Hashing

Spørgsmål 7 fra Exam Questions

Kevin Vinther

December 18, 2023

Table of Contents i

Hashing

What is hashing? i

- A big universe $|U| >> m$
- A hash function $h(x) = i$
- Given a value, x, the hash function will calculate an index to an array with size m
- We want to minimize the amount of values hashing to the same value
- However, we know from the pigeonhole principle, even in the best case scenario, if we have $m + 1$ items there must be at least one collision.
- How do we do we get minimum collisions?

Universal Hashing

- Universal Hashing!
- We want to choose a hash function $h : U \to [m]$ randomly and independently of the keys that we wish to hash.
- The result of this is provably good performance on average for all inputs.

Getting good hash functions i

- How do we get a good hash function, and what is a good hash function?
- Ideally, a good hash function would have chance of collision be $1/m$, where m is the size of the table.
- This means that each index has the same chance of being picked.
- For universal hashing, we want a *family* of carefully designed hash functions. Formally:
- Let $\mathcal{H}$ be a finite collection of hash functions such that $h : U \to [m]$ for each $h \in \mathcal{H}$
- Then $\mathcal{H}$ is **universal** if the following holds:

Getting good hash functions ii

- Let $h \in \mathcal{H}$ be randomly chosen. Then $\forall k, l \in U$ with $k \neq l$ $p(h(k) = h(l)) \leq \frac{1}{m}$
- I.e., det jeg sagde før. Hver hash funktion skal jeg sandsynlighed for kollision være mindre end $\frac{1}{m}$. (wtf, har lige lagt mærke til at jeg kun har skrevet på engelsk indtil nu? sorry, det skal jeg nok ændre)
- Husk hashing with chaining:
- For hvert index i tabellen tilknyttes der en linked list, til hvis flere elementer hasher til samme index.

Getting good hash functions iii

Theorem (11.3 Cormen)

Suppose h is chosen randomly from a universal collection $\mathcal{H}$ of hash functions from U to $[m]$. Assume we have used h to hash a set $S \subseteq U$ with $|S| = n$ using chaining to resolve collisions. Let $T = T[0], T[1], \ldots, T[m-1]$ be the table of linked lists we obtain when $T[i]$ is a linked list containing those elements $x \in S$ for which $h(x) = i$.

- Fra dette teorem vil vi vise følgende:
  - Hvis $k \notin S$, så er $E(n_{h(k)}) \leq \frac{n}{m} = \alpha$. Altså, vi regner med at der er $\frac{n}{m}$ elementer i denne linked list, som $h(k)$ hasher til.
  - Hvis $k \in S$, så $E(n_{h(k)}) \leq \alpha + 1$

Getting good hash functions iv

- Til at bevise dette, bruger vi selvfølgelig vores yndlingsværktøj, indicator random variables <3
- $\forall k, l \in U, k \neq l$ define $X_{kl} = \begin{cases} 1 & \text{if } h(k) = h(l) \\ 0 & \text{if } h(k) \neq h(l) \end{cases}$
- Siden $\mathcal{H}$ er universal, gælder det at $p(h(k) = h(l)) \leq \frac{1}{m}$
- For et "fixed" $k \in U$, lad $Y_k = |\{l \in S \setminus \{k\} | h(k) = h(l)\}|$
- Altså, Y_k er antallet af nøgler i $S \setminus \{k\}$ hvilke har den samme hashværdi som k. Dermed har vi:
- $Y_k = \sum_{l \neq k, l \in S} X_{kl}$
- Vi kan dermed bounde Y_k, da vi kender et bound på X_{kl}:

Getting good hash functions v

$$
\begin{aligned}
E(Y_k) &= E(\sum_{l \neq k, l \in S} X_{kl}) \\
&= \sum_{l \neq k, l \in S} E(X_{kl}) \\
&\leq \sum_{l \neq k, l \in S} \frac{1}{m}
\end{aligned}
\tag{1}
$$

- Hvis $k \notin S$, så $n_{h(k)} = Y_k$ og $|\{l \in S | l \neq k\}| = |S| = n$ dermed

$$
E(n_{h(k)}) = E(Y_k) \leq \sum_{l \neq k, l \in S} \frac{1}{m} = \frac{|S|}{m} = \frac{n}{m} = \alpha
$$

Getting good hash functions vi

- Hvis $k \in S$ så $n_{h(k)} = Y_k + 1$ og
  $|\{l \in S | l \neq k\}| = |S| - 1 = n - 1$ dermed

$$E(n_{h(k)} = 1 + E(Y_k) \leq 1 + \sum_{l \neq k, l \in S} \frac{1}{m} = 1 + \frac{n-1}{m} \leq 1 + \alpha$$

Corollary 11.4 i

Corollary (11.4 Cormen)

Using universal hashing and chaining starting from an empty table with m slots, it takes expected time $O(n)$ to handle any sequence of `INSERT`, `SEARCH` and `DELETE` operations when $O(m)$ of them are `INSERT`.

