RAD (5) Hashing

Strærk Universalitet

Strongly universal hash functions

Recall hash function $h : U \to [m]$ universal if, for all $x \neq y \in U$:

$$\Pr[h(x) = h(y)] \leq 1/m.$$

Hash function $h : U \to [m]$ strongly universal if, for all $x \neq y \in U$, $(h(x), h(y))$ uniform in $[m]^2$, that is, for any $q, r \in [m]$:

$$\Pr[(h(x), h(y)) = (q, r)] = 1/m^2.$$

Strong universality implies universality:

$$\Pr[h(x) = h(y)] = \sum_{q \in [m]} \Pr[(h(x), h(y)) = (q, q)] = m/m^2 = 1/m.$$

Equivalent definition of strong universality:

pairwise independence $\forall x \neq y \in U$, $h(x)$ and $h(y)$ independent.

uniformity $\forall x \in U$, $h(x)$ uniform in $[m]$.

Multiply-mod-prime

Let $U = [u]$ and pick prime $p \geq u$. For any $a, b \in [p]$, and $m < u$, define $h^m_{a,b} : U \to [m]$ by

$$h^m_{a,b}(x) = ((ax + b)\ mod\ p)\ mod\ m$$

hvis a er uniform i $[p]_+ = \{1, \ldots, p-1\}$
og b er uniform i$[p] = \{1, \ldots, p-1\}$
så er $h^m_{a,b} : U \to [m]$ en universal hash function
Tager konstant tid og plads

```
MultiplyModPrime(x)
        q = 89
```

```
p = 2^q - 1 // mesinner prime
a,b = random.Bigint
y = ((a*x+b) & p) + ((a*x+b) >> q)
if (y >= p)
        y -= p
return y & ((2^l - 1))
```

Multiply Shift

Def: en hash funktion $h : U \to [m]$ er c-approksimativt universal hvis, $\forall x \neq y \in U : Pr[h(x) = h(y)] \leq c/m$.

Let $u = 2^w$ and $m = 2^l$ være toer-potenser. lad a være uniformt tillfældig ulige integer i $[u]$. Definer *mulitply-shift* hash funktionen $h_a : [u] \to [m]$ ved:

$$h_a(x) = \lfloor \frac{(ax) \ mod \ 2^w}{2^{w-l}} \rfloor$$

Dertil er h_a 2-approksimativt universal
C kode: $(a * x) >> (w - l)$

Fremlæggelse

Universality

A hash function $h : U \to [m]$ is universal if, for all

$$x \neq y \in U : Pr[h(x) = h(y)] \leq 1/m$$

Is a truely random h also universal? Yes, but we also have practical universal hash functions. Giver lille kollisions sandsynlighed

Hashfunktioner

hvad er en hashfunktion kort

- **Multiply-mod-prime**
  Tager konstant tid og plads

$$h^m_{a,b}(x) = ((ax + b)\ mod\ p)\ mod\ m$$

  hvis a er uniform i $[p]_+ = \{1, \ldots, p-1\}$
  og b er uniform i$[p] = \{1, \ldots, p-1\}$
  så er $h^m_{a,b} : U \to [m]$ en universal hash function
- **Multiply shift**

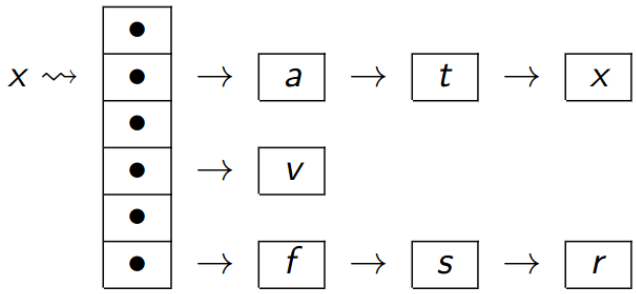
Den ene er hurtigere fordi den bruger færre multiplikationer. De er begge universelle.

Hash tabel

Vores mål: vellighold at S ⊆ U, |S| = n, so et x ∈ U, sig hvis x ∈ S.

ide: lad $m \geq n$ vælg en universal hashfunktion $h : U \to [m]$. Gem et array L hvor L[i] er en linked list over $\{y \in S | h(y) = i\}$

Then $x \in S \iff x \in L[h(x)]$.



Membership checked in $\mathcal{O}(|L[h(x)]| + 1)$ time

Total space $\mathcal{O}(n + m)$.

Can also add/remove x in $\mathcal{O}(|L[h(x)]| + 1)$ time.

