ХЕШИРОВАНИЕ

ХЕШ - ФУНКЦИЯ

— особое преобразование любого объема информации, в результате которого получается некое отображение, **образ**, называемый **хэшем** (hash) — уникальная короткое значение, которое присуща только этому массиву входящей информации.

Похоже по уникальности на:

- Отпечаток пальца человека
- Структура ДНК человека

Свойства:

- Уникальность
- Необратимость
- Изменяемость
- Быстрая вычислимость

ХЕШ - ТАБЛИЦА

— структура данных, реализующая интерфейс ассоциативного массива. Представляет собой эффективную структуру данных для реализации словарей, а именно, она позволяет хранить пары (ключ, значение),

- выполнять три операции за $O(1)$:
  - операцию добавления новой пары,
  - операцию поиска значения по ключу
  - операцию удаления значения по ключу.

СРАВНИМ С МАССИВОМ

Массив (таблица с прямой адресацией)

- Резервируется много неиспользуемой памяти
- Ограничение по возможным типам индексов
- Долгий поиски при неизвестных ключах $O(n)$

Хеш - таблица

- Снижается требования к памяти до **M** – размер таблицы
- Основа реализации ассоциативного массива
- Все операции быстрые
- Благодаря свойствам применима в большом кластере задач

ГДЕ ИСПОЛЬЗУЕТСЯ ?

- Храним пароли?
- Ассоциативный массив?
- Защита медиафайлов: защита от нелегального распространения ?
- Использует для поимки вирусов ?
- Защита от фальсификации передаваемой информации ?
- Электронная подпись
- Блокчейн?
- Для ускорения поиска данных в БД

ИЗВЕСТНЫЕ АЛГОРИТМЫ

- SHA-1, SHA-2, SHA-3
- MD6, MD5, MD4

ХЕШ-ФУНКЦИЯ

Метод деления

h(key) = key % m,

- **key** – целое числовое значение ключа,
- **m** - размер таблицы, простое число

Метод умножения

h(key) = [m (key · A % 1)]

- ***0 < A < 1*** – константа, рекомендуют золотое сечение: **A = (√5 -1) / 2 ≈ 0.618**
- **m** - размер таблицы

Работа с символами (аддитивный)

while (*str) h += (*str)++

- не различаются схожие слова и анаграммы, т.е. **h(*XY*) = h(*YX*)**
- $m = 256$ (обычно)
- Модификация: используя $\oplus$ (побитово)

Универсальное хеширование

- Подразумевает **случайный** выбор хэш-функции из некоторого множества во время **выполнения** программы
- Например случайные **A** в методе умножения

КОЛЛИЗИЯ

$$\exists\, x \neq y : \; h(x) = h(y)$$

Как бороться с коллизиями?

Стараться предотвратить

- Искать другую хеш – функцию, использовать другие методы формирования
- Подобрать более оптимальную, особенно если есть сведения о свойствах входящих данных или закономерностях