- Bevis:
- Vi indsætter $O(m)$ elementer, så $|S| \in O(m)$, hvilket betyder at $\alpha = \frac{n}{m}$ er $O(1)$
- Dermed er den orventede længde af hver liste i tabellen (table?) $O(1)$, så hver operation tager $O(1)$ forventede tid, så $O(n)$ for alle operationer.

Konstruktion af universal class i

- Vi ved nu at en universal class of hash functions er ret OP. Hvordan laver vi en?
- Cormens metode:
- Vælg et primtal der er større end eller lig med universet: $p \geq |U|$. Antag at $U \leq \{0, 1, 2, 3, \ldots, p-1\}$. Konstruer $\mathcal{Z}_p = \{0, 1, 2, \ldots, p-1\}$ og $\mathcal{Z}_p^* = \{1, 2, \ldots, p-1\}$
- Fordi p er et primtal, kan vi åbenbart løse ligninger modulo p.
- Nu kommer funktionerne:
- For $a \in Z_p^*$ og $b \in Z_p$, definér $h_{ab} = ((ak + b) \mod p) \mod m) h_{ab} : Z_p \rightarrow Z_m$

Konstruktion af universal class ii

- $Z_p \to Z_m$ betyder at den tager fra $\{0, 1, \ldots, p-1\}$ til $\{0, 1, \ldots, m-1\}$.
- Lad dermed: $\mathcal{H} = \mathcal{H}_{pm} = \{h_{ab} | a \in Z_p^*, b \in Z_p\}$

Theorem (11.5 Cormen)

The class $\mathcal{H}$ is universal

- Vi er ligeglade med beviset. Det er ikke en del af pensum.
- Jørgen siger de er svære for ham, så derfor gider vi ikke engang prøve at kigge på dem (rip mig der brugte 2 dage på at forstå dem før jeg så videoen der sagde det ikke var en del af pensum)

Universal Hashing KT i

- Vi vil til gengæld meget gerne se hvordan KT gør det:)
- Vi skal bruge et primtal $p \approx n$, som størrelse af hash tabellen H.
- For at bruge heltals aritmetik når vi designer hash funktionerne, identificerer vi universet med vektorer af formen $x = (x_1, x_2, \ldots, x_r)$. Hvor r er et heltal, og $0 \leq x_i < p$, for hvert i.
- For eksempel kan vi først bestemme U med heltal i sekvensen $[0, N-1]$, og derefter bruge konsekvente blokke af $\lfloor \log p \rfloor$ bits af u til at definere de korresponderende koordinater x_i.
- Hvis $U \subseteq [0, N-1]$, så vil vi bruge et antal af koordinater $r \approx \log N / \log n$

Universal Hashing KT ii

- Lad $\mathcal{A}$ være sættet af alle vektorer af formen $a = (a_1, a_2, \ldots, a_r)$, hvor a_i er et heltal i sekvensen $[0, p-1]$ for hvert $i = 1, \ldots, r$.
- For hvert $a \in A$ definerer vi den lineære funktion:

$$h_a(x) = \left(\sum_{i=1}^{r} a_i x_i \right) \mod p$$

- Dette gennemfører den tilfældige implementation af dictionaries.
- Vi definerer hash funktioner til at være $\mathcal{H} = \{h_a : a \in A\}$. For at eksekvere `MakeDictionary`, vælger vi en tilfædlig hashfunktion fra $\mathcal{H}$

Analyse af datastrukturen i

- Hvis vi bruger hash funktionen beskrevet før, så definerer kollisionen $h_a(x) = h_a(y)$ en lineær ligning modulo primtallet p.

Theorem (13.24)

For any prime p and any integer $z \neq 0 \mod p$, and any two integers α, β if $\alpha z = \beta z \mod p$ then $\alpha = \beta \mod p$

- Bevis:
- Antag at $\alpha z = \beta z \mod p$. Ved at rearrangere leddene får vi $z(\alpha - \beta) = 0 \mod p$, og dermed $z(\alpha - \beta)$ er divisibelt med p. Men $z \neq 0 \mod p$, så z er ikke divisibelt med p.

