Haszowanie

wtorek, 21 maja 2024 14:16

wykład nie będzie ciekawy

Problem (stały słownik)

Dane: $k_1, \dots, k_n$

Zadanie: utworzyć strukturę danych (sensowną) umożliwiającą dostęp do kluczy w stałym czasie

Chcemy mieć słownik (insert, find, delete) w tablicy

$h: U \to \{0, \dots, m-1\}$ — z naszego uniwersum → w adresy

insert(k)

miejsce w tablicy H określa h.

$H[h(k)]$

Zapamiętywanie w listach: k_i jest pamiętany w liście $H[h(k_i)]$

Jak dobrać h

Schemat urnowy

m - liczba urn

n - liczba kul

$m = n$

Jaka jest oczekiwana liczba kulek w 1-rej urnie

X = liczba kulek w pierwszej urnie

$$x_i = \begin{cases} 1 & \text{jeśli kulka } i\text{-ta jest w pierwszej urnie} \\ 0 & \text{wpp} \end{cases}$$

$$X = \sum_{i=1}^{n} x_i$$

$$E[X] = \sum_{i=1}^{n} E[x_i] = 1, \quad E[x_i] = \frac{1}{m}$$

Niech X - maksymalna liczba kulek w jednej urnie

Fakt $E[X] = O\left(\frac{\log n}{\log\log n}\right)$

Są funkcje zachowujące się jak funkcje losowe

- $h(k) = k \bmod m$
- $h(k) = \lfloor m \cdot (k \cdot A - \lfloor k \cdot A \rfloor) \rfloor$, $A \in (0,1)$, $k \in \mathbb{N}$ lub jakiś napis, sprytnie

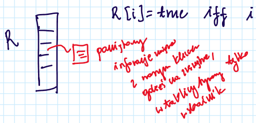
Mamy Uniwersum i klucze od k_1 do k_m

$U = \{k_1, \dots, k_m\}$

$S \subseteq U$ → tworzymy słowniki

Gdy $|U|$ jest małe, to może warto tworzyć wektor charakterystyczny

$R[i] = true$ iff $i \in S$



pamiętamy informacje wraz z innymi danymi, gdzie są trzymane tylko w tablicy danych klucz

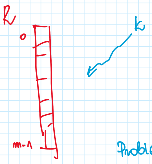
Find(k), Insert(k), Delete(k) } $O(1)$

ale tylko działa jak U małe

Co gdy $|U|$ duże? A $|S|$ nie za duże

Chcemy użyć pamięci proporcjonalnej do S

$|R| = O(S)$



Czarowi mowa: f. która będzie umiała policzyć, k umieścić

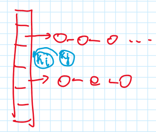
Wstawiamy klucze $\{k_1, \dots, k_n\}$

Problem będzie gdy $h(k_i) = h(k_j)$ gdzie $j < i$

Kolizje

Gdy U wielkie to kolizje są prawdopodobne do uniknięcia

Pomysł: Wstawiając klucze do tablicy będziemy je ustawiać na listę $h(k_i)$ a nie w pole k_i

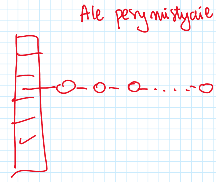


Insert $(k_i) = O(1)$ jeśli tylko wyliczenie wartości h() jest w czasie stałym (element wstawiamy na początek listy)

Find (k) - O(długość listy h(k))

Delete (k) — tak samo

Ale pesymistycznie wszystkie elementy mogą trafić do jednej listy :(



nie znamy za wiele kluczy w S wcześniej

Inny sposób pamiętania elementów

- adresowanie otwarte

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

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

$h(0,k) + h'(k)$

$\langle h(0,k), h(1,k), \dots, h(m-1,k) \rangle$ ← permutacja liczb $0, \dots, m-1$

Najpopularniejsze metoda liniowa

$h(i,k) = (h'(k) + i) \bmod m$

h(5) jak nie będzie wolne to wypróbujemy następną (6) itd.

długi ciąg będzie w komórkach pod tym pierwszym, bo wtedy mamy największe szanse na trafienie. One są bardziej prawdopodobne

nie mamy list, tylko tablicę

Adresowanie otwarte → klucze pamiętamy w tablicy

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

klucz — numer próby wstawiania klucza — lokalizacja w tablicy

Tendencja do zlepiania się

Ta komórka będzie miała dużo większe ppb trafienia

Metoda kwadratowa

$$h(i,k) = (h'(k) + c_1 \cdot i + c_2 i^2) \mod m$$

$c_1, c_2 \to$ stałe

będzie zlepek ale zależny od stałych

Ale jest to lepsze niż w liniowej, bo będzie więcej poprawek i inne próby więc trochę mniej zlepione

podwójne haszowanie

$$h(i,k) = (h'(k) + i \cdot h''(k)) \mod m$$

h' – wyznacza start; h'' – poprawki skoku

Idealistycznie

ciąg $h(0,k), \dots, h(m-1,k)$ jest z jednakowym prawdopodobieństwem dowolną permutacją liczb $0, \dots, m-1$

