Simulación de una Tabla Hash

El trabajo consiste en simular una Tabla Hash con m = 20 posiciones y queremos almacenar valores enteros en el rango [0,1000].

Como las claves son números enteros, vamos a usar la función hash1 de la Familia Universal para enteros estudiada en clase

H: ((K * a + b) % p) % m.

Primero, debemos elegir los valores de los parámetros a, b y p. Para ello, debemos de tener en consideración lo siguiente:

Para que una función hash para enteros sea una familia universal debe cumplir para todo a,b:

- 1<=a<=p-1,
- 0<=b<=p-1,

, siendo P un número primo > |U| y m la cardinalidad de la función hash. Con las condiciones dadas, elegimos los valores a=32, b = 2 y p=97. Por tanto, la función que utilizaremos será:

H: ((K * 32 + 2) % 97) % 20.

Partiendo de una tabla vacía, simular las siguiente secuencia de operaciones explicando detalladamente cada paso:

- Insertamos diez valores aleatorios elegidos del rango posible sin repetición.
  - Insertar {24, 63, 58, 70, 41, 10, 2, 98, 12, 49}
- Seleccionamos dos de estas claves al azar para borrarlas.
  - borrar {98, 70} (borrar implica primero buscar para mover el cursor y después borrar).
- Buscar dos valores de clave: una de las que se ha borrado previamente y otra que no haya sido borrada.
  - Buscar {98, 10}

Se recomienda usar una hoja de cálculo para realizar el cálculo de las funciones hash sobre los valores de clave seleccionados. La operación resto de división a:b (a % b) suele ser "RESIDUO(A;B)" en la hoja de cálculo.

La tabla inicial, dados los datos dados, es:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19

CASO 1. ENCADENAMIENTO

- h(24) = 11 → isEmpty(tabla[11]) = true → tabla[11].add(24).
- h(63) = 18 → isEmpty(tabla[18]) = true → tabla[18].add(63)
- h(58) = 15 → isEmpty(tabla[15]) = true → tabla[15].add(58)
- h(70) = 11 → isEmpty(tabla[11]) = false → tabla[11].new(70)
- h(41) = 13 → isEmpty(tabla[13]) = true → tabla[13].add(41)
- h(10) = 11 → isEmpty(tabla[11]) = false → tabla[11].new(10)
- h(2) = 6 → isEmpty(tabla[6]) = true → tabla[6].add(2)
- h(98) = 14 → isEmpty(tabla[14]) = true → tabla[14].add(98)
- h(12) = 15 → isEmpty(tabla[15]) = false → tabla[15].new(12)
- h(49) = 18 → isEmpty(tabla[18]) = false → tabla[18].new(49)

Tras realizar las inserciones, la tabla inicial quedaría del siguiente modo:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
						2					10		41	98	12			49	
											70				58			63	
											24								

- h(98) = 14 → isEmpty(tabla[14]) = false && tabla[14].find(98)
  - tabla[14].remove(98)
- h(70) = 11 → isEmpty(tabla[11]) = false && tabla[11].find(70)
  - tabla[11].remove(70)
- h(98) = 14 → isEmpty(tabla[14]) = true → return false
- h(10) = 11 → isEmpty(tabla[11]) = false && tabla[11].find(10)
  - return true

Tras realizar las eliminaciones, la tabla quedaría así:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
						2					10		41		12			49	
											24				58			63	

CASO 2. LINEAR PROBING

- h(24) = 11 → isEmpty(tabla[11]) = true → tabla[11].add(24).
- h(63) = 18 → isEmpty(tabla[18]) = true → tabla[18].add(63)
- h(58) = 15 → isEmpty(tabla[15]) = true → tabla[15].add(58)
- h(70) = 11 → isEmpty(tabla[11]) = false → Colisión
  - isEmpty(tabla[12]) = true → tabla[12].add(70).
- h(41) = 13 → isEmpty(tabla[13]) = true → tabla[13].add(41)
- h(10) = 11 → isEmpty(tabla[11]) = false → Colisión
  - isEmpty(tabla[12]) = false → Colisión
  - isEmpty(tabla[13]) = false → Colisión
  - isEmpty(tabla[14]) = true → tabla[14].add(11)
- h(2) = 6 → isEmpty(tabla[6]) = true → tabla[6].add(2)
- h(98) = 14 → isEmpty(tabla[14]) = false → Colisión
  - isEmpty(tabla[15]) = false → Colisión
  - isEmpty(tabla[16]) = true → tabla[16].add(98)
- h(12) = 15 → isEmpty(tabla[15]) = false → Colisión
  - isEmpty(tabla[16) = false → Colisión
  - isEmpty(tabla[17]) = true → tabla[17].add(12)
- h(49) = 18 → isEmpty(tabla[18]) = false → Colisión
  - isEmpty(tabla[19]) = true → tabla[19].add(49)

Tras realizar las inserciones, la tabla inicial quedaría del siguiente modo:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
						2					24	70	41	10	58	98	12	63	49

- h(98) = 14 → isEmpty(tabla[14]) = false && !tabla[14].find(98)
  - isEmpty(tabla[15]) = false && !tabla[14].find(98)
  - isEmpty(tabla[16]) = false && tabla[16].find(98)

