

## Guía 4: Hashing

1. Al construir una función de hashing perfecta y mínima con el método visto en clase explique por que es necesario que  $M$  sea mayor a  $N$ .
2. Encontrar el  $m$  mínimo y los parámetros  $a$  y  $b$  de forma tal que la función de hashing  $ax + b \bmod m$  sea perfecta para las siguientes claves: 1,3,5,12
3. Tenemos vectores en 4 dimensiones y usamos "the hashing trick" usando el método de una única función de hashing (es decir sin signo) para reducirlos a 3 dimensiones. Sabemos que la matriz asociada a la función de hashing usada es la siguiente (por filas): [1,0,0; 0,0,1; 0,1,0; 1,0,0]. Se pide construir la función de hashing  $h(x)$  equivalente a la matriz presentada.
4. Dada la siguiente función de hashing que pertenece a la familia Universal de Carter-Wegman para números enteros:  $h(x) = [(4 \cdot x + 3) \bmod 13] \bmod 5$ . Usamos  $h$  para construir un esquema FKS para las siguientes claves: 20,40,70,10,100. Indicar la estructura final resultante y la en caso de ser necesario la segunda función de hashing a usar para el segundo nivel teniendo en cuenta que debe ser pertenecer a la familia  $[(a \cdot x + b) \bmod 13] \bmod m$
5. Tenemos un total de 10.000 claves de 32 bytes c/u. Si usamos el esquema FKS y la primer tabla tiene 1000 posiciones. ¿Cuánto espacio necesitamos en total para almacenar las 10.000 claves?
6. Supongamos que asignamos a cada letra del alfabeto un número de la forma  $A=1$ ,  $B=2$ ,  $C=3$ , etc... Proponemos como función de hashing sumar el valor correspondiente a cada carácter del string y luego tomar el módulo con un cierto número primo  $p$ . Analizar la función propuesta indicando:
  - a) Cantidad de colisiones
  - b) Facilidad de encontrar sinónimos
  - c) Eficiencia
  - d) Efecto avalancha
7. Determinar si las siguientes afirmaciones son V / F justificando la respuesta:
  - a. Una función de hashing criptográfica produce muy pocas colisiones.
  - b. La construcción de DavisMeyer es necesaria para que la función de hashing produzca muy pocas colisiones.

- c. El efecto avalancha se produce cuando muchas claves hashean a un mismo valor generando muchas colisiones .
- d. Una función de hashing para strings debe poder generar el resultado muy rápidamente.
- e. Utilizando una función de hashing con resolución de colisiones Hopscotch de distancia máxima 4 me aseguro insertar como mínimo 4 sinonimos.
- f. Dado que una función de hashing criptográfica tiene pocas colisiones es ideal para asegurarnos buscar claves rápidamente y su uso es recomendable en la mayoría de los casos.
- g. Si las claves a hashear son numéricas y aleatorias entonces no es necesario que el número  $m$  sea primo en la función de hashing .  $a \cdot x \bmod m$ .
- h. Aumentar la cantidad de funciones de hashing o la cantidad de registros por bucket en el método del cuckoo sirve para reducir la cantidad de accesos promedio que necesitamos para recuperar una clave.
- i. Si  $H$  es una familia de funciones de hashing universal definida por  $h(x,a)$  en donde “ $a$ ” es un parámetro entonces la familia  $H_2$  definida  $h_2(x,a)=h(x,a) \bmod p$  también es universal.
- j. Una función de hashing criptográfica como SHA-256 genera menos colisiones que una función genérica como Jenkins pero es mucho mas lenta.
- k. El tiempo necesario para generar una función de hashing perfecta usando Hash & Displace depende de la cantidad de colisiones que genere la primer función de hashing.
- l. Dado un conjunto de claves numéricas de 32 bits la función de hashing  $x \% 1000$  es igual de buena que la función de hashing  $x \% 1001$ .
- m. Una función de hashing puede ser perfecta y sin embargo no servir como función de hashing criptográfica.