Analyse af datastrukturen ii

- Siden p er et primtal, følger det at $\alpha - \beta$ må være divisibelt af p, som er hvad teoremet stater.
- Vi vil nu bevise hovedresultatet i vores analyse.

Theorem (13.25)

The class of linear functions $\mathcal{H}$ deifned above is universal.

- **Bevis:**
- Lad $x = (x_1, x_2, \ldots, x_r)$ og $y = (y_1, y_2, \ldots y_r)$ være to distinkte elemneter af U.
- Vi må vise at sandsynligheden af $h_a(x) = h_a(y)$, for et tilfældigt valgt $a \in A$ er højest $\frac{1}{p}$.
- Siden $x \neq y$, så må der være et index j, således at $x_j \neq y_j$.

Analyse af datastrukturen iii

- Vi vil nu finde en måde at vælge en tilfældig vektor $a \in A$ på.
- Først vælger vi alle koordinatoerne a_i, hvor $i \neq j$. Så, til sidst, vælger vi koordinaten a_j.
- Vi vil viser at uanset hvordan alle de andre koordinator a_i var valgt, er sandsynligheden for at $h_a(x) = h_a(y)$ taget over det endelige valg af a_j, er præcis $\frac{1}{p}$.
- Det følger at $p(h_a = h_a(y)) = 1/p$ ligeså.
- Konklusionen er intuitivt klar: sandsynligheden er $\frac{1}{p}$ uanset hvordan vi vælger de andre a_i, så er den $\frac{1}{p}$ generelt.
- Der er også et direkte bevis af dette hvor der bliver brugt conditional probability.

Analyse af datastrukturen iv

- Lad ε være hændelsen at $h_a(x) = h_a(y)$, og lad $\mathcal{F}_b$ være hændelsen at alle koordinater a_i (hvor $i \neq j$) får en sekvens af værdier b.
- Vi vil vise at $P(\varepsilon|\mathcal{F}_b) = \frac{1}{p}$ for alle b.
- Det følger at $P(\varepsilon) = \sum_b P(\varepsilon|\mathcal{F}_b) \cdot p(\mathcal{F}_b) = (1/p) \sum_b = 1/p$
- For at konkludere beviset, antager vi at værdierne er blevet valgt arbitrært for alle andre koordinator a_i, og at vi ser sandsynligheden af at vælge a_j således at $h_a(x) = h_a(y)$. Ved at rearrangere leddene, ser vi at $h_a(x) = h_a(y)$ hvis og kun hvis

$$a_j(y_j - x_j) = \sum_{i \neq j} a_i(x_i - y_i) \mod p$$

Analyse af datastrukturen v

- Siden valget for alle a_i $(i \neq j)$ er fikset, kan vi se højrehåndssiden som værende en fixed kvantitet m.
- Derudover, lad os definere $z = y_j - x_j$.
- Nu er det nok at vise at der er præcis en værdi $0 \leq a_j < p$ som satisfier $a_j z = m \mod p$

Perfect Hashing i

- Hashing har en fantastisk average-case performance. Kan vi også få en fantastisk worst-case?
- Ja! Så længe nøglerne er statiske.
- Vi siger at en hashing teknik er **perfect hashing** hvis $O(1)$ hukommelsesadgang skal bruges til at lave search i worst case.

