CI2612: Algoritmos y Estructuras de Datos II

Blai Bonet

Universidad Simón Bolívar, Caracas, Venezuela

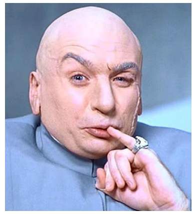
Hashing universal

Objetivos

- Hashing universal
- Hashing perfecto

Introducción

Si un **adversario malévolo** escoge las claves a ser insertadas en una tabla de hash (con función de hash conocida), el adversario puede seleccionar n claves que mapeen al mismo slot y **forzar un desempeño de peor caso**



http://i.kinja-img.com

Una forma de contrarestar esta situación es elegir la función de hash de forma **aleatoria e independiente** a las claves a insertar

Esta técnica, llamada **hashing universal**, puede garantizar buen desempeño esperado sin importar la claves usadas por el adversario

Hashing universal

Al inicio de la ejecución, hashing universal elige una función de hash de **forma aleatoria** de un conjunto $\mathcal{H}$ de **funciones disponibles,** y la utiliza para realizar todas las operaciones sobre la tabla

La randomización garantiza que con gran probabilidad ninguna secuencia de operaciones resulte en un comportamiento peor caso

Como en `RandomizedQuicksort`, el algoritmo puede comportarse de forma diferente en múltiples llamadas sobre la misma entrada

Clases universales de funciones de hash

Considere una clase $\mathcal{H}$ de funciones de hash $h : U \to \{0, \ldots, m-1\}$ (m es el número de slots)

Decimos que $\mathcal{H}$ es una **clase universal de funciones de hash** si para cada par de claves **distintas** $k, \ell \in U$, el número de funciones $h \in \mathcal{H}$ tal que $h(k) = h(\ell)$ es a lo sumo $|\mathcal{H}|/m$

I.e. $\mathcal{H}$ es una clase universal de funciones de hash cuando

$$\max_{k,\ell \in U, k \neq \ell} |\{h \in \mathcal{H} : h(k) = h(\ell)\}| \;\leq\; |\mathcal{H}|/m$$

I.e. $\mathcal{H}$ es universal si para todo par de **claves distintas** k y ℓ, la probabilidad de que una función $h \in \mathcal{H}$ seleccionada **uniformemente al azar** genere una colisión es $\mathbb{P}(h(k) = h(\ell)) \leq 1/m$

Análisis de hashing universal

Considere la selección aleatoria de $h \in \mathcal{H}$ y defina la v.a. indicadora $X_{k\ell} = \mathbb{I}\{h(k) = h(\ell)\}$ para dos claves $k, \ell \in U$

Tomando esperanza con respecto a la elección de h y recordando la definición de $\mathcal{H}$, $\mathbb{E}[X_{k\ell}] = \mathbb{P}(h(k) = h(\ell)) \leq 1/m$

Defina $Y_k = \sum_{\ell \in T, \ell \neq k} X_{k\ell}$ igual al número de claves en la tabla T que son distintas a k y que son mapeadas por h al **mismo slot** de k

La esperanza de Y_k es

$$\mathbb{E}[Y_k] \;=\; \textstyle\sum_{\ell \in T, \ell \neq k} \mathbb{E}[X_{k\ell}] \;\leq\; \sum_{\ell \in T, \ell \neq k} \frac{1}{m}$$

Análisis de hashing universal

Utilizamos h para insertar elementos en una tabla T con **resolución de colisiones por encadenamiento**

Consideramos dos casos al buscar un elemento x con clave k en una tabla T con n elementos:

Caso 1: $k \notin T$. Entonces, toda la lista en el slot $h(k)$ es revisada. El número de elementos en dicha lista es $n_{h(k)} = Y_k$

Por otro lado, $|\{\ell \in U : \ell \in T \wedge \ell \neq k\}| = n$ ya que $k \notin T$

Entonces,

$$\mathbb{E}[n_{h(k)}] \;=\; \mathbb{E}[Y_k] \;\leq\; \textstyle\sum_{\ell \in T, \ell \neq k} \frac{1}{m} \;=\; \frac{n}{m} \;=\; \alpha$$

Análisis de hashing universal

Utilizamos h para insertar elementos en una tabla T con **resolución de colisiones por encadenamiento**

Consideramos dos casos al buscar un elemento x con clave k en una tabla T con n elementos:

Caso 2: $k \in T$. Como Y_k no cuenta la clave k, $n_{h(k)} = 1 + Y_k$

Por otro lado, $|\{\ell \in U : \ell \in T \wedge \ell \neq k\}| = n - 1$ ya que $k \in T$