    - tabla[16].remove(98)
- h(70) = 11 →isEmpty(tabla[11]) = false && !tabla[11].find(70)
    - isEmpty(tabla[12]) = false && tabla[12].find(70)
    - tabla[12].remove(70)
- h(98) = 14 → isEmpty(tabla[14]) = false && !tabla[14].find(98)
    - isEmpty(tabla[15]) = false && !tabla[15].find(98)
    - isEmpty(tabla[16]) = false && !tabla[16].find(98)
    - isEmpty(tabla[17]) = false && !tabla[17].find(98)
    - isEmpty(tabla[18]) = false && !tabla[18].find(98)
    - isEmpty(tabla[19]) = false && !tabla[19].find(98)
    - isEmpty(tabla[0]) = true → return false
- h(10) = 11 → isEmpty(tabla[11]) = false && !tabla[11].find(10)
    - isEmpty(tabla[12]) = false && !tabla[12].find(10)
    - isEmpty(tabla[13]) = false && !tabla[13].find(10)
    - isEmpty(tabla[14] = false && tabla[14].find(10) → return true

Al final, la tabla se nos quedaría así.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
						2					24	X	41	10	58	X	12	63	49

CASO 3. RANDOM PROBING c = 9

- h(24) = 11 → isEmpty(tabla[11]) = true → tabla[11].add(24).
- h(63) = 18 → isEmpty(tabla[18]) = true → tabla[18].add(63)
- h(58) = 15 → isEmpty(tabla[15]) = true → tabla[15].add(58)
- h(70) = 11 → isEmpty(tabla[11]) = false → Colision
    - isEmpty(tabla[0]) = true → tabla[0].add(70)
- h(41) = 13 → isEmpty(tabla[13]) = true → tabla[13].add(41)
- h(10) = 11 → isEmpty(tabla[11]) = false → colision
    - isEmpty(tabla[0]) = false → colision
    - isEmpty(tabla[9]) = true → tabla[9].add(10)
- h(2) = 6 → isEmpty(tabla[6]) = true → tabla[6].add(2)
- h(98) = 14 → isEmpty(tabla[14]) = true → tabla[14].add(98)
- h(12) = 15 → isEmpty(tabla[15]) = false → colision

  - isEmpty(tabla[4]) = true → tabla[4].add(12)
- h(49) = 18 → isEmpty(tabla[18]) = false → colision
  - isEmpty(tabla[7]) = true → tabla[7].add(49)

Tras realizar las inserciones, la tabla inicial quedaría así:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
70				12		2	49		10		21		41	98	58			63	

- h(98) = 14 → isEmpty(tabla[14]) = false && tabla[14].find(98)
  - tabla[14].remove(98)
- h(70) = 11 → isEmpty(tabla[11]) = false && !tabla[11].find(70)
  - isEmpty(tabla[0]) = false && tabla[0].find(70)
  - tabla[0].remove(70)
- h(98) = 14 → isEmpty(tabla[14]) = false && !tabla[14].find(98)
  - isEmpty(tabla[3]) = true → return false
- h(10) = 11 → isEmpty(tabla[11]) = false && !tabla[11].find(10)
  - isEmpty(tabla[0]) = false && !tabla[0].find(10)
  - isEmpty(tabla[9]) = false && tabla[9].find(10) → return true

Tras realizar el las eliminaciones, la tabla quedaría así:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
X				12		2	49		10		21		41	X	58			63	

Caso 4: Rehashing

Para la resolución de este caso, vamos a utilizar las siguientes funciones:

- h2 = ((K * 2 + 15) % 97) % 20
- h3 = ((K * 4 + 20) % 97) % 20

Si encontramos error en h1, buscaremos la siguiente casilla vacía usando h2. Si se repite, usando h3.

En caso de no encontrar casilla vacía al utilizar h3, aplicaremos linear probing.

- h1(24) = 11 → isEmpty(tabla[11]) = true → tabla[11].add(24)
- h1(63) = 18 → isEmpty(tabla[18]) = true → tabla[18].add(63)
- h1(58) = 15 → isEmpty(tabla[15]) = true → tabla[15].add(58)

- h1(70) = 11 → isEmpty(tabla[11]) = false → colision
    - h2(70) = 3 → isEmpty(tabla[3]) = true → tabla[3].add(70)
- h1(41) = 13 → isEmpty(tabla[13]) = true → tabla[13].add(41)
- h1(10) = 11 → isEmpty(tabla[11]) = false → colision
    - h2(10) = 0 → isEmpty(tabla[0]) = true → tabla[0].add(10)
- h1(2) = 6 → isEmpty(tabla[6]) = true → tabla[6].add(2)
- h1(98) = 14 → isEmpty(tabla[14]) = true → tabla[14].add(98)
- h1(12) = 15 → isEmpty(tabla[15]) = false → colision
    - h2(12) = 4 → isEmpty(tabla[4]) = true → tabla[4].add(12)
- h1(49) = 18 → isEmpty(tabla[18]) = false → colision
    - h2(49) = 1 → isEmpty(tabla[1]) = true → tabla[1].add(49)

Tras realizar las inserciones, la tabla inicial quedaría así:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
10	49		70	12		2					24		41	98	58			63	

- h1(98) = 14 → isEmpty(tabla[14]) = false && tabla[14].find(98)
    - tabla[14].remove(98)
- h1(70) = 11 → isEmpty(tabla[11]) = false && !tabla[11].find(70)
    - h2(70) = 3 → isEmpty(tabla[3]) = true && tabla[3].find(70)
    - tabla[3].remove(70)
- h1(98) = 14 → isEmpty(tabla[14]) = false && !tabla[14].find(98)
    - h2(98) = 2 → isEmpty(tabla[2]) = true → return false
- h1(10) = 11 → isEmpty(tabla[11]) = false && !tabla[11].find(10)
    - h2(10) = 0 → isEmpty(tabla[0]) = false && tabla[0].find(10)
    - return true

Tras realizar las eliminaciones, la tabla quedaría así:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
10	49		X	12		2					24		41	X	58			63	