Eksamensdisposition - Hash tables

Søren Mulvad, rbn601

17. juni 2019

- **Redegørelse for datastruktur og anvendelse**
- **Bestemmelse af forventet opslagstid**
- **Opbygning af hash table der altid sikrer konstant opslagstid**
- **Bestemmelse af pladsforbrug for 2-level hashing**

Eksamensdisposition - Hash tables

Redegørelse for datastruktur og anvendelse

Vi bestemmer en hashfunktion h der mapper et univers U til sættet af tal $[m]$:

$$h : U \to [m] = \{0, \dots, m-1\}$$

For en c-universel hashing gælder, at for $x, y \in U$ hvor $x \neq y$, da:

$$\mathbb{P}[h(x) = h(y)] \leq \frac{c}{m}$$

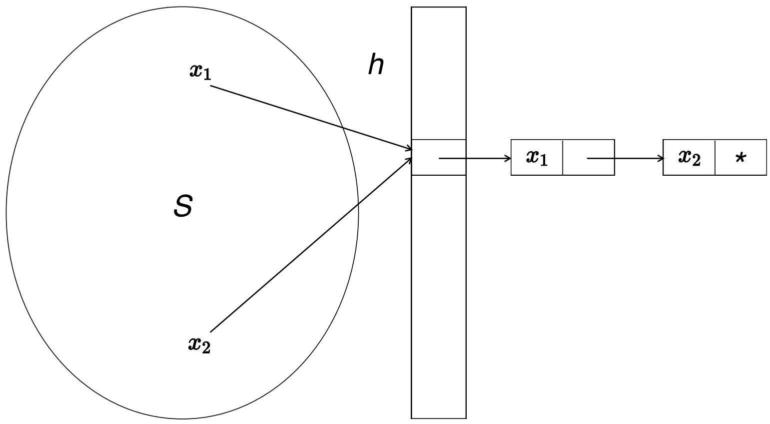
Hvis $c = 1$ kalder vi den blot for universel.

Vi kan implementere en 2-universel hashing funktion på f.eks. følgende måder:

$$((ax + b) \bmod p) \bmod m \qquad\qquad (a'x) >> (w - l)$$

hvor vi i MulShift har at a' skal være ulige og $m = 2^l$.

Hashing funktioner benyttes særligt ofte i det man typisk kalder dictionaries. Lad os sige vi har en liste S med $|S| = n$ nøgle/værdi-par. Så laver et array L der er $m \geq n$ langt med lister og laver en hashing funktion $h : U \to [m]$.
Da vil $L[i]$ bestå af alle de elementer hvis nøgle hashes til dette index i:



Figur 1: Hash tabel med chaining

Vi ser nemt at denne struktur vil bruge $O(n + m)$ plads.

Bestemmelse af forventet opslagstid

Tiden det tager at finde ud af om $x \in U$ er i S er proportional med listens længde, $O(1 + |\,L[h(x)]\,|)$.

Antag nu, at $x \notin S$, h er universel, $m \geq n$ og lad $I(y)$ være en indikator-variabel som er 1 hvis $h(x) = h(y)$ og 0 ellers. Da er det forventede antal elementer i $L[h(x)]$:

$$
\begin{aligned}
\mathbb{E}\left[\,|L[h(x)]|\,\right] &= \mathbb{E}\left[\sum_{y\in S} I(y)\right] && (1)\\
&= \sum_{y\in S} \mathbb{E}\left[I(y)\right] && (2)\\
&= \sum_{y\in S} \mathbb{P}\left[I(y)=1\right] \\
&\leq \sum_{y\in S} \frac{1}{m} && (3)\\
&= n/m && (4)\\
&\leq 1
\end{aligned}
$$

I (1) bruger vi, at antallet af elementer i listen $L[h(x)]$ må være alle dem som hasher til værdien $h(x)$, altså alle y så $h(y) = h(x)$.
I (2) bruger vi Linearity of Expectation.
I (3) benytter vi vores antagelse om at vores hashing function er universel, hvorved uligheden gælder.
I (4) benytter vi, at der i alt er n elementer i S.

Opbygning af dictionary der altid sikrer konstant opslagstid

Antag vi har en universel hash funktion $h : U \to [m]$. hvor $m \geq 1$ er et heltal vi vælger senere. For ethvert $i \in [m]$, definer $S_i = \{x \in S | h(x) = i\}$ og $n_i = |S_i|$. For ethvert sæt S_i er køretiden $O(1)$ hvis det er tomt eller kun indeholder et element. Men hvis mange elementer hasher til samme index vil det tage lang tid.

Lad os definere C_h til at være det totale antal nøgler i S der kolliderer under hashfunktionen h. Da vil $C_h = \sum_{i\in[m]} \binom{n_i}{2}$ (f.eks. vil det for ét indeks i give 0 når $n_i = 0$ eller $n_i = 1$, $C_h = 1$ når $n_i = 2$, $C_h = 3$ når $n_i = 3$ osv.)