Entonces,

$$\mathbb{E}[n_{h(k)}] \;=\; 1 + \mathbb{E}[Y_k] \;\leq\; 1 + \sum_{\ell \in T, \ell \neq k} \tfrac{1}{m} \;=\; 1 + \tfrac{n-1}{m} \;<\; 1 + \alpha$$

Análisis de hashing universal

Utilizamos h para insertar elementos en una tabla T con **resolución de colisiones por encadenamiento**

Consideramos dos casos al buscar un elemento x con clave k en una tabla T con n elementos:

Conclusión: si $\mathcal{H}$ es universal, las búsquedas toman tiempo $O(1 + \alpha)$

Garantía provista por hashing universal

Considere un esquema de hashing universal donde las colisiones son resueltas por encadenamiento

Considere una **secuencia cualquiera de n operaciones** de tipo diccionario en una tabla de hash con m **slots** inicialmente vacía, donde se realizan $O(m)$ inserciones

Veamos que el tiempo esperado para ejecutar las operaciones es $\Theta(n)$

Como existen $O(m)$ inserciones, $\alpha = O(1)$. Las inserciones y eliminaciones toman tiempo constante. Las búsquedas requieren tiempo esperado $O(1 + \alpha) = O(1)$

Garantía: las n operaraciones requieren tiempo esperado $\Theta(n)$

Garantía provista por hashing universal

Garantía: las n operaraciones requieren tiempo esperado $\Theta(n)$

El tiempo esperado es con respecto a la elección al azar de la función de hash

La garantía no depende de la secuencia de operaciones realizada

I.e. no existe una secuencia de operaciones que resulte en desempeño de peor caso

Clases universales de funciones de hash

Hashing universal depende de la existencia de clases universales de funciones

Mostraremos como construir una clase universal de funciones de hash utilizando dos métodos distintos:

- método matricial
- método algebraico

Clase universal: método matricial

Consideramos el caso cuando el número de slots en la tabla es $m = 2^b$ (potencia de 2) y las claves son de u bits (i.e. $\in \{0, \ldots, 2^{1+u} - 1\}$)

Sea M una matriz de 0/1 de dimensión $b \times u$. Definimos la función de hash $h_M(k) = M \times k$ (módulo 2), donde k es interpretada como un **vector columna** de dimensión $u \times 1$ y el valor $h(k)$ es un **vector columna** de dimension $b \times u$

E.g. para $b = 3$ y $u = 4$, considere $k = 10 = 1010_b$

$$M = \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 \end{bmatrix} \qquad \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 \end{bmatrix} \times \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix}$$

entonces $h_M(k) = h_M(1010_b) = 110_b = 6$

Clase universal: método matricial

Considere la clase $\mathcal{H} = \{h_M : M$ es matriz 0/1 de dimensión $b \times u\}$

Veamos que para **claves distintas** $x \neq y$: $\mathbb{P}(h(x) = h(y)) = \frac{1}{m} = \frac{1}{2^b}$

Multiplicar $M \times k$ es equivalente a **sumar columnas seleccionadas** de M (módulo 2); e.g. para

$$M = \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 \end{bmatrix} \qquad \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 \end{bmatrix} \times \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix}$$

la primera y tercera columna de M se suman módulo 2 y resultan en el lado derecho de la igualdad

Clase universal: método matricial

Considere la clase $\mathcal{H} = \{h_M : M$ es matriz 0/1 de dimensión $b \times u\}$

Veamos que para **claves distintas** $x \neq y$: $\mathbb{P}(h(x) = h(y)) = \frac{1}{m} = \frac{1}{2^b}$

Considere ahora $x \neq y$ y suponga que difieren en el i-ésimo bit, en particular, $x_i = 0$ y $y_i = 1$

La elección al azar de $h = h_M$ se puede pensar como la **construcción aleatoria** de M. En particular, suponga que hemos construido M completamente excepto la i-ésima columna:

- como $x_i = 0$, el valor $h(x)$ no cambia al variar dicha columna
- como $y_i = 1$, el valor $h(y)$ cambia con cada valor distinto de la i-ésima columna: si el j-ésimo bit de la columna cambia (flips), el j-ésimo bit de $h(y)$ también cambia (flips)

Clase universal: método matricial

Considere la clase $\mathcal{H} = \{h_M : M$ es matriz 0/1 de dimensión $b \times u\}$