Методы разрешения коллизий

- Цепочек (закрытая)
- Открытая адресация и ее модификации

ХОРОШАЯ ХЕШ - ФУНКЦИЯ

- Быстро вычисляется
- Минимизирует число коллизий

МЕТОДЫ БОРЬБЫ С КОЛЛИЗИЯМИ

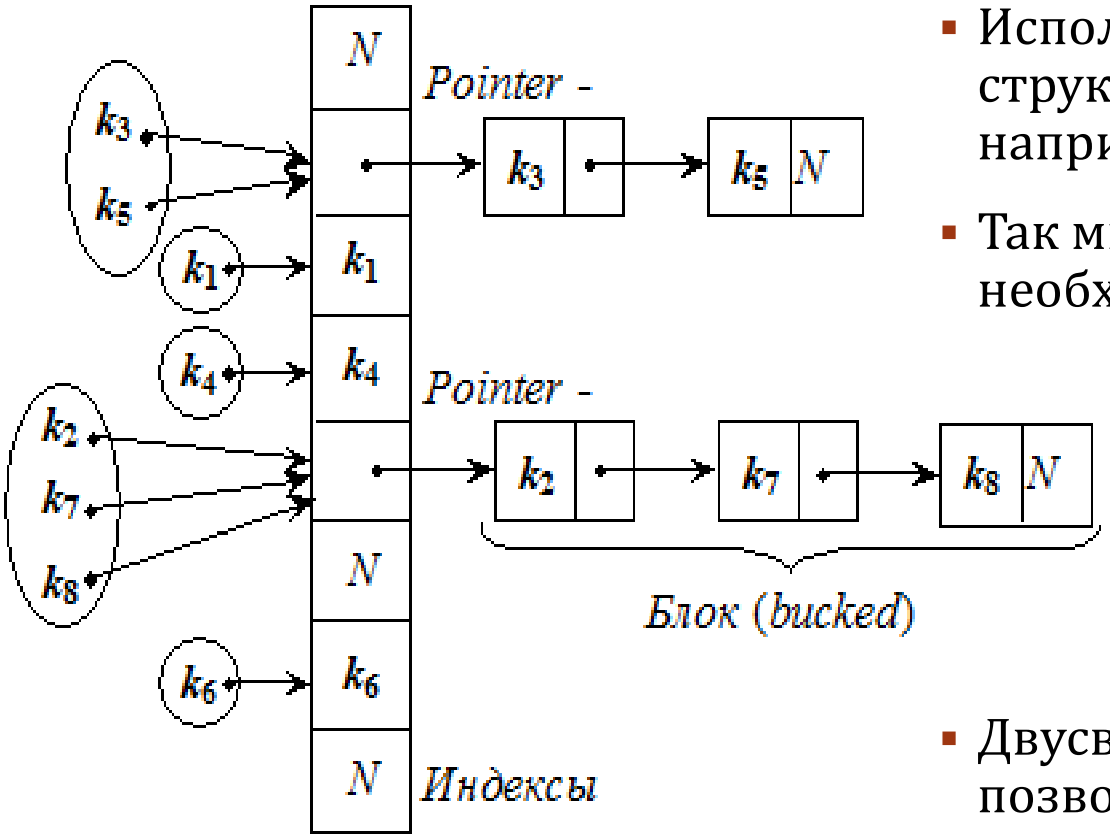
Метод цепочек

- Исходная таблица может быть маленького размера
- Память выделяется по необходимости
- Выбор среди динамических структур велик
- Комбинация структур между собой для оптимизации
- Менее чувствителен к типу ключей
- При частом удалении выгоднее

Открытая адресация

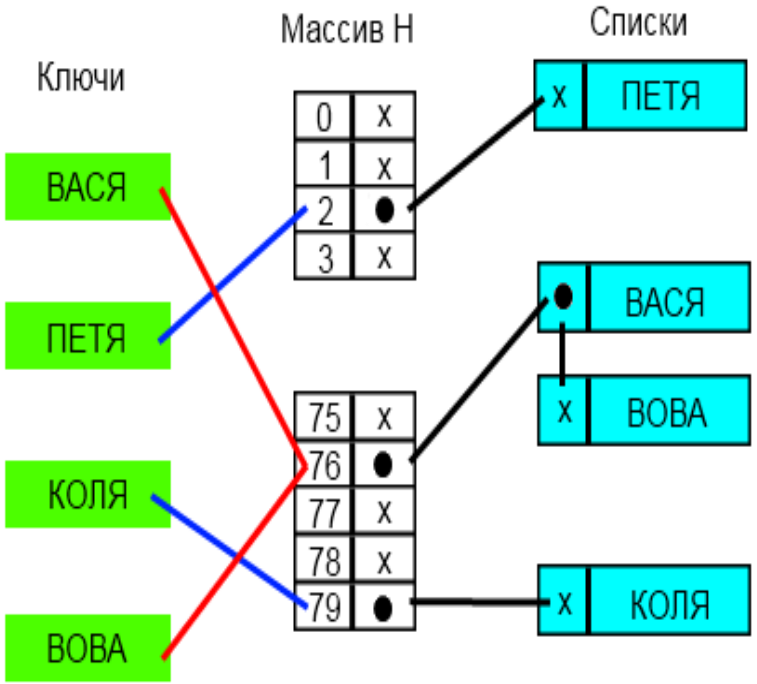
- Таблица фиксирована и непрерывна
- Исследование на порядок быстрее
- Выбор методов исследования большой
- Много выведенных опытным путем оптимальных классов методов
- Проще в реализации и дополнительных ограничениях
- Нет «висячих» ссылок и сложностей управления динам. структурам