$\leq \frac{1}{1-\alpha}$ $\quad a = \frac{n}{m}$ $\quad$ jest porażka przy poszukiwaniu

$\leq \frac{1}{\alpha} \cdot \ln \frac{1}{1-\alpha}$ $\quad$ jest sukces

k_i

$p_i \Rightarrow$ ppb że wykonamy i prób

Chcemy policzyć oczekiwaną liczbę prób:

$$E = 0 \cdot p_0 + 1 \cdot p_1 + \dots + i \cdot p_i$$

Niech q_i to ppb wykonania co najmniej i prób

$$p_i = q_i - q_{i+1}$$

p_i – ppb że wykonamy dokładnie i prób

Podstawiając: $E = q_1 - q_2 + 2 \cdot (q_2 - q_3) + 3 \cdot (q_3 - q_4) + \dots =$

$= q_1 + q_2 + q_3 + q_4 \dots$ = czyli to ciąg geometryczny $\leq \frac{1}{1-\alpha}$

$q_1 = 1$

$q_2 = \frac{n}{m}$ ($n \Rightarrow$ liczba zajętych komórek, $m \Rightarrow$ liczba miejsc w tablicy)

q_2 – pierwszy strzał w zajętą komórkę, drugi dobry

$q_3 = \frac{n}{m} \cdot \frac{n-1}{m-1} \leq \left(\frac{n}{m}\right)^2$

gdzie $\alpha = \frac{n}{m} < 1$

Rodzina funkcji haszujących

$\longrightarrow$ losujemy i używamy

DEFINICJA

Rodzina $\mathcal{H}$ funkcji haszujących z $\mathcal{U}$ w $\{0, \dots, m-1\}$ jest uniwersalna jeśli

$$\forall_{x,y \in \mathcal{U},\ x \neq y} \quad |h: h \in \mathcal{H}, h(x) = h(y)| \leq \frac{|\mathcal{H}|}{n}$$

Przykłady

$|\mathcal{U}| < m^{r+1}$

$\forall a \in \langle 0, m^{r+1}) \quad h_a(x) = \sum_{i=0}^{r} a_i x_i \mod m$

$a_0, a_1, \dots, a_r$ – reprezentacja liczby a w systemie m-arnym

$x_0, x_1, \dots, x_r$ – — || — x

— || —

Przykład 2

Niech p – liczba pierwsza $|\mathcal{U}| < p$

$\forall_{a \in \mathbb{Z}_p^*,\ b \in \mathbb{Z}_p}$ (bez 0) definiujemy $h_{ab}(x) = ((a \cdot x + b) \mod p) \mod m$ (rozmiar tablicy)

$$\mathcal{H}_{pm} = \{h_{ab} : a \in \mathbb{Z}_p^* \ \& \ b \in \mathbb{Z}_p\}$$

FAKT

$\mathcal{H}_{pm}$ jest rodziną uniwersalną

Niech $h'_{ab}(x) = (ax + b) \mod p$

Niech $k \neq l$ różne klucze

$s = h'(k)$

$t = h'(l)$

Spostrzeżenia:

1) $s \neq t$

nie wprost: że są równe

$s = (ak + b) \mod (p) = (al + b) \mod p = t$

$0 = a(k - l) \mod p$ – a: z zał. $\neq 0$; $(k-l)$: różne; zatem sprzeczność ↯

tu były jakieś rachunki

Każda para s, t (t. że $s \neq t$) jest obrazem pary k, l dla pewnej funkcji h'_{ab}

$$t = (ak + b) \bmod p$$

$$s = (al + b) \bmod p$$

$$t - s = (a(k - l)) \bmod p$$

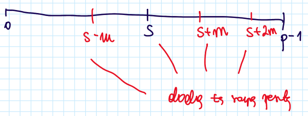
$$a = (t - s)(k - l)^{-1} \bmod p$$

↓ element odwrotny w Z_p, więc a będzie jednoznacznie wyznaczone

Stąd $b = (t - ak) \bmod p$

Czyli istnieje bijekcja między wszystkimi funkcjami, a wszystkimi takimi parami

Chcąc policzyć ppb wylosowania funkcji z H_{pm}, która daje kolizję na kluczach k, l wystarczy policzyć ppb wylosowania pary liczb s, t t. że $s \neq t$ i $s \equiv t \bmod m$



tych liczb jest $\lceil \frac{p}{m} - 1 \rceil$, bo nie wyłączając s, bo nie jest różny od samego siebie

$$\left\lceil \frac{p}{m} - 1 \right\rceil \leq \frac{p+m-1}{m} - 1 = \frac{p-1}{m}$$

stąd już teza

dowód początek wykładu 22