Veamos que para **claves distintas** $x \neq y$: $\mathbb{P}(h(x) = h(y)) = \frac{1}{m} = \frac{1}{2^b}$

E.g. considere $x = 1010_b$, $y = 0011_b$ y la **matriz incompleta** M:

$$M = \begin{bmatrix} 1 & 0 & 0 & a \\ 0 & 1 & 1 & b \\ 1 & 0 & 1 & c \end{bmatrix} \qquad \begin{bmatrix} 1 & 0 & 0 & a \\ 0 & 1 & 1 & b \\ 1 & 0 & 1 & c \end{bmatrix} \times \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix}$$

Por otro lado, $h_M(y) = [0 + a \;\; 1 + b \;\; 1 + c]^t$

Entonces, $h_M(x) = h_M(y)$ si y solo si $a = 1$, $b = 0$ y $c = 1$

Clase universal: método matricial

Considere la clase $\mathcal{H} = \{h_M : M$ es matriz 0/1 de dimensión $b \times u\}$

Veamos que para **claves distintas** $x \neq y$: $\mathbb{P}(h(x) = h(y)) = \frac{1}{m} = \frac{1}{2^b}$

Un solo valor para la i-ésima columna hace $h(x) = h(y)$

Existen $m = 2^b$ distintas i-ésimas columnas

$$\mathbb{P}(h(x) = h(y)) \;=\; \frac{1}{m} \;=\; \frac{1}{2^b}$$

Por lo tanto, $\mathcal{H}$ es una clase universal de funciones de hash

Clase universal: método algebraico

Primero escogemos un **número primo** p tal que todas las claves $k \in U$ son menores a p y $p > m$ (m es número de slots en T)

Definimos $\mathbb{Z}_p = \{0, 1, 2, \ldots, p-1\}$ y $\mathbb{Z}_p^* = \{1, 2, \ldots, p-1\}$

Para $a \in \mathbb{Z}_p^*$ y $b \in \mathbb{Z}_p$, definimos la función de hash:

$$h_{ab}(k) \;=\; ((ak + b) \bmod p) \bmod m$$

y la clase de funciones de hash $\mathcal{H}_{pm} = \{h_{ab} : a \in \mathbb{Z}_p^*$ y $b \in \mathbb{Z}_p\}$

Mostraremos que $\mathcal{H}_{pm}$ es una clase universal de funciones de hash

La clase $\mathcal{H}_{pm}$ es universal

Considere dos claves distintas k y ℓ en $\{0, 1, \ldots, p-1\}$, y

$$r \;=\; (ak + b) \bmod p \qquad\qquad s \;=\; (a\ell + b) \bmod p$$

Observaciones:

- Como $k \neq \ell$ (y $\mathbb{Z}_p$ es un campo), entonces $r \neq s$
- Cada uno de los $p(p-1)$ pares (a, b), con $a \neq 0$, genera un **par distinto** (r, s) con $0 \leq r, s < p$ y $r \neq s$ (existen $p(p-1)$ pares)
- Para claves k y ℓ fijas, si seleccionamos un par (a, b) al azar de forma uniforme, el par (r, s) resultante es cualquiera de los $p(p-1)$ pares con igual probabilidad

La clase $\mathcal{H}_{pm}$ es universal

La probabilidad de que las claves k y ℓ colisionen es igual a la probabilidad que $r \bmod m = s \bmod m$ (denotado $r \equiv s \bmod m$) cuando r y s son **escogidos de forma uniforme** en $\{0, 1, \ldots, p-1\}$ con $r \neq s$

Para r fijo, de los $p-1$ posibles valores para s (distintos de r) solo existen a lo sumo $\lceil p/m \rceil - 1$ valores tales que $r \equiv s \bmod m$

Defina $x = r \bmod m$. Los y congruentes con x mod m están entre

$$x,\ x+m,\ x+2m,\ x+3m,\ \ldots,\ x+\left(\left\lceil \frac{p}{m} \right\rceil - 1\right)m$$

Existen a lo sumo $\lceil p/m \rceil - 1$ de ellos distintos a $x = r \mod m$

La clase $\mathcal{H}_{pm}$ es universal

La probabilidad de que las claves k y ℓ colisionen es igual a la probabilidad que $r \bmod m = s \bmod m$ (denotado $r \equiv s \bmod m$) cuando r y s son **escogidos de forma uniforme** en $\{0, 1, \ldots, p-1\}$ con $r \neq s$