Vi ønsker en hashfunktion så $C_h = 0$, hvorved der for alle sæt S_i gælder de er tomme eller kun indeholder ét element.

Vi har pr. definition af C_h følgende, hvor vi stadig lader indikatorvariablen $I(y)$ betegne $[h(x) = h(y)]$:

$$
C_h = \sum_{\substack{\{x,y\}\in S\\ x\neq y}} I(y) \tag{5}
$$

Da vi antager h er universel har vi, at $\mathbb{E}[I(y)] \leq 1/m$ når $x \neq y$:

$$\begin{aligned}
\mathbb{E}[C_h] &= \mathbb{E}\left[\sum_{\substack{\{x,y\}\in S\\ x\neq y}} I(y)\right] \\
&= \sum_{\substack{\{x,y\}\in S\\ x\neq y}} \mathbb{E}[I(y)] && (6)\\
&\leq \binom{n}{2} \cdot \frac{1}{m} && (7)\\
&= \frac{n!}{2!(n-2)!} \cdot \frac{1}{m} \\
&= \frac{1}{2m} \cdot \frac{n!}{(n-2)!} \\
&= \frac{n(n-1)}{2m} && (8)
\end{aligned}$$

I (6) bruger vi linearity of expectation.
I (7) har vi $\binom{n}{2}$ da $n = |S|$ og vi vælger to elementer. Forventningen $1/m$ er givet jf. vores antagelse.
Frem til (8) er det simpel købmandsregning.

Nu kan vi bestemme:

$$\begin{aligned}
\mathbb{P}\left[C_h > \frac{n(n-1)}{m}\right] &= \mathbb{P}[C_h > 2\mathbb{E}[C_h]] \\
&\leq \mathbb{P}[C_h \geq 2\mathbb{E}[C_h]] \\
&\leq \frac{\mathbb{E}[C_h]}{2\mathbb{E}[C_h]} && (9)\\
&= \frac{1}{2}
\end{aligned}$$

Hvor vi i (9) benytter Markovs ulighed.

Hvis vi vælger en ny hashfunktion h hver gang $C_h > n(n-1)/m$ vil det forventede antal gange vi skal gøre det være højest $1/(1/2) = 2$, da det er geometrisk distribueret.

Hvis vi i stedet vælger $m = n^2$ vil det forventede antal hashfunktioner h vi skal prøve før vi får $C_h = 0$ stadig være højest 2. Det ser vi ved at indsætte $m = n^2$, hvorved vi får:

$$C_h \leq \frac{n(n-1)}{n^2} < 1 \qquad \rightarrow \qquad C_h = 0$$

efter forventet højest 2 forsøg. Det gælder da vi kun arbejder med heltal.

Med denne metode får vi altså en dictionary med 0 kollisioner, og derved $O(1)$ opslagstid. Hver eneste forsøg på at finde en passende h tager $O(n+m)$ tid, så det vil også være den forventede køretid.

Til gengæld bruger vi $\Theta(n^2)$ plads.

Bestemmelse af pladsforbrug for 2-level hashing

Lad os nu vælge $m = n$. Nu kan vi finde en hash funktion h så $C_h \leq n$ i forventet højest 2 forsøg (da $C_h \leq \frac{n(n-1)}{n} < n$ jf. hvad vi viste lige før).

Det andet niveau består af en 1-level hash tabel for hvert S_i hvor $|S_i| = n_i \geq 1$. Så for hvert $i \in [n]$ hvor $n_i \geq 1$ lad $h_i : U \to [m_i]$ være en universel hashfunktion hvor vi definerer $m_i = n_i^2$. Hvis der sker nogle kollisioner prøver vi igen med en ny h_i. På samme måde kan vi beregne at vi forventet højest skal bruge 2 forsøg før vi finder en h_i der opfylder dette.

Da er pladsforbruget:

$$\begin{aligned}
O\left(1+n+m+\sum_{i\in[m]}(1+n_i+m_i)\right) &= O\left(n+\sum_{i\in[m]}\left(1+n_i+n_i^2\right)\right) && (10)\\
&= O\left(n+\sum_{i\in[m]}(2n_i+n_i(n_i-1))\right)\\
&= O\left(n+\sum_{i\in[m]}\left(2n_i+2\binom{n_i}{2}\right)\right) && (11)\\
&= O\left(n+2\sum_{i\in[m]}\left(n_i+\binom{n_i}{2}\right)\right)\\
&= O(n+2n+2C_h)\\
&= O(n)
\end{aligned}$$

I (10) benytter vi $n = m \geq 1$ og vores værdi for m_i.
I (11) benytter vi igen $\binom{n}{2} = n(n-1)/2$.

Hver eneste forsøg på at finde en h_i tager $O(n_i + m_i)$ tid, så den forventede tid det tager at finde en passende h_i er også $O(n_i + m_i)$. Det har vi lige bestemt er $O(n)$, så vi får altså en samlet køretid for denne procedure (med 1-level delen også) på $O(n)$.