1 Алго ревью. Fixed Set

1.1 Описание алгоритма

1) Подбор хэш фукнции из универсального семейства, для достаточно равномерного распределения по бакетам

2) Построение хэш таблицы без коллизий для каждого бакета

Пусть $n-$ размер хэш таблицы первого уровня. 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 спустившився до ключа по значения хэша, потом сравнив истинный ключ и входной ключ на равенство.

1.2 Доказательство корректности

Множество функций вида

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

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

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:

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

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

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

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

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

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 колличества коллизий, по неравенству Маркова:

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

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

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

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

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

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

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

Теперь точные значения констант:

$a, b-$ подбираются рандомно на каждом шаге из $\mathbb{F}_p$

$p-$ большое простое число, например: $p = 1e9 + 7$

$K = 3$, тогда для обоих уровней вероятность построения подходящей хэш таблицы будет $P \geqslant \frac{1}{2}$, значит матожидание количества попыток для построения - $\frac{1}{P} = 2$

1.3 Расход времени

1) Первый уровень: построение хэш-таблицы за $O(n)$, а ответ на запрос за $O(1)$

2) Второй уровень: построение хэш-таблицы за $\sum_{i=1}^{n} O(|B_i|^2) = O(Kn)$, а ответ на запрос за $O(1)$

Итого: $O(n)$

1.4 Расход памяти

$\sum_{i=1}^{n} O(|B_i|^2) = O(Kn)$ - размер итоговой хэш таблицы. При взятом мной $K = 3$, расход памяти будет линейной.