Хеширование

АТД "множество"

- find(k)
- insert(k)
- erase(k)

Реализации:

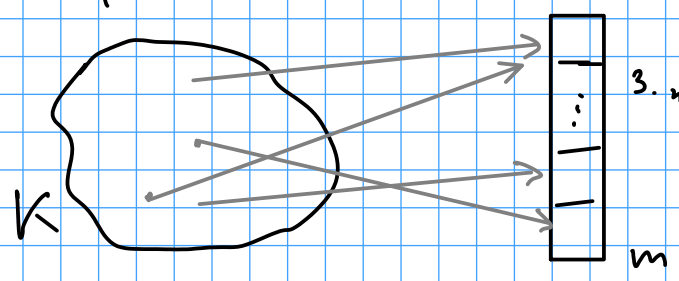
1. "Прямая адресация"
   Заводим массив размера $|K|$
   + $O(1)$ на $\forall$ запрос
   − $O(|K|)$ памяти
2. Дерево поиска
   + $O(\log n)$ на $\forall$ запрос
   + $O(n)$ памяти
   − нужно отношение порядка
3. Хеш-таблица
   + $O(1)$ на запрос в среднем −
   + $O(n)$ памяти

Хеш-функция

$$h: K \to \{0, \dots, m-1\}$$

Хеш-таблица

1. Заводим массив размера m
2. Храним эл-т k в ячейке $h(k)$



3. "Разрешаем коллизии" *

Виды хеш-функций

1. $k \subseteq \mathbb{Z}$

$$h(k) \equiv k \bmod m$$

2. $k \subseteq [0, 1]$

$$h(k) = [mk]$$

2.1 $k \subseteq \mathbb{Z}$

$$h(k) = [m \{kA\}] \bmod m, \quad A = \frac{1+\sqrt{5}}{2}$$

3. Строки

Полиномиальный хеш:

$$S = S_0 S_1 S_2 \dots S_n$$

$$h(S) = S_0 \cdot X^n + S_1 \cdot X^{n-1} + \dots + S_n X^0 \bmod m$$

$$НОД(X, m) = 1$$

Утв: $h(S_0 \dots S_{i+1}) = h(S_0 \dots S_i) \cdot X + h(S_{i+1})$

Следствие: Алгоритм Рабина-Карпа

Можно найти подстроку P в строке T за время $O(|P| + |T|)$ при условии "хорошего" полиномиального хеширования

▷ По тексту T длины n построим массив H: $H[i] = h(T[i] \dots T[i+|P|])$

Пройдём по H и найдём $h(P)$.

Если мы возьмём $m \sim n \Rightarrow O(1)$ коллизий в среднем.

$$H[i] = \left(H[i-1] - h(T[i-1]) x^{|P|}\right) x + h(T[i+|P|])$$

◁

4. IP адреса

$m = 257$

$h(x_1.x_2.x_3.x_4) = a_1x_1 + a_2x_2 + a_3x_3 + a_4x_4 \mod m$

a_1, a_2, a_3, a_4 — случайные (либо простые)

Какая хеш-функция хорошая?

~ Вероятность коллизий $\frac{1}{m}$

$$P_{\substack{k_1, k_2 \\ k_1 \neq k_2}}\left[h(k_1) = h(k_2) \right] \leq \frac{1}{m}$$

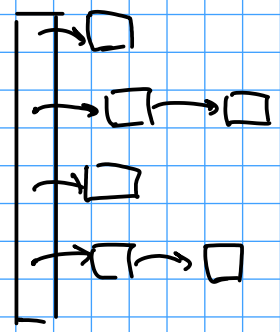
Проблема: Мы не знаем распределение на K

Гипотеза равномерного хеширования

~ $h(k)$ равновероятно на $\{0, \dots, m-1\}$

* Разрешение коллизий

1. Методом цепочек



храним в связных списках

Утв: Поиск отсутствующего элемента занимает в среднем $O(1+\alpha)$, где $\alpha = \frac{n}{m}$ – коэффициент заполнения. (в предположении равномерного хеширования)

▷ $E[\#\text{операций}] = 1 + E[\text{длина цепочки}] =$

$= 1 + \frac{n}{m} = O(1+\alpha)$ (длина цепочки $h(k)$) ◁

Утв: Успешный поиск эл-та в таблице в среднем занимает $O(1+\alpha)$ по всем элементам таблицы.

