Хеш-таблицы

Юрий Литвинов
yurii.litvinov@gmail.com

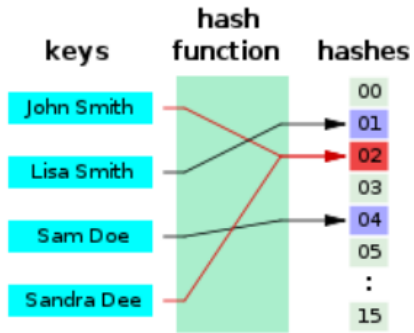
10.11.2020

Хеш-таблица

- Ещё одна реализация АТД “множество” или “ассоциативный массив”
- Требует в среднем константного времени для операций вставки, удаления и поиска
  - Быстрее любых деревьев
  - Очень похожа на массив
- Хранит значения неупорядоченными
- Сильно зависит от качества хеш-функции

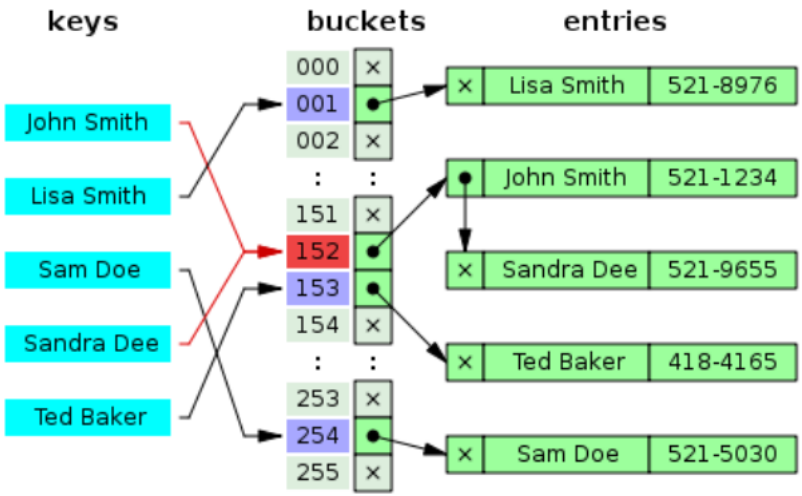
Хеш-функция

- Некоторая функция, отображающая большое (потенциально бесконечное) множество ключей в конечное (и маленькое) множество хеш-значений
  - Не инъективна
- Чем “случайнее” она это делает, тем лучше
  - Немного разным ключам должны соответствовать сильно разные значения
- Зачем
  - Факторизуем множество ключей по классам эквивалентности, образованным ключами с равными хеш-значениями, будем хранить в массиве фактор-множества^U
  - Хеш-значения можно использовать как индексы массива, где лежит что-то, что позволяет найти ключ (сегменты), и чем лучше хеш-функция перемешает ключи, тем меньше вероятность коллизии