Bevis at, for x∈S, E[|L[h(x)]|] ≤ 1

antag at $m \geq n$, vil vise at for $x \in S, E[|L[h(x)]|] \leq 1$

antallet af elementer der hasher til samme sted som x er antallet af y'er i S hvor dens hashværdi er den samme som x eller de y'er der lander i samme linked list som x.
$|L[h(x)]| = |\{y \in S | h(y) = h(x)\}|$

$$E[|L[h(x)]|] = E[|\{y \in S | h(y) = h(x)\}|] =$$

antallet af elementer i listen er summen af de gange hvor y og x hasher til samme værdi. Det betyder altså at h(y)=h(x) er en indikator variabel, enten hasher de samme værdi ellers gør de ikke.

$$E[\sum_{y \in S} [h(y) = h(x)]] =$$

Bruger linearity of expectation

$$\sum_{y \in S} E[h(y) = h(x)] =$$

Der gælder at forventningen for en indikator variabel er sandsynligheden for at den er 1.

$$\sum_{y \in S} Pr[h(y) = h(x)] \leq$$

antallet af elementer i S eller |S| er n. Denne ulighed holder fordi vi har en universal hashfunktion så sandsynligheden for at vi kolliderer er $\leq \frac{1}{m}$, når vi indsætter et element, men vi skal indsætte $|S| = n$ elementer.

$$|S| \frac{1}{m} = \frac{n}{m}$$

Den forventet søgetid er n/m, men eftersom $m \geq n$ kan brøkken aldrig blive større end 1.

$$\frac{n}{m} \leq 1$$

vi kan altså upperbounde søgetiden til 1 med andre ord er søgetiden O(1). Vi har altså en **Forventet** søgetid på O(1)

Her er vores table dynamisk, så vi kan justere størrelsen

1 level hashing

Den er statisk, så vi vælger én størrelse for tabellen og så bliver den ikke større. Vi vil gerne for faktisk konstant søgetid og ikke forventet som i hashing med chaining.

$$E[C] = \sum_{\{x,y\} \subseteq S} Pr[h(x) = h(y)] \leq$$

Vi har en universal hash funktion sandsyndlighed for kollision 1/m, vi vælger 2 værdier fra S derfor vælger vi 2 værdier ud a |S|=n

$$\binom{n}{2} \frac{1}{m} = \frac{n(n-1)}{2} \cdot \frac{1}{m} = \frac{n^2 - n}{2m}$$

Lad os nu bruge markov til at kigge på Sandynligheden for at kollision:

$$Pr[C > 2E[C]] < \frac{1}{2}$$

Lad $m = n^2$, da bliver $E[C] < 1/2$ fordi vores n^2 spiser $n(n-1)$ som bliver mindre end 1.

$$Pr[C > 2\frac{1}{2}] < \frac{1}{2} \rightarrow Pr[C > 1] < \frac{1}{2}$$

altså gælder der at plads for bruget er $O(n+m) = O(n+n^2) = O(n^2)$
Forventet antal af kollisioner er mindre end 1/2, sandsynligheden for at vi får 1 eller flere kollisioner er højst 1/2

Hvis m=n gælder der $E[C] < \frac{n}{2}$

$$Pr[C > n] < \frac{1}{2}$$

her er pladsforbruget $O(n)$

2 Level hashing

Givet et statisk set S, |n|, har vi nu:

- en hashfunktion $h : U \to [n], S_i = \{x \in S | h(x) = i\}, n_i = |Si|$,

Der gælder for antallet af kollisioner: hvis vi vælger 2 elementer fra n_i er kollisions sandsynligheden mindre end n.

$$C = \sum_{i \in [n]} \binom{n_i}{2} < n$$

- hvert $i \in [n]$, med $n_i > 0$ har en kollisions fri hashfunktion på S_i $h_i : U \to [n_i^2]$
  Dvs der er en hashfunktion der hasher et element ind i den første tabel med størrelse n og så i hashfunktioner der hver især hasher et element ind i den indre tabel som har størrelsen n_i^2

dvs der kan være n tabeller med n_i^2 elementer i.

Bestemmelse af pladsforbrug for 2-level hashing

Vi har en tabel med størrrelse n og så har vi n_i tabeller med størrelse n_i^2

Pladsforburget er derfor følgende:
n er størrelsen for den ydre tabel og summen størrelsen af alle de indre tabeller.

$$O(n + \sum_{i \in [n]} n_i^2)$$

Der gælder at $n_i^2 = n_i + 2\binom{n_i}{2} = n_i + \frac{2n_i(n_i-1)}{2} = n_i - n_i + n_i^2$

$$O(n + \sum_{i \in [n]} (n_i + 2\binom{n_i}{2}))$$

Det gælder at $\sum_{i \in [n]} 2\binom{n_i}{2} = O(2C)$, summen af $\sum_{i \in [n]} n_i = O(n)$

$$O(n + n + 2C) = O(C) = O(n)$$

C er højst n derfor gælder $O(C) = O(n)$.