

Хеш-таблицы

Шовкопляс Григорий

Введение в алгоритмы и структуры данных

Хеш-таблицы

Дисклеймер

• В данной лекции будет несколько казуальных вилами писанных доказательств, потому что продвинутая теория вероятностей это сложно...



Зачем нужны хеш-таблицы?

Применение хеш-таблиц

- Для абстрактных структур данных set и map
- Set (множество):
 - insert(x)
 - delete(x)
 - contains(x)
- Мар (ассоциативный массив):
 - put(k, v)
 - delete(k)
 - get(k)

Некоторая реализация Мар

 $0 \le k \le M - 1$

М – маленькое

```
массив а[0..М - 1]
put(k, v)
  a[k] = v
get(k)
  return a[k]
delete(k)
  a[k] = null
```

Некоторая реализация Мар

```
0 \le k \le U - 1
U — большое
```

M — маленькое функция $h(k): [0..U) \rightarrow [0..M)$

```
массив а[0..М - 1]
put(k, v)
  a[h(k)] = v
get(k)
  return a[h(k)]
delete(k)
  a[h(k)] = null
```

Некоторая реализация Мар

- Что может пойти не так?
- Коллизия: $x \neq y : h(x) = h(y)$
- Почему коллизии вообще бывают?
- Хорошая хеш-функция: случайная хеш-функция
 - Все немного сложнее, но сегодня считаем так
- Если хеш-функция хорошая, то вероятность коллизии маленькая?

Вероятность коллизии

- Есть массив размера n
- Какое минимальное число элементов нужно запихать туда, чтобы получить коллизию?
- С вероятностью > 0?
 - два
- С вероятностью < 50%?
 - $-\sqrt{n}$
 - Парадокс дней рождений

Парадокс дней рождения

- Если есть группа из хотя бы 23 человек
- Вероятность совпадения дней рождения у двух из них > 50%
- Если группа на 60 человек, то больше 99%
- Казуально: 253 пары людей, вероятность того, что совпадет внутри хотя бы одной пары большая
- p(n) вероятность, что у всех будут различные даты ДР

•
$$p(n) = 1 \times \left(1 - \frac{1}{365}\right) \times \left(1 - \frac{2}{365}\right) \dots \times \left(1 - \frac{n-1}{365}\right) =$$

$$= \frac{365 \times 364... \times (365 - n + 1)}{365^n} = \frac{365!}{365^n (365 - n)!}$$

Парадокс дней рождения

Вероятность коллизии: 1 - p(n)

n	1 - p(n)
10	12%
20	41%
30	70%
50	97%
100	99. 99996%
200	99.999999999999999999999999999999999999
300	$(1 - 7 \times 10^{-73})$ × 100%
350	$(1 - 3 \times 10^{-131})$ × 100 %
366	100%

Вероятность коллизии

- Вывод: коллизии это нормально
- Надо научится их обрабатывать

- Каждая ячейка массива, теперь список
- В случае коллизии храним по одному хешу несколько элементов

Тупенькая реализация Мар

Какие проблемы?

```
put(k, v)
  a[h(k)].add(\{k, v\})
get(k)
  for p : a[h(k)]
    if p.first == k
      return p.second
  return null
delete(k)
  a[h(k)].delete(k)
```

- Оценим время работы (казуально)
- Худший случай O(n)
- В среднем O(n/M)
 - $n \sim M \rightarrow O(1)$

Открытое хеширование (адресация)

Открытое хеширование

- Списки не очень эффективны и хранить сложно
- Давайте обойдемся обычным массивом
- Если случилась коллизия идем дальше, пока не встретим пустую ячейку
- Вставим туда

Открытое Хеширование

Реализация Мар

```
put(k, v)
  i = h(k)
  while a[i] ≠ empty
    i = (i + 1) % M
  a[i] = \{k, v\}
```

Открытое Хеширование

Реализация Мар

```
get(k)
  i = h(k)
 while a[i] ≠ empty
    if a[i].first == k
      return a[i].second
    i = (i + 1) % M
  return null
```

Открытое хеширование

- Удалять научимся позже
- Какого размера выбрать массив, чтобы работало быстро?
- Если M = n, плохо
- Если M = 2n, вероятность, что следующая пустая ½
- Если закончилось место?
 - Расширим массив
 - Поменяем Хеш-функцию и перехешируем
 - Амортизированно O(1), как в векторе

Открытое хеширование

- Почему удалять нетривиально?
- Как тогда удалять?

Открытое Хеширование

План-капкан, не будем удалять явно

```
delete(k)
  i = h(k)
  while a[i] ≠ empty
    if a[i].first == k
      a[i] = \{rip, rip\}
      ripCnt++
      break
    i = (i + 1) % M
  if ripCnt + size > M / 2
    doRehashing()
```

Открытое Хеширование

Как сделать без «заглушек»?

Если правее есть элемент, хеш которого левее перенесем его в дырку, получим новую, и снова

```
delete(k)
  i = h(k)
  while a[i] ≠ empty
    if a[i].first == k
      a[i] = empty
      j = (i + 1) % M
      while a[j] ≠ empty
        if <u>h(a[j].first) <= i</u>
          swap(a[i], a[j])
          i = j
          break
      j = (j + 1) % M
    i = (i + 1) % M
//не рассмотрен случай, когда заканчиваем
```

Откуда берутся хорошие хешфункции

Хеш-функции для чисел

- Случайных хеш-функций не бывает
- Бывают псевдо-случайные
- Семейство хеш-функций
- h(x) = (Ax % P) % M
 - P > М и простое
 - A случайное

Хеш-функции для строк

- Полиномиальный хеш
- $h(S) = ((A^0 s_0 + A^1 s_1 ...) \% P) \% M$
 - P > М и простое
 - А случайное (обычно берут простое больше алфавита)
- $h(S) = ((A^{n-1}S_0 + A^{n-2}S_1 ...) \% P) \% M$
- Почему вероятность коллизии небольшая?

Десерт

Десерт

- Фильтр Блума
- Хеширование кукушки

Bce!