МЕТОДЫ ЦЕПОЧЕК



- Используем динамические структуры данных, например, списки
- Так мы используем только необходимую память
- Двусвязыний список удалять позволяет быстрее

МЕТОДЫ ЦЕПОЧЕК (ЗАКРЫТАЯ)



Операции:

- Вставка
- Поиск
- Удаление

Модификации:

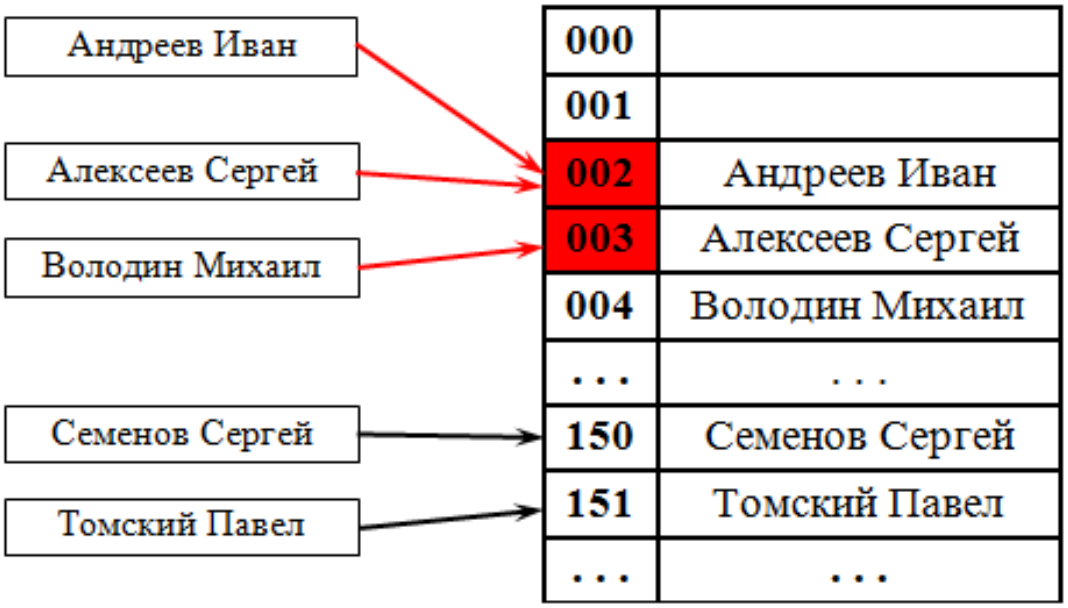
- На двусвязном списке
- Деревья

ОТКРЫТАЯ АДРЕСАЦИЯ

Методы исследования

- Последовательный поиск
- Линейный поиск
- Квадратичный поиск
- Двойное хеширование
- Метод «Кукушки»

