NATIONAL RESEARCH UNIVERSITY

АЛГОРИТМЫ И СТРУКТУРЫ ДАННЫХ

Хеширование (часть 2 из 2)

Рамон Антонио Родригес Залепинос
arodriges@hse.ru

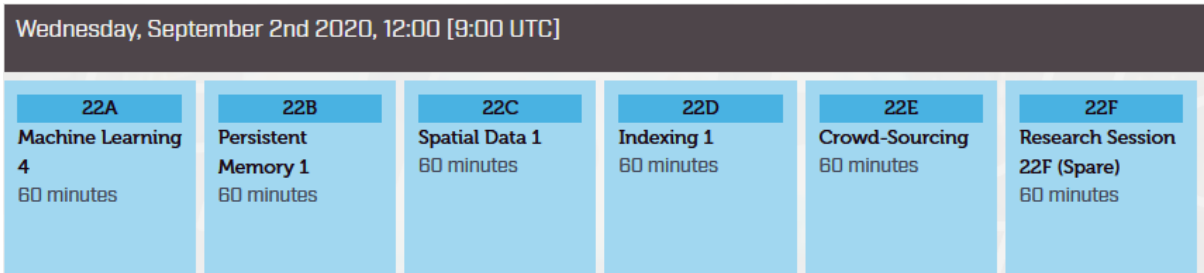
Структура модуля 2

	№		Дата	Тема лекции	№	Домашние задания
Модуль № 2	7	1	27 окт	Хеширование, хэш таблицы 2	2b	ДЗ-Каникулы
	8	2	03 ноя	Фильтры	3	Задание на C++
	9	3	10 ноя	СД для вторичной памяти	4	Задание на C++
	10	4	17 ноя	Пространственные СД	5	Задание на C++
	11	5	24 ноя	Параллельные СД	6	Задание на C++ и/или C# (зависит от выбранного уровня сложности)
	12	6	01 дек	Параллельные СД 2		
	13	?	08 дек	Деревья в оперативной памяти		Примерно за 2 недели заканчиваются ДЗ
	14	8	15 дек	Современные тренды		
				СЕССИЯ с 21.12.2020		

Highlights:

- лекции, семинары и ДЗ синхронизированы
- ? – возможно будет еще одна лекция, с другой темой
- некоторые представления об эффективности СД будут развеяны:
  - на семинаре вы собственноручно, на практике сравните производительность красно-черных деревьев и хэш таблиц (если не нужны next & prev)
- пространственные СД – обширный класс
  - в современном мире, около 80% всех данных содержат географическую привязку: ссылка 1 (Forbes), ссылка 2 (Carto)
  - отдельные секции на значимых конференциях (e.g., VLDB: https://vldb2020.org/program.html)

Требование к студентам на лекции: слушайте внимательно!



Польза КДЗ: Roaring Bitmaps & Invisible Join

Возможно сейчас не до конца понятны

- все преимущества bitmaps
- тонкости современных CPU
- другие нюансы

В этом случае мы работаем на будущее:

- вы будете постепенно получать новые, дополнительные знания
- обновлять свое представление о bitmaps в ходе своей карьеры

Дополнительная польза: вы также

- изучали профессиональный код RoaringBitmaps
- разбирались со статьями о bitmaps

Надеемся, что опыт выполнения КДЗ

будет полезным для Вашей будущей карьеры!

Если распределение ключей известно…

Например,

K – вещественные числа, причем в диапазоне $[0,1)$ и равномерно распределены

$k \in U, 0 \leq k < 1$

Тогда

$$h(k) = \lfloor km \rfloor$$

удовлетворяет требованию простого равномерного хеширования

Хорошая хеш-функция не должна коррелировать с закономерностями, которым подчиняются ключи

Метод деления

Простейший способ

$$h(k) = k \; mod \; m$$

Каких значений m избегать?

- Например, если $m = 2^p$, то $h(k)$ - просто p младших бит числа k
- Лучше, чтобы значение хеш-функции зависело от всех бит ключа
- m может быть простым числом, достаточно далеким от степени 2

Метод умножения

$$h_A(k) = \lfloor m(Ak \ mod \ 1) \rfloor$$

- Генерирует значения в диапазоне $\{0, \dots, m-1\}$
- $Ak \ mod \ 1$ получает дробную часть Ak
- Мы перестаем зависеть от m
- Значение A должно быть аккуратно подобрано

Например (константа Дональда Кнута):

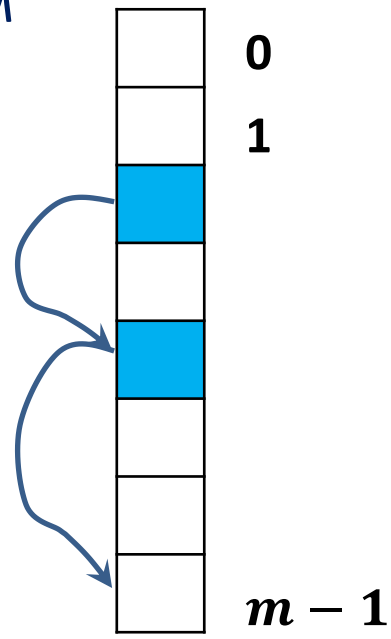
$$A \approx \frac{\sqrt{5}-1}{2} = 0.6180339887 \dots$$