Perfect Hashing ii

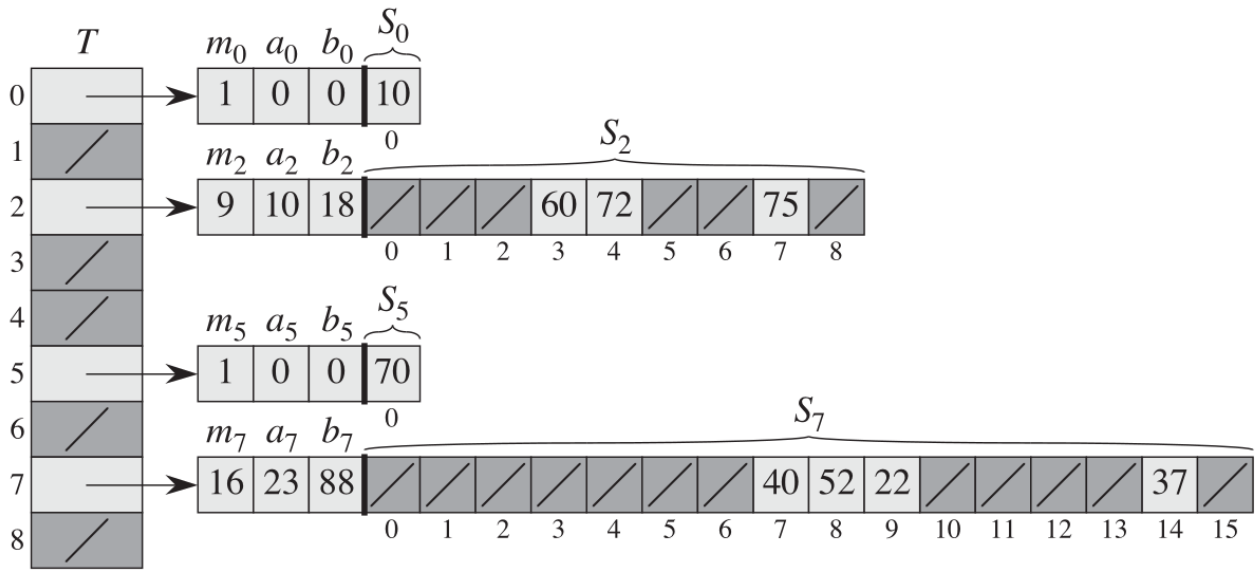

Figure 11.6 Using perfect hashing to store the set $K = \{10, 22, 37, 40, 52, 60, 70, 72, 75\}$. The outer hash function is $h(k) = ((ak + b) \bmod p) \bmod m$, where $a = 3$, $b = 42$, $p = 101$, and $m = 9$. For example, $h(75) = 2$, and so key 75 hashes to slot 2 of table T. A secondary hash table S_j stores all keys hashing to slot j. The size of hash table S_j is $m_j = n_j^2$, and the associated hash function is $h_j(k) = ((a_j k + b_j) \bmod p) \bmod m_j$. Since $h_2(75) = 7$, key 75 is stored in slot 7 of secondary hash table S_2. No collisions occur in any of the secondary hash tables, and so searching takes constant time in the worst case.

Perfect Hashing iii

- Det første niveau er det samme som ved hashing med chaining.
- I stedet for at bruge en linked list, bruger vi et andet, mindre hash table S_j med en assiceret hash funktion h_j.
- Ved at vælge hash funktionerne h_j nøje, kan vi garantere at der ikke er nogen kolissioner på det andet niveau.
- For at garanterer at der ikke er nogen kollisioner på det andet niveau, skal vi lade størrelsen m_j af hash tabellen $S - j$ være lig med n_j^2

Perfect Hashing iv

Theorem (11.9)

Suppose that we store n keys in a hash table of size $m = n^2$ *using a hash function h ranodmly chosen from a universal class of hash functions. Then, the probability is less than* $1/2$ *that there are any collisions.*

- **Bevis**:
- Der er $\binom{n}{2}$ par af nøgler der kan kollidere.
- Hver par kolliderer med sandsynlighed $1/m$ (da det er en universal hash function).
- Lad X være en random variable der tæller antallet af kollisioner. Når $m = n^2$, er det forventede antal af kollisioner:

$$
\begin{aligned}
E[X] &= \binom{n}{2} \cdot \frac{1}{n^2} \\
&= \frac{n^2 - n}{2} \cdot \frac{1}{n^2} \\
&< 1/2
\end{aligned}
\tag{2}
$$

Theorem (11.10)

Suppose that we store n keys in a hash table of size $m = n$ using a hash function h randomly chosen from a universal class of hash functions. Then, we have

$$E\left[\sum_{j=0}^{m-1} n_j^2\right] < 2n$$

where n_j is the number of keys hashing to slot j

- **Bevis:**

Perfect Hashing vii

- Vi starter med den følgende identitet, hvilket holder for enhver ikke negativ heltal a:

$$a^2 = a + 2\binom{a}{2}$$

Count-min sketch i

- Lad S være en lang, måske uendelig, strøm af data. Vores mål er at estimere frekvenserne af elementer som forekommer ofte i S.
- Lad b, l være heltal som vi bestemmer sebnere.
- Lad $\mathcal{H}$ være en universal familie af hash funktione fra universet U som indeholder alle mulige elementer som kan være i strømmen til sættet af heltal $\{1, 2, \ldots, b\}$ og lad $h_1, h_2, \ldots, h_l$ være distincte medlemmer fra $\mathcal{H}$ valgt tilfældigt.
- Under dette siger vi at funktionerne h_i er **universelle** hvis de er tilfældigt valgt fra en universel hash familie $\mathcal{H}$.
- Ved brug af disse hash funktione bygger vi en $l \times b$ array M af tællere.