ПОСЛЕДОВАТЕЛЬНЫЙ МЕТОД

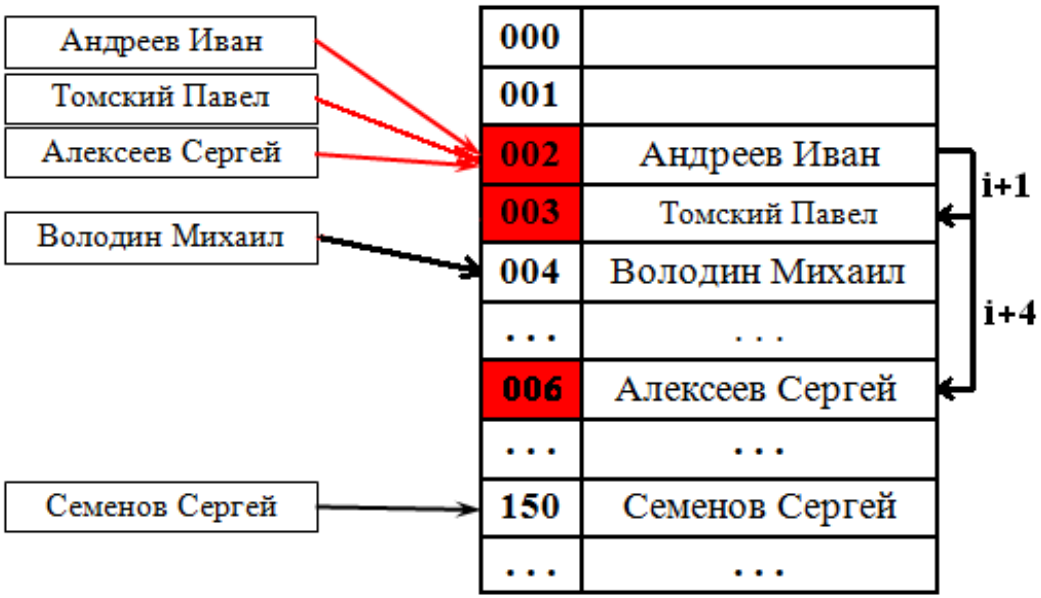


- При попытке добавить элемент в занятую ячейку i начинаем последовательно просматривать ячейки
  - i+1,
  - i+2,
  - i+3
  - и так далее, пока не найдём свободную ячейку.
- В неё и запишем элемент

ЛИНЕЙНЫЙ МЕТОД

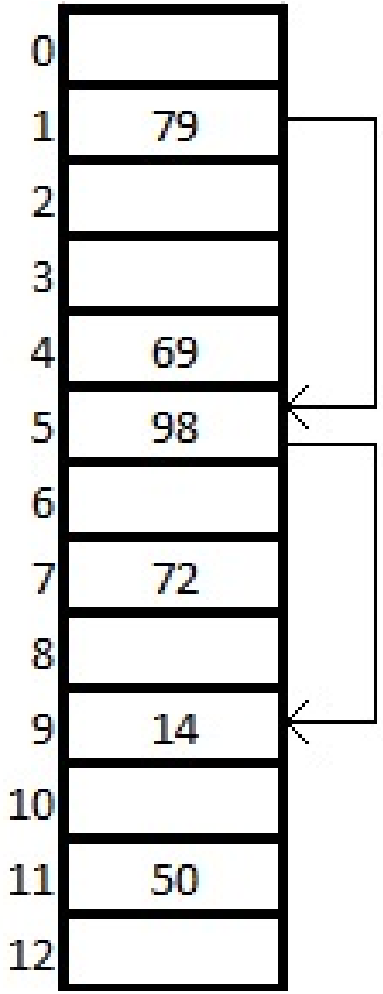
- Выбираем шаг q.
- При попытке добавить элемент в занятую ячейку i начинаем последовательно просматривать ячейки
  - i+(1·q),
  - i+(2·q),
  - i+(3·q)
  - и так далее, пока не найдём свободную ячейку.
- В найденную запишем элемент.
- По сути последовательный поиск - частный случай линейного, где q=1.

КВАДРАТИЧНЫЙ МЕТОД



- Шаг q не фиксирован, а изменяется квадратично: q=1,4,9,16...
- Соответственно при попытке добавить элемент в занятую ячейку i начинаем последовательно просматривать ячейки
  - i+1,
  - i+4,
  - i+9
  - и так далее, пока не найдём свободную ячейку.

ДВОЙНОЕ ХЕШИРОВАНИЕ



— основанный на использовании двух хеш-функций для построения различных последовательностей исследования хеш-таблицы.