Хеш-таблицы с открытой адресацией

В случае коллизии (ячейка занята) ищем другое свободное место (ячейку) в этой же таблице до m раз

- Все объекты непосредственно хранятся в таблице
- Каждая ячейка содержит либо объект либо NIL
- Таблица может оказаться заполненной (при этом вставка более невозможна)
- Экономия памяти (нет списков, указателей на списки) ➔ можно создавать хеш-таблицы бо́льшего размера, легче хранить в кэшах

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



$$h: U \times \{0,1,\ldots,m-1\} \mapsto \{0,1,2,\ldots,m-1\}$$

Очередной поиск места – испытание (probe)

$$\langle h(k,0), h(k,1), \ldots, h(k,m-1) \rangle$$

Линейное исследование

Пусть вспомогательная хеш-функция задана в виде

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

Тогда данном методе используется хеш-функция

$$h(k,i) = (h'(k) + i) \; mod \; m$$

где i принимает значения в диапазоне $[0, m)$

Недостаток: вероятность заполнения пустой ячейки, которой предшествуют i заполненных ячеек $=(i+1)/m$

Образуются длинные серии заполненных ячеек

Квадратичное исследование

Пусть вспомогательная хеш-функция задана в виде

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

Тогда данном методе используется хеш-функция

$$h(k,i) = (h'(k) + \mathrm{c}_1 i + \mathrm{c}_2 i^2) \; mod \; m$$

где i принимает значения в диапазоне $[0, m)$, $c_1, c_2 \neq 0$

Неформально: «прыгаем» дальше от заполненной ячейки, чем в линейном исследовании

Двойное хеширование

Две вспомогательные хеш-функции

$$\mathrm{h}_1, h_2 : U \mapsto \{0,1, \dots, m-1\}$$

Тогда данном методе используется хеш-функция

$$h(k,i) = (h_1(k) + ih_2(k)) \; mod \; m$$

где i принимает значения в диапазоне $[0, m)$

$h_2(k)$ и m должны быть взаимно простыми, например, $m = 2^p$, $h_2(k)$ возвращает только нечетные значения

Обыкновенная кукушка

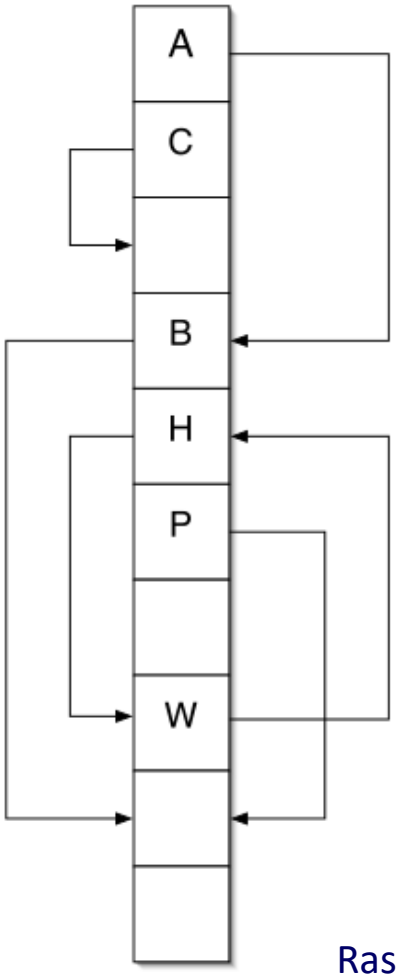
Кукушонок выбрасывает яйца хозяев из гнезда.

Птенец кукушки превосходит по размеру своих «приемных родителей» в два-три раза.

https://nat-geo.ru/fact/pernatye-egoisty/

Кукушкино хеширование (предложено в 2001 г.**)**

- Можем хранить элемент в ячейке $h_1(x)$ либо $h_2(x)$
- Если обе ячейки заняты, вытолкнуть элемент из $h_1(x)$
- Очередная ячейка занята – выталкивать до n раз, потом «перехешировать» (напр. увеличить размер таблицы в 2 раза)



procedure insert(x)
  if $T[h_1(x)] = x$ **or** $T[h_2(x)] = x$ **then return**;
  pos $\leftarrow h_1(x)$;
  loop n **times** {
    if T[pos] = NULL **then** { T[pos] $\leftarrow x$; **return**};
    $x \leftrightarrow T$[pos];
    if pos= $h_1(x)$ **then** pos$\leftarrow h_2(x)$ **else** pos$\leftarrow h_1(x)$;}
  rehash(); insert(x)
end

Rasmus Pagh, Cuckoo Hashing for Undergraduates, 2006

NATIONAL RESEARCH UNIVERSITY

Благодарю за внимание!

Рамон Антонио Родригес Залепинос
arodriges@hse.ru