Diccionarios randomizados

Teoría de Algoritmos I (75.29 / 95.06)

Ing. Víctor Daniel Podberezski
vpodberezski@fi.uba.ar

Diccionarios

Existe

Un universo U de posibles elementos extremadamente grande

Queremos una estructura para

manipular un subconjunto S de U de tamaño apreciablemente menor.

Definiremos

en n elementos de S como la cantidad máxima a manipular.

Deseamos

almacenarlos, recuperarlos, y/o eliminarlos en tiempo constante por operación

Funciones de Hashing

Sea

H un Hash Table, vector de n posiciones.

la función $h: U \rightarrow \{0,1,\ldots n-1\}$ mapea un elemento de U a una posición

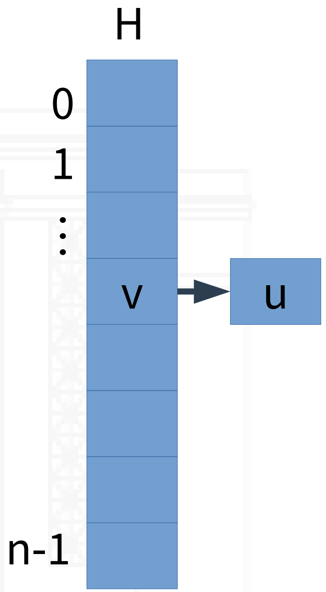
Cada elemento $u \in U$ a almacenar

se almacena en la posición de H que indica $h(u)$

Sinónimos y colisiones

Se conoce como sinónimos

a $u,v \in U$ / $h(v) = h(u)$



Si 2 o más sinónimos se intentan almacenar en H

Se produce una colisión: se deben almacenar en el mismo lugar.

La estructura, debe lidiar con las colisiones

Existen diferentes maneras de hacerlo

Por ejemplo: almacenar cada sinónimo como una lista enlazada

Tiempos de acceso

Si H no tiene almacenado sinónimos

El tiempo por operación es calcular h(u) ← O(1)

Si H tiene colisiones

El tiempo por operación es h(u) + recorrer la lista de sinónimos en la posición H(h(u))

Queremos que

la función de hashing distribuya lo mejor posible los elementos en el intervalo de H

Evitando que la tabla H contenga muchos sinónimos

Una buena función de hashing

Queremos

minimizar la probabilidad de colisiones.

Se suelen utilizar

funciones del estilo h(u): u mod p,   con p primos.

Funcionan bien empíricamente y para la mayoría de los subconjuntos de elementos

Deseamos que

que podamos probar su eficiencia con alta probabilidad

Una buena función de hashing (cont.)

Nos atrae

la idea de distribuir uniformemente y para eso utilizar una función aleatoria

No podemos incluir lo probabilístico en las posiciones,

sino no podemos asegurar volver a encontrar un elemento almacenado.

Queremos que la posibilidad de una colisión

Sea probabílisticamente pequeña

Clase universal de funciones de Hashing

Una familia de funciones de hashing $\mathcal{H}$

debe cumplir 2 condiciones

Cada h de $\mathcal{H}$

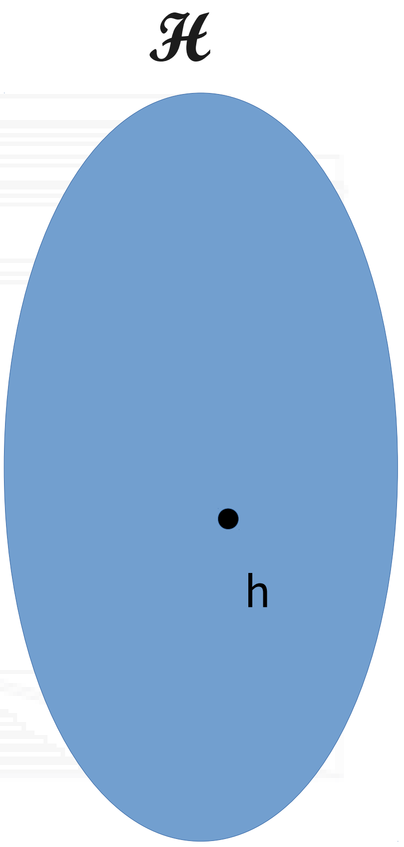
se debe representar de manera compacta y

se debe calcular de forma eficiente.