(h1(k) + i * h2(k)) % m, i=(0,1,...,m−1)

- вероятность совпадения значений сразу двух **независимых** хеш-функций ниже, чем одной
- h1 - обычная хеш-функцией, чтобы последовательность исследования могла охватить всю таблицу, h2 должна возвращать значения:
  - не равные 0
  - независимые от h1
  - взаимно простые с величиной хеш-таблицы

УДАЛЕНИЕ ИЗ ТАБЛИЦЫ

Использование флага «удалено = true»

- Помечаем удаленные элементы, как **deleted**
- **Insert** – будет рассматривать такую ячейку, как пустую
- **Search** - будет рассматривать, как занятую и продолжать поиск

Восстановление свойств таблицы

при удалении элемента сдвигать всё последующие на q позиций назад

- элемент с другим хешем должен остаться на своём месте
- в цепочке не должно оставаться "дырок

ПЕРЕХЕШИРОВАНИЕ

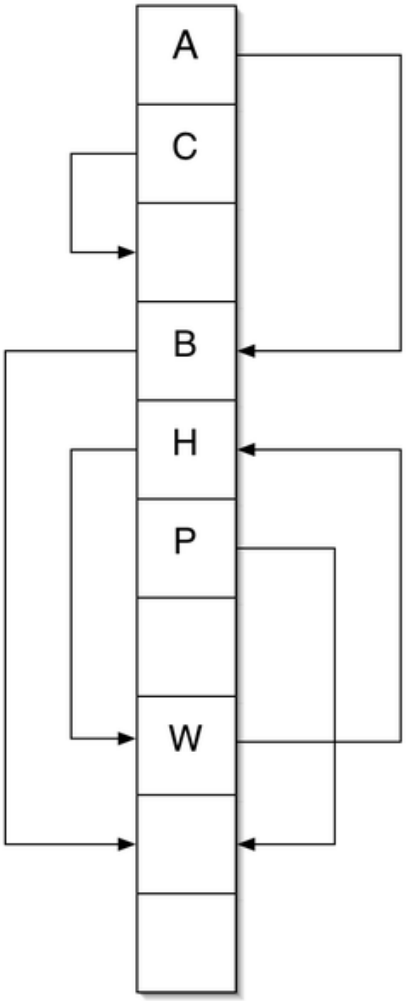
Метод цепочек

- **Условие** когда в хеш-таблицу добавлено **4m/3** элементов
- n — размер хеш-таблицы,
- новую хеш-таблицу размера 2**m**
- сменим хеш-функцию так, чтобы она выдавала значения [0..2**m**−1]
- последовательно переместим в нее все элементы первой таблицы.

Открытая адресация

- **Условие** когда хеш-таблица заполнена на **m/2**
- увеличивая размер таблицы в 2 раза
- сменить хеш - функции под новый размер
- переместить в нее элементы из первой

МЕТОД КУКУШКИ



- Две вариации:
  - Две независимые хеш функции
  - Две таблицы для двух хеш – функций

Алгоритм **insert**

- Если одна из ячеек с индексами h1(x) или h2(x) свободна, кладем в нее
- Иначе запоминаем элемент из занятой ячейки и помещаем туда новый
- Проверяем от сохраненного элемента вторую хеш – функцию
- Если занято, то сохраняем элемент, записываем ранее сохраненный и продолжаем дальше поиск

FAQ

- Что такое хеш – функция и хеш таблица?
- Чем отличается от массива, в чем преимущества ?
- Где применяется хеширование?
- Какая хеш функция является «хорошей»?
- Какими методами можно построить хеш функцию?
- Какие операции реализуются на хеш – таблице?
- Какая оценка операций ?
- Что такое коллизии, как с ними бороться?
- Метод цепочек?
- Методы открытой адресации:
  - Последовательный
  - Линейный
  - Квадратичный ?
- Принцип двойного хеширования?
- Почему двойное хеширования более эффективный метод?
- Как удалять из таблицы при использовании открытой адресации (два подхода)?
- Когда и зачем необходимо перехешировать ?
- Метод кукушки