Para r fijo, de los $p-1$ posibles valores para s (distintos de r) solo existen a lo sumo $\lceil p/m \rceil - 1$ valores tales que $r \equiv s \bmod m$

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

La probabilidad que k colisione con ℓ es igual

$$\mathbb{P}(h(k) = h(\ell)) \ =\ \mathbb{P}(r \equiv s \mod m) \ \leq\ \frac{(p-1)/m}{p-1} \ =\ \frac{1}{m}$$

Hashing perfecto

Hashing perfecto

Hashing puede usarse para obtener **desempeño perfecto** cuando el conjunto de claves es **estático**; i.e. el conjunto de claves no cambia

En aplicaciones diversas, la tabla es **pre-cargada** con las claves y luego usada para determinar si una clave dada pertenece o no a K (el conjunto de n claves estáticas)

Un esquema de hashing es **perfecto** cuando las operaciones de búsqueda toman **tiempo constante en el peor caso**

Obtendremos un esquema de hashing perfecto utilizando hashing universal

Idea para obtener hashing perfecto

Primero veamos que una clase universal $\mathcal{H}$ de funciones de hash para un tabla con $m = n^2$ slots es un hash perfecto con **gran probabilidad**

De hecho, existen $\binom{n}{2}$ pares de claves distintas. Al elejir de forma aleatoria una función $h \in \mathcal{H}$, la probabilidad de obtener una colisión entre dos claves distintas cualesquiera es $\leq 1/m$ (def. hashing univ.)

El valor esperado del número X de colisiones está acotado por

$$\mathbb{E}[X] \;=\; \sum_{k,\ell \in K, k \neq \ell} \mathbb{E}[X_{k\ell}] \;\leq\; \binom{n}{2}\frac{1}{m} \;=\; \frac{n(n-1)}{2m} \;\leq\; \frac{1}{2}$$

donde $X_{k\ell}$ es la v.a. que indica una colisión bajo h entre k y ℓ

Por la desigualdad de Markov $\mathbb{P}(X \geq t) \leq \mathbb{E}[X]/t$:

$$\mathbb{P}(X \geq 1) \;\leq\; \mathbb{E}[X] \;\leq\; 1/2$$

Esquema de 2 niveles para hashing perfecto

Utilizando la idea anterior, elegimos funciones $h \in \mathcal{H}$ repetidamente hasta encontrar una función h^* que genere cero colisiones

El problema de esta idea es que require de espacio cuadrático en n que puede ser excesivo

Veremos como un esquema de **hashing de 2 niveles** resuelve el problema de espacio (ambos niveles con hashing universal)

La idea es tener un hashing de primer nivel con n slots donde las colisiones dentro de cada slot se resuelven utilizando otro esquema de hashing de segundo nivel en lugar de encadenamiento

Esquema de 2 niveles para hashing perfecto

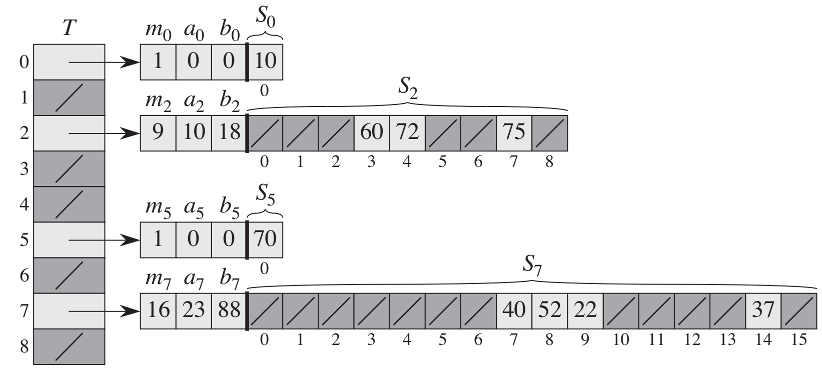


Imagen de Cormen et al. Intro. to Algorithms. MIT Press

$K = \{10, 22, 37, 40, 52, 60, 70, 72, 75\}$, $p = 101$ y $h = h_{ab}$ con $a = 3$ y $b = 42$