Para cualquier u,v $\in$ U

la probabilidad, seleccionando h de la familia $\mathcal{H}$ al azar,

De que h(u) = h(v) es a lo sumo 1/n



Cantidad de colisiones esperadas

Sea

$\mathcal{H}$ una clase de funciones de hashing universales que mapea el universo U al set {0, 1, …, n-1 },

S un subset arbitrario de U de tamaño máximo n

u $\in$U (cualquier elemento)

Definimos

la variable aleatoria X como la cantidad de elementos s $\in$S para los que h(s) = h(u) para una selección aleatoria de h $\in\mathcal{H}$

La variable aleatoria X_s para un elemento aleatorio s $\in$S, igual a 1 si h(s)=h(u), sino igual a 0

Cantidad de colisiones esperadas (cont.)

Como h ∈ $\mathcal{H}$

$E[X_s] = Pr[X_s = 1] \leq 1/n$

Entonces

$$E[X] = \sum_{s \in S} E[X_s] \leq |S| \frac{1}{n} \leq 1$$

La cantidad de elementos esperados que colisionan con u de S

es contante (no importa el tamaño de n) y es como mucho 1.

Diseño de una C. de F. U. de H.

Usaremos

un número p primo ≈ n como el tamaño de la tabla de Hash H.

Representaremos a cada elemento u ∈ U

como un vector $X = (x_1, x_2, \ldots, x_r)$ con un r fijo, $0 \leq x_i < p$ para cada i

Sea A el set de todos los vectores de la forma:

$a=(a_1, a_2, \ldots, a_r)$ con $0 \leq a_i < p$ para cada i

Definimos la función lineal $h_a(x) = (\sum_{i=1}^{r} a_i . x_i) \, mod \; p$

Diseño de una C. de F. U. de H. (cont.)

Definiremos la familia de funciones de hashing

$$H = \{ h_a / a \in A \}$$

Para construir el diccionario

Se selecciona un número primo $p \geq n$

Se genera aleatoriamente uniforme un vector $a \in A$

Con esta se define h_a

Para almacenar la función de hashing

Solo hace falta almacenar el vector a seleccionado ← es compacto

… solo falta verificar su probabilidad de generar colisiones

Una propiedad previa necesaria...

Sea

p primo

$z \neq 0 \bmod p$ (z no es divisible por p).

Entonces

$a z = m \bmod p$

Tiene como mucho una solución con $0 \leq a < p$

Es nuestra propuesta una C.F.H.U?

Sean $x=(x_1,x_2,\ldots x_r)$, $y=(y_1,y_2,\ldots y_r) \in U$

Como $x \neq y$,

entonces tiene que existir al menos un j tal que $x_j \neq y_j$

Vamos a elegir el vector random a

Elegimos al azar todos los los a_i / $i \neq j$

El elegir a_j tal que $h_a(x) = h_a(y)$ ← forzamos a que sean sinónimos

Es nuestra propuesta una C.F.H.U? (cont.)

Como queremos

$h_a(x) = h_a(y) \rightarrow h_a(x) - h_a(y) = 0$

Entonces:

No divisible por p

$$\underbrace{a_j.(x_j - y_j)}_{z} = \underbrace{\sum_{i \neq j} a_i(x_i - y_i)}_{m} \bmod p \quad \Rightarrow \quad a_j.z = m \bmod p$$

Valor fijo ya que todos estos a_i están fijados

Por la propiedad

Solo existe un valor a_j que satisface esta igualdad ($0 \leq a_j < p$).

Entonces solo existe una probabilidad de 1/p de elegirlo.

El resto de los valores no influyen en esta selección. Por lo tanto la probabilidad global es 1/p (por lo tanto es una clase universal de funciones de hashing)

Presentación realizada en Julio de 2020