

# 1 Алго ревью. Fixed Set

## 1.1 Описание алгоритма

- 1) Подбор хэш функции из универсального семейства, для достаточно равномерного распределения по бакетам
- 2) Построение хэш таблицы без коллизий для каждого бакета

## 1.2 Доказательство корректности

Множество функций вида

$$h_{ab}(x) = ((ax + b) \bmod p) \bmod m$$

где  $a \in \mathbb{F}_p \setminus \{0\}, b \in \mathbb{F}_p, p \in \mathbb{P}, p > m$

порождают универсальное множество, что дает равновероятность коллизии для всех пар ключей. То есть,  $C$  — количество коллизий:

$$C = \sum_{i < j} \mathbb{I}[h(a_i) = h(a_j)]$$

$$E[C] = \sum_{i < j} P[h(a_i) = h(a_j)]$$

Для каждого  $i$  ровно  $m - i$  подходящих ключей, можно оценить сверху, матожидание количества коллизий:

$$E[C] = \sum_{i < j} P[h(a_i) = h(a_j)] \leq \sum_{i < j} \frac{1}{m} = C_n^k \frac{1}{m} = \frac{n(n-1)}{2m}$$

- 1) Первый уровень хэширования: Будем подбирать  $h(x)$  до тех пор пока, коллизии по бакетам равномерно не располагаются, равномерность будет оценивать суммой квадратов размеров бакетов:

$$\sum_{i=1}^n B_i^2 \leq Kn, K = const$$

$$\sum_{i=1}^n B_i^2 = 2 \sum_{i=1}^n \frac{B_i(B_i - 1)}{2} + n \leq Kn$$

$$\sum_{i=1}^n \frac{B_i(B_i - 1)}{2} = C \leq \frac{n(K - 1)}{2}$$

- 2) Второй уровень хэширования, берем как размер хэш таблицы как  $m = n^2$  и пытаемся найти  $g(x)$  который работает без коллизий и раскидывает ключи в разные подбакеты

Теперь оценим вероятность получения подходящей количества коллизий, по неравенству Маркова:

$$P[C \leq U] = 1 - P[C \geq U + 1] \geq 1 - \frac{\mathbb{E}[C]}{U + 1} \geq 1 - \frac{n(n-1)}{2m(U+1)}$$

$$P[C \leq U] \geq 1 - \frac{n(n-1)}{2m(U+1)}$$

1) Для первого уровня  $m = n, U = \frac{n(K-1)}{2}$

$$P[C \leq U] \geq 1 - \frac{n-1}{n(K-1)+2} \geq 1 - \frac{n}{n(K-1)} \geq 1 - \frac{1}{K-1}, (*)$$

2) Для второго уровня  $m = n^2, U = 0$  :

$$P[C \leq U] \geq 1 - \frac{n-1}{2n} \geq 1 - \frac{n}{2n} \geq \frac{1}{2}, (**)$$

Получается осталось подобрать такую константу  $K$ . Чем она больше, тем больше памяти будет занимать хэш таблица, но будет быстрее найдется подходящая хэш функция. И наоборот.

### 1.3 Алгоритм

Найдем такой  $h(x)$  что итоговая хэш таблица будет иметь линейную память. Она найдется за константное количество попыток(следует из (\*)). Потом раскидаем по бакетам ключи и построим для каждого хэш таблицу размера  $m = n^2$  которая работает без коллизий. Она тоже за константное количество попыток найдет для каждого бакета идеальную хэш функцию  $g(x)$ (следует из (\*\*)). Если в хэш таблицах бакетов сохранить сами ключи, то всегда гарантированно за константное время будет выявлять наличие элемента: сначала спустившись до ключа по значения хэша, потом сравнив истинный ключ и входной ключ на равенство.

### 1.4 Расход времени

- 1) Первый уровень:  $O(n)$
- 2) Второй уровень  $\sum_{i=1}^n O(|B_i|^2) = O(Kn)$
- Итого:  $O(n)$

### 1.5 Расход памяти

$\sum_{i=1}^n O(|B_i|^2) = O(Kn)$  - размер итоговой хэш таблицы