

Хеширование

АТД „множество“

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

Реализации:

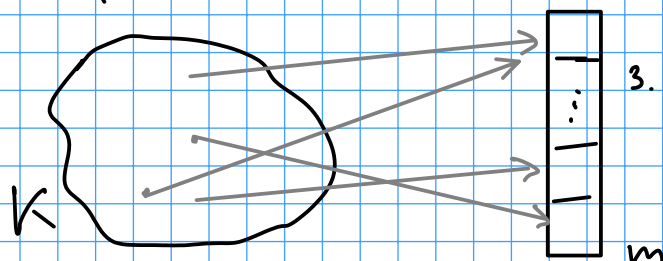
1. „Прямая адресация“
Заводим массив размера $|K|$
+ $O(1)$ на \forall запрос
- $O(|K|)$ памяти
2. Дерево поиска
+ $O(\log n)$ на \forall запрос
+ $O(n)$ памяти
- нужно относительно порядка
3. Хеш-таблица
+ $O(1)$ на запрос в среднем -
+ $O(n)$ памяти

Хеш-функция

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

Хеш-таблица

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



3. „Разрешает коллизии“ *

Вуга хемс - функции

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

$$h(k) \equiv k \pmod{m}$$

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

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

2.1 $k \in \mathbb{Z}$

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

3. Строти

Полиномнальни хемс:

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

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

$$\text{НОД}(x, m) = 1$$

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

Следствие: Алгоритм Радина-карпа

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

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

Пройдем по H и найдем $h(P)$.

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

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



4. IP адреса

$$m = 252$$

$$h(x_1, x_2, x_3, x_4) = a_1 x_1 + a_2 x_2 + a_3 x_3 + a_4 x_4 \pmod{m}$$

a_1, a_2, a_3, a_4 — случайные
(мало простое)

Какая хеш-функция хорошая?

~ Вероятность коллизий $\frac{1}{m}$

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

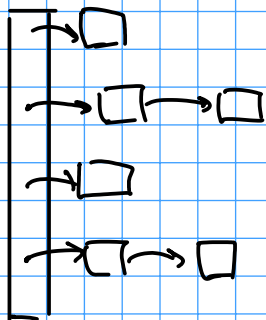
Проблема: Мы не знаем распределение на K

Типично равномерного хеширования

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

* Разрешение коллизий

1. Методом цепочек



хранит в отдельной структуре

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

$$\begin{aligned} \triangleright E[\# \text{ операций}] &= 1 + E[\text{длина цепочки}] = \\ &= 1 + \frac{n}{m} = O(1 + \alpha) \quad \text{в среднем. } h(k) \end{aligned}$$

△

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

Д кэ 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.

$$\begin{aligned} E[\text{успешный поиск}] &= \frac{1}{n} \cdot \sum_{i=1}^n E[\# \text{операций insert}] \\ &= \frac{1}{n} \sum_{i=1}^n \left(1 + \frac{i-1}{m}\right) = 1 + \frac{1}{nm} \cdot \sum_{i=1}^n (i-1) = \\ &= 1 + \frac{1}{nm} \cdot \frac{(n-1) \cdot n}{2} = 1 + \underbrace{\frac{n}{2m}}_d - \frac{1}{2m} = O(1+d) \quad \triangleleft \end{aligned}$$

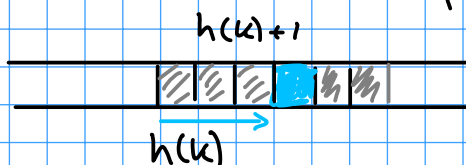
Следствие:

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

2. Метод последовательных проб

1. Линейные пробы

$d < 1$



При поиске проставляем $\{h(k) + i\}$; пока не встретим пустую ячейку

Проблема: кластеризация

2. Метод квадратичных проб

При поиске будем просматривать

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

Проблема: вторичные кластеры

3. Двухэтажное хеширование

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

Просматривает $\{h(k, i)\}$

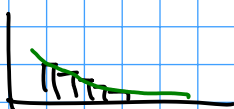
$$h(k, i) = h_1(k) + h_2(k, i)$$

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

$$\begin{aligned} \Delta \quad E[\#\text{коллизии}] &= 1 \cdot (1-\alpha) + 2 \cdot \alpha(1-\alpha) + 3 \cdot \alpha^2(1-\alpha) + \dots \\ &= \sum_{i=1}^{\infty} i \cdot \alpha^{i-1} \cdot (1-\alpha) = (1-\alpha) \cdot \underbrace{\sum_{i=1}^{\infty} i \cdot \alpha^{i-1}}_{= \frac{1}{(1-\alpha)^2}} = (1-\alpha) \cdot \frac{1}{(1-\alpha)^2} \\ &= \frac{1}{1-\alpha} \quad \triangleleft \end{aligned}$$

УТВ: Поиск эл-та у таблицы занимает в среднем $O(\frac{1}{\alpha} \log \frac{1}{1-\alpha})$

$$\begin{aligned} \Delta \quad \frac{1}{n} \sum_{i=1}^n E[\#\text{коллизии}] &= \frac{1}{n} \sum_{i=1}^n \frac{1}{1 - \frac{i-1}{m}} = \frac{1}{n} \sum_{i=1}^n \frac{m}{m-i+1} = \\ &= \frac{m}{n} \sum_{i=1}^n \frac{1}{m-i+1} = \frac{m}{n} \sum_{x=m-n+1}^m \frac{1}{x} \approx \frac{m}{n} \int_{m-n}^m \frac{1}{x} dx = \\ &= \frac{m}{n} \log\left(\frac{m}{m-n}\right) = \frac{1}{\alpha} \log \frac{1}{1-\alpha} \quad \triangleleft \end{aligned}$$



NB: Проблема этих методов — удаление
(можно попытаться удалить элемент)