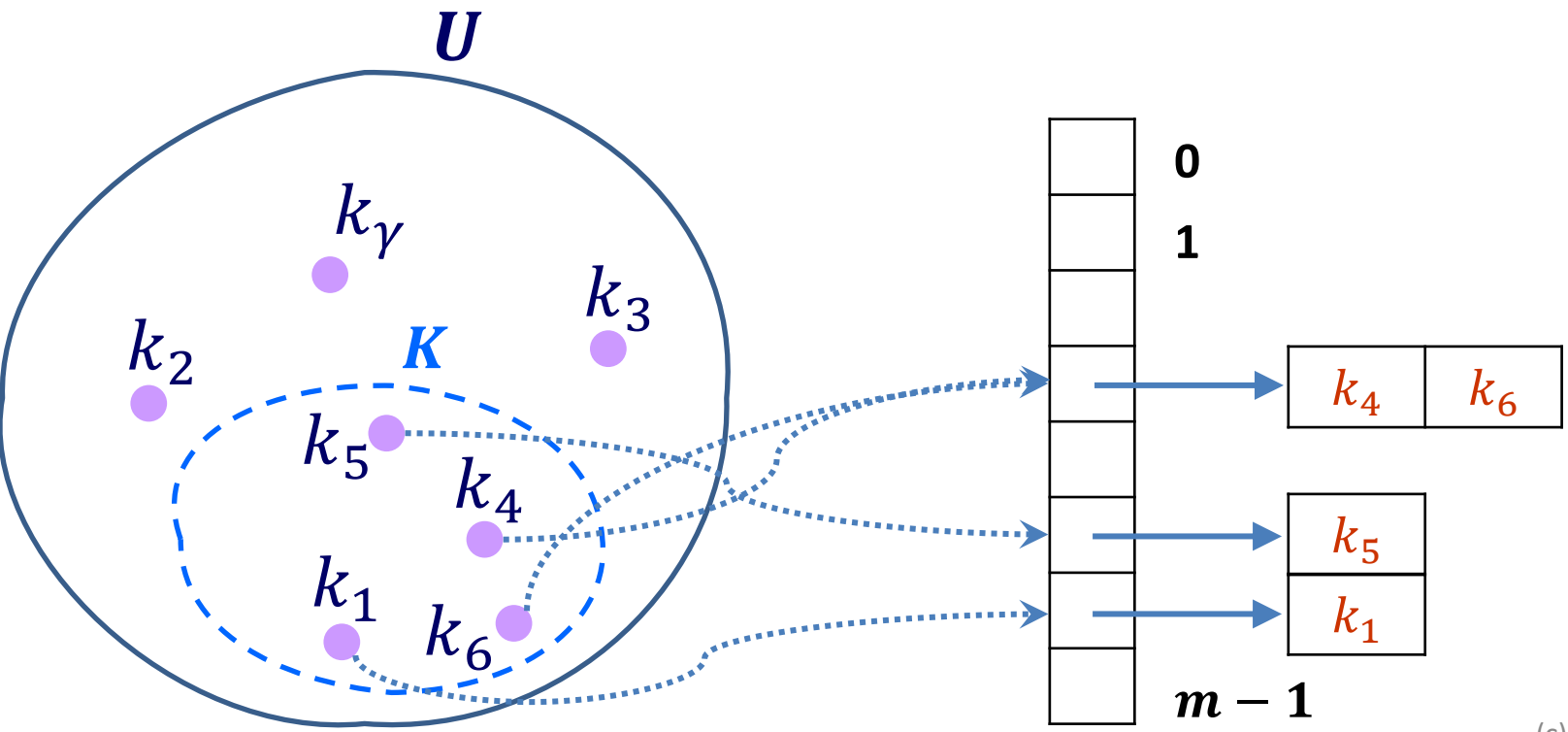
Универсум ключей – множество всех возможных ключей

$U = \{k_1, k_2, k_3, \ldots, k_\gamma\}$

$K = \{k_1, k_4, k_5, k_6\}$ – реально сохраненные ключи

Разрешение коллизий при помощи цепочек

- В каждой ячейке указатель на список объектов
- Вставка нового элемента в начало списка за $O(1)$
- Худшее время $\Theta(n)$ – все элементы в одной цепочке
- Время работы *в среднем* зависит от качества хэш-функции, она должна быть *равномерной*



Хеширование с цепочками: неуспешный поиск

Если

- Разрешение коллизий – цепочки
- Простое равномерное хеширование

Тогда

$$E[\text{время неуспешного поиска}] = \Theta(1 + \alpha)$$

Хеширование с цепочками: успешный поиск

Вспомним, $\alpha = n/m, m = const$

$$E[\text{время успешного поиска}] = \Theta(1 + \alpha)$$

(такое же, как и время неуспешного поиска)

NATIONAL RESEARCH UNIVERSITY

Благодарю за внимание!

Рамон Антонио Родригес Залепинос
arodriges@hse.ru