Algoritmo para hashing perfecto

1. Elegir primo p mayor a todas las claves y $m = n$
2. Elegir una función $h_{ab} \in \mathcal{H}_{pm}$ uniformemente al azar
3. Se pasan todas las claves por h_{ab} para calcular el número de colisiones por slot: $n_0, n_1, \ldots, n_{m-1}$
4. Repetir 3–4 hasta que $\sum_{j=0}^{m-1} n_j^2 \leq 4n$
5. Para $j = 0, 1, \ldots, m-1$ hacer
6. Elegir una función $h_j = h_{a_j b_j} \in \mathcal{H}_{pm_j}$ al azar donde $m_j = n_j^2$
7. Calcular el número de colisiones generadas por h_j para las n_j claves en el j-ésimo slot
8. Repetir 5–6 hasta que no existan colisiones para h_j

Tiempo total esperado (ejercicios) es $\Theta(n)$; espacio total es $\Theta(n)$

2 niveles de hashing universal

Para el **primer nivel**, escogemos un primo p mayor a todas las claves en K, $m = n$ y una función de hash al azar en la clase universal $\mathcal{H}_{pm}$

En el **segundo nivel**, para cada slot j con n_j elementos, elegimos repetidamente una función de hash $h_j = h_{a_j b_j}$ al azar en la clase universal $\mathcal{H}_{pm_j}$ donde $m_j = n_j^2$ hasta encontrar una que no genere colisiones (cuando $n_j = 1$, escogemos $a_j = b_j = 0$ directamente)

Como ya vimos, la función h_j la podemos encontrar en muy pocas iteraciones

Por construcción, tenemos un esquema de hashing perfecto. Veamos que la cantidad de espacio requerida tiene valor esperado $\Theta(n)$

Análisis de espacio requerido para hashing perfecto

Se necesitan n slots en la tabla de primer nivel y $\sum_{j=0}^{n-1} n_j^2$ slots para todas las tablas de segundo nivel

$$\begin{aligned}
\mathbb{E}\left[\sum_{j=0}^{n-1} n_j^2\right] &= \mathbb{E}\left[\sum_{j=0}^{n-1} n_j + 2\binom{n_j}{2}\right] = n + 2\mathbb{E}\left[\sum_{j=0}^{n-1}\binom{n_j}{2}\right] \\
&= n + 2\mathbb{E}\left[\sum_{k,\ell\in K, k\neq\ell} X_{k\ell}\right] \leq n + 2\binom{n}{2}\frac{1}{m} \\
&= n + \frac{n(n-1)}{m} = 2n - 1 < 2n
\end{aligned}$$

Por la desigualdad de Markov: con al menos $1/2$ de probabilidad el espacio usado en el segundo nivel es a lo sumo $4n$:

$$\mathbb{P}\left(\textstyle\sum_{j=0}^{n-1} n_j^2 \geq 4n\right) \leq 2n/4n = 1/2$$

Resumen

- Se pueden definir esquemas de hash que sean robustos ante adversarios y esquemas de hashing perfecto
- Hashing universal se fundamenta en la existencia de clases universales de funciones de hash
- Dos clases universales de funciones: método matricial y método algebraico
- Hashing perfecto se logra implementando un esquema de hashing universal de dos niveles

Ejercicios (1 de 2)

1. Muestre $\lceil\frac{p}{q}\rceil \leq \frac{p+q-1}{q} < \frac{p}{q} + 1$ para **enteros** $p, q > 0$
2. Implemente esquemas de hashing universal que utilice: una clase universal de funciones de hash usando el método matricial, y una clase de universal de funciones de hash usando el método algebraico
3. Evalue ambos esquemas sobre sequencias de operaciones de longitud m
4. Construya un hash perfecto para $K = \{23, 67, 12, 7, 75, 35, 42, 44, 45\}$
5. Calcule el número esperado de iteraciones que tiene que realizar un algoritmo que busca una función de hash perfecta (sin colisiones) para un conjunto K de n claves con una tabla de tamaño $m = n^2$ cuando se selecciona al azar una función candidata $h \in \mathcal{H}$, donde $\mathcal{H}$ es una clase universal de funciones de hash (asuma que evaluar la función de hash toma tiempo constante)

Ejercicios (2 de 2)

6. Calcule el número esperado de iteraciones que toma encontrar un función de hash $h = h_{ab}$ para hashing perfecto tal que $\sum_{j=0}^{m-1} n_j^2 \leq 4n$

7. Acote el tiempo esperado para construir un hash perfecto para n claves dadas usando hashing universal de 2 niveles

8. Implemente un programa que reciba un conjunto K de claves y genere un hashing perfecto de 2 niveles para K

9. Evalue su implementación de hashing perfecto sobre varios conjuntos K de claves y estime el desperdicio de espacio en función de $n = |K|$