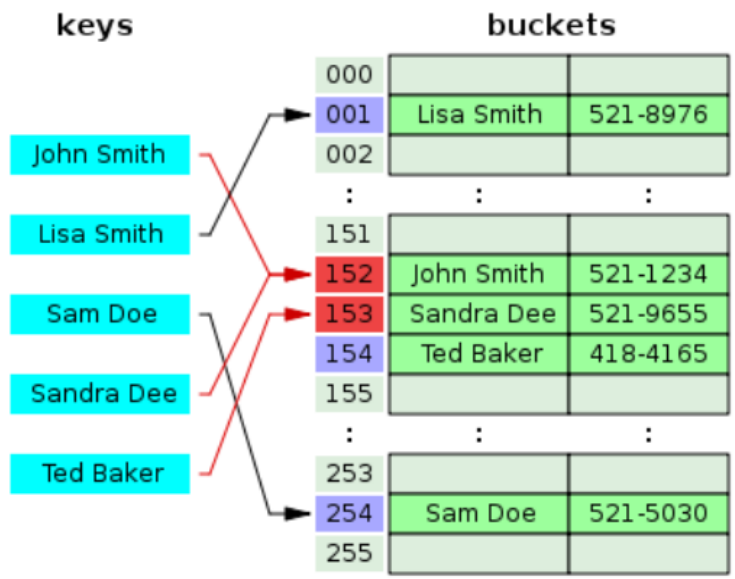
Хеш-таблица со списками значений

- Парадокс дней рождения: В группе, состоящей из 23 или более человек, вероятность совпадения дней рождения (число и месяц) хотя бы у двух людей превышает 50%
- Будем хранить в массиве список ключей с одинаковым хеш-значением
  - Можно и самобалансирующееся двоичное дерево поиска, чтобы быстро найти нужный ключ
    - Но не нужно



Хеш-таблица, “открытая адресация”

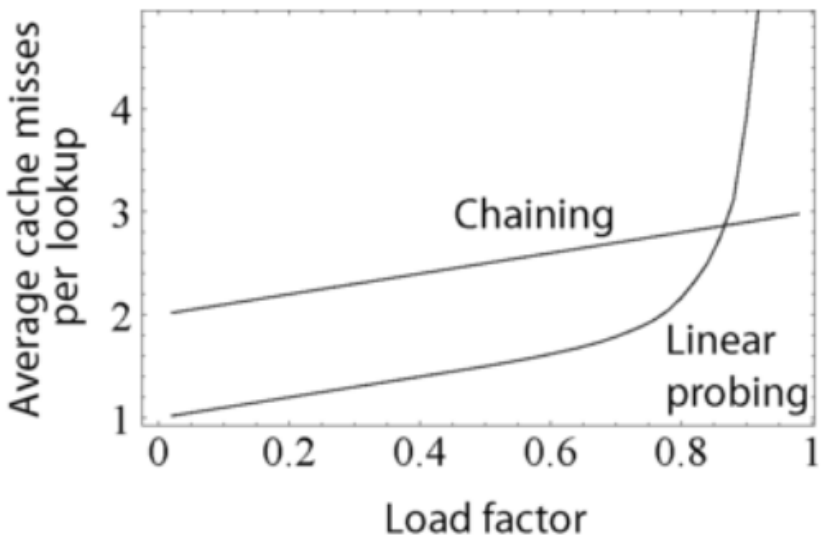
- Второй способ: будем хранить в хеш-таблице сами ключи со значениями, а если ячейка уже занята, брать следующую (может быть, по какому-нибудь сложному правилу)
- Удаление требует дополнительной информации
  - Пустая ячейка будет воспринята как конец цепочки
  - Обычно делают флаг “ячейка была удалена”
    - При вставке — вставляют
    - При поиске и удалении — идут дальше



Коэффициент заполнения

Пусть n — число элементов в хеш-таблице, k — число сегментов (в английской литературе сегменты называются buckets). Коэффициент заполнения хеш-таблицы $L = n/k$

- Коэффициент заполнения должен быть примерно равен 1 для хеш-таблиц со списками и < 0.7 для хеш-таблиц с открытой адресацией
  - Динамическое изменение размеров массива
    - Требуется заново переложить все значения



Выбор хеш-функции

- Должна быть возможно более случайной
- Должна зависеть только от ключа
- Должна считаться быстро
- Например:

```
int h(char *value) {
    int result = 0;
    for (int i = 0; value[i] != '\0'; ++i)
        result = (result + value[i]) % hashSize;
    return result;
}
```

- Обычно хеш-функция возвращает просто целое число, а хеш-таблица сама “загоняет” его в нужный диапазон значений

Ещё про хеш-функции

- Для целых чисел вполне сойдёт id
- “Совершенная хеш-функция” — инъективна
- Универсальная хеш-функция — семейство функций
- Криптографические хеш-функции
  - MD5
  - SHA1
  - Небыстро считаются, поэтому не подходят
- Хеш-функции для сложных типов данных
  - Сумма или произведение хеш-функций элементов, как для строки
    - Значение полинома $a[0] * p^n + a[1] * p^{n-1} + ... + a[n]$, особенно если p — простое (Rolling hash)
  - xor хеш-функций элементов