▷ Не можем применить гипотезу р.х. к эл-ту, который мы ищем, т.к. он уже в таблице присутствует.

Замечание: поиск $\forall$ эл-та требует столько же операций, сколько мы потратили на его добавление.

$$E[\text{успешный поиск}] = \frac{1}{n}\cdot\sum_{i=1}^{n} E[\#\text{операц. insert}]$$

$$= \frac{1}{n}\sum_{i=1}^{n}\left(1+\frac{i-1}{m}\right) = 1+\frac{1}{nm}\cdot\sum_{i=1}^{n}(i-1) =$$

$$= 1+\frac{1}{nm}\cdot\frac{(n-1)\cdot n}{2} = 1+\underbrace{\frac{n}{2m}}_{\alpha}-\frac{1}{2m} = O(1+\alpha)$$

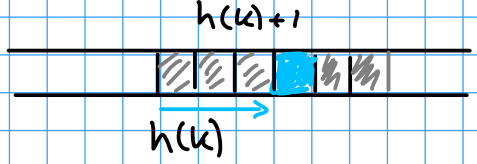
◁

Следствие:

Если $n \sim m \Rightarrow O(1)$ в среднем на $\forall$ операцию с хеш-таблицей.

2. Метод последовательных проб

1. Линейные пробы $\alpha < 1$



При поиске просматриваем $\{h(k)+i\}_i$, пока не встретим пустую ячейку

Проблема: кластеризация

2. Метод квадратичных проб

При поиске будем просматривать $\{h(k) + c_1 i + c_2 i^2\}_i$ пока не встретим пустую ячейку

Проблема: вторичная кластеризация

3. Двойное хеширование

$h : K \times \{0, \dots m-1\} \to \{0, \dots m-1\}$

Просматривает $\{h(k, i)\}$

$$h(k, i) = h_1(k) + i h_2(k)$$

Утв: Поиск отсутствующего эл-та требует $O(1/(1-\alpha))$ операций

▲ $E[\#\text{попыток}] = 1 \cdot (1-\alpha) + 2 \cdot \alpha(1-\alpha) + 3 \cdot \alpha^2(1-\alpha) + \dots =$

$$= \sum_{i=1}^{\infty} i \cdot \alpha^{i-1} \cdot (1-\alpha) = (1-\alpha) \cdot \underbrace{\sum_{i=1}^{\infty} i \cdot \alpha^{i-1}}_{} = (1-\alpha) \cdot \frac{1}{(1-\alpha)^2}$$

$$= \frac{1}{1-\alpha}$$

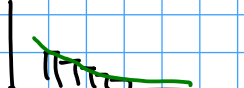
◁

Утв: Поиск эл-та из таблицы занимает в среднем $O\left(\frac{1}{\alpha} \log \frac{1}{1-\alpha}\right)$

▲

$$\frac{1}{n} \sum_{i=1}^{n} E[\#\text{попыток}] = \frac{1}{n} \sum_{i=1}^{n} \frac{1}{1 - \frac{i-1}{m}} = \frac{1}{n} \sum_{i=1}^{n} \frac{m}{m-i+1} =$$

$$= \frac{m}{n} \sum_{i=1}^{n} \frac{1}{m-i+1} = \frac{m}{n} \sum_{x=m-n+1}^{m} \frac{1}{x} \le \frac{m}{n} \int_{m-n}^{m} \frac{1}{x} dx =$$



$$= \frac{m}{n} \log\left(\frac{m}{m-n}\right) = \frac{1}{\alpha} \log \frac{1}{1-\alpha}$$

◁

NB: Проблема этих методов — удаление (нужно помечать удалённые ячейки)