

Хеширование

Алексей Владыкин

СПбГУ ИТМО

25 февраля 2011

Определения

- Преобразование данных произвольной длины и типа в битовую строку фиксированной длины (n -битное число).
- От англ. «to hash» — рубить, крошить, перемешивать.
- Hash — американское блюдо из говядины, картошки и лука.



- Хеш-таблицы
 - Реализация множеств
 - Реализация отображений
 - Фильтр Блума
- Хеш-функции
 - Алгоритм Рабина — Карпа (см. лекцию про строки)
 - Контроль целостности данных

Множество и отображение

```
public interface Set<T> {  
    void add(T element);  
    void remove(T element);  
    boolean contains(T element);  
    int size();  
}
```

```
public interface Map<K, V> {  
    void put(K key, V value);  
    V get(K key);  
    void remove(K key);  
    boolean contains(K key);  
    int size();  
}
```

- В каких задачах требуются эти типы данных?
- Как реализовать множество через отображение?

Отображения в языках программирования

- **PHP, JavaScript:** все массивы являются хеш-таблицами.
- **Perl, Python:** хеш — самостоятельный встроенный тип данных.
- **Java, C++:** реализации множеств и отображений есть в стандартной библиотеке.
- **C:** «сделай сам».

Listing 1: Perl

```
my %hash;  
$hash{'foo'} = 'bar';  
print "$hash{'foo'}\n";
```

Listing 2: Java

```
Map<String, String> map =  
    new HashMap<String, String>();  
map.put("foo", "bar");  
System.out.println(map.get("foo"));
```

Реализация отображения: идея хеш-таблицы

- Ключ отображения — целое или вещественное число, строка, структура — имеет бесконечное или очень большое конечное множество значений.
- В каждый момент времени отображение содержит лишь небольшое подмножество ключей и соответствующих им значений.
- При помощи хеш-функции можно отобразить множество ключей в множество чисел $\{0, \dots, n - 1\}$ и использовать обычный массив из n элементов.
- Каков тип элемента массива?
- Основная проблема — коллизии, когда $h(k_1) = h(k_2)$.

Стратегии разрешения коллизий

- В i -й ячейке массива хранится связный список элементов с хеш-кодами ключей $h(k) = i$.
- В i -й ячейке массива хранится только один элемент, а новые элементы с тем же хеш-кодом размещаются в других ячейках, выбираемых по какому-либо алгоритму.
Пример: линейная последовательность проб: $h_j(k) = h(k) + js$

Эффективность хеш-таблиц

- Хорошая хеш-функция и заполненность массива меньше 80% дают среднюю эффективность всех операций $O(1)$.
- По мере заполнения массив следует увеличивать.
- Качество хеш-функции:
 - равномерное распределение значений;
 - изменение значения при изменении любого бита ключа.
- Разобрать примеры хеш-функций.

Фильтр Блума

- Отсеивает заведомо безуспешные операции поиска данных по ключу.
- Хеш-коды ключей, имеющих в хранилище, заносятся в побитно упакованную таблицу.
- По таблице можно быстро определить, что ключа нет в хранилище, и не надо тратить время на поиск.
- Наличие хеш-кода ключа в таблице не гарантирует присутствие данных в хранилище, т.к. может иметь место коллизия.

Контроль целостности данных

- Контрольная сумма (checksum) — значение некоторой хеш-функции, используемое для проверки неизменности данных.
- Если контрольная сумма не сходится, то данные испорчены.
- Если контрольная сумма сходится, то данные, скорее всего, верны.
- Примеры: CRC-32, MD5, SHA*.
- Если данные защищаются от преднамеренной подделки, а не просто случайных ошибок, то хеш-функция называется криптографической, и должна быть криптостойкой.

Рекомендуемая литература

-  Кормен Т., Лейзерсон Ч., Ривест Р.
Алгоритмы: построение и анализ.
М.: МЦНМО, 1999. — 960 с., 263 ил. // Глава 12
-  Седжвик Р.
Фундаментальные алгоритмы на С.
К.: ДиаСофт ЮП, 2003. — 1136 с. // Глава 14
-  Столяр С. Е., Владыкин А. А.
Информатика: Представление данных и алгоритмы.
СПб.: Невский Диалект; М.: БИНОМ. Лаборатория знаний,
2007. — 382 с.: ил. // Глава E