Count-min sketch ii

- Til at starte med gælder $M_{i,j} = 0$ for all $i \in [l], j \in [b]$
- Vi forarbejder strømmen som den kommer som følgende med det nuværende element x fra S:
  - For hvert $i \in [l]$ øger vi tælleren $M_{i,h_i(x)}$ med en.
  - Så hvert nyt element af S øger værdien af præcis l entries (?) i tabellen.
- Antag nu at vi har forarbejdet de første n elementer af strømmen.
- Vi kalder den ordnede sekvens af de første n elementer i S for S_n, e.g., $S = \{A, B, C, D, A, B, C, D, \ldots\}$, så er $S_5 = \{A, B, C, D, A\}$

Count-min sketch iii

- Hvad kan vi sige om frekvenserne, f_x af disse elementer x som forekommer mindst en gang i S_n baseret p åudelukkende vores array af tællere?
- Lad x være et fixed element der forekommer i S_n, og lad os først se hvad tælleren $M_{i,h_i(x)}$ faktisk tæller.
- For vores egens sindstilstands skyld betegner vi $M_{i,h_i(x)}$ som $Z_{i,x}$. Læg mærke til at dette er et random variable, fordi h_i er valgt tilfældigt fra $\mathcal{H}$.
- Lad indicator random variablen $I_{i,x}$ være defineret som følger (på elementerne af S_n).

$$I_{i,x} = \begin{cases} 1 & \text{if } h_i(x) = h_i(y) \\ 0 & \text{otherwise} \end{cases}$$

Count-min sketch iv

- Vi kan nu udtrykke $Z_{i,x}$ som følgende:

$$Z_{i,x} = f_x + \sum_{\{y \in S_n | y \neq x\}} f_y \cdot I_{i,x}(y)$$

- Det følger at $Z_{i,x}$ er en upper bound på f_x.
- Lad os bestemme det forventede værdi af $Z_{i,x}$.
- For dette bruger vi at $p(I_{i,x}(y) = 1) \leq \frac{1}{b}$, da h_i er universal.
- Vi bruger også at summen af frekvenserne af alle elementer forekommer mindst en gang i S_n er n.

Count-min sketch v

$$
\begin{aligned}
E[Z_{i,x}] &= E[f_x + \sum_{\{y \in S_n | y \neq x\}} f_y \cdot I_{i,x}(y)] \\
&= E[f_x] + E[\sum_{\{y \in S_n | y \neq x\}} f_y \cdot I_{i,x}(y)] \\
&= f_x + \sum_{\{y \in S_n | y \neq x\}} f_y \cdot E[I_{i,x}(y)] \\
&\leq f_x + \sum_{\{y \in S_n | y \neq x\}} f_y \cdot \frac{1}{b} \\
&= f_x + \frac{1}{b} \sum_{\{y \in S_n | y \neq x\}} f_y \\
&\leq f_x + \frac{n}{b}
\end{aligned}
$$

Count-min sketch vi

- Så i forventningen er tælleren $Z_{i,z}$ for f_x "off" med højest $\frac{n}{b}$. Siden n kan være kæmpe, og vi kun bruger et fixed sæt af b tællere, så kan vi ikke forvente et bedre estimat end nogen der afhænger af n.
- Husk at Markov's Inequality implier at sandsynligheden for en random variable er mindst dobbelt sin forventede er højest $1/2$.
- Dermed:

$$p(Z_{i,x} - f_x \geq \frac{2n}{b}) \leq 1/2$$

.

Count-min sketch vii

- Dette holder for alle værdier $i \in [l]$, så hvis vi sætter $\hat{f}_x = \min_{i \in [l]} Z_{i,x}$ får vi en upper bound på f_x, og siden hash funktionerne $h_1, h_2, \ldots, h_l$ er uafhængige af hinanden får vi at:

$$p(\hat{f_x}] - f_x > \frac{2n}{b}) \leq (1/2)^l$$

- Antag nu at vi er givet værdierne ϵ og δ, hvor vi vil finde værdien at vores estimat $\hat{f}_x$ er "off" med mere end ϵn er højest δ.
- Ved brug af udregningerne fra før, kan vi udregne brugbare værdier af b og l baseret på ϵ og δ. Det følger at hvis vi tager $b = \frac{2}{\epsilon}$ og $l = \log_2\left(\frac{1}{\delta}\right)$, så

$$p(\hat{f}_x - f_x \geq \epsilon n) \leq \delta$$

Count-min sketch viii

- Dermed, ved brug af $b \cdot l = \frac{2}{\epsilon} \cdot \log_2 \left(\frac{1}{\delta}\right)$ tællere (størrelsen af arrayet M) kan vi få den ønskede akkurathed uafhængigt af n, hvilket kunne være kæmpe.