TP4: Organización directa: Hashing

Alumno: Juan Cruz Mateos
Mat. 15134

1) Utilizando como función hash **f(x) = x mod 1301**, conceptualice un archivo directo de **1300 posiciones** y ubique los siguientes valores:

10168, 21362, 8515, 7993, 32434, 29786, 35175, 50255, 32305, 24971, 10177, 53101, 12901

Indiquen la cantidad máxima de claves, y cómo calcularía el factor de ocupación. ¿Existieron colisiones? Para 1500 registros ¿qué probabilidad de colisiones hay?

Clave	f(Clave) = Clave mod 1301
10168	1061
21362	546
8515	709
7993	187
32434	1210
29786	1164
35175	48
50255	817
32305	1081
24971	252
10177	1070
53101	1061 // colision con clave 10168
12901	1192

*En el archivo **ejercicio01.py** se encuentra el programa para calcular los valores obtenidos.*

a. Cantidad maxima de claves:

La cantidad maxima de claves que se prodran almacenar en el archivo sera 1300, puesto que es el tamaño maximo de registros que puede almacenar. La cantidad maxima de direcciones devueltas por la funcion de hash es 1301, estando el espacio de direcciones definido por el intervalo [0, 1300]. Como el archivo solo dispone de 1300 registros (0 … 1299), cuando la funcion hash devuelva el valor 1300 no existira la direccion fisica donde almacenarlo, por lo que puede redireccionarse a la posicion 0 del archivo.

b. Factor de ocupacion:

Se define el factor de ocupacion como el cociente entre el numero de registros ocupador y el numero de registros disponibles. Suponiendo que el archivo guarda unicamente un registro por direccion y no resuelve las colisiones sino que las ignora pisando el valor anterior con el nuevo (la clave 53101 colisiona con la clave 10168, y como la direccion ya se encuentra ocupada ocurre desbordamiento), la cantidad final de resistros ocupados luego de insertar las claves correpondientes en sus direcciones es de 12. Luego,

factor de ocupacion = # registros_ocupados / # registros_disponibles
= 12 / 1300
= 0.0092

c. Colisiones:

- la clave a) 10168 colisiona con la clave l) 53101, ambas apuntando a la direccion 1061.

d. La probabilidad que una determinada direccion posea x registros asignados se puede calcular como:

$$p(x) \sim (r/N)^x * e^{(-r/N)} / x! \qquad \text{para } N \text{ y } r \text{ grandes}$$

p(x): probabilidad de que una direccion dada tenga x registros asignados
N: numero de direcciones disponibles
r: numero de registros a guardar

Además, para N direcciones, el numero esperado de direcciones con x registros se cualcula como:

$$N * p(x)$$

Asumiendo que unicamente un registro puede ser asignado a una direccion dada, el numero esperado de desbordamientos es:

$$N * p(0) – (N-r)$$

Luego, la probabilidad de que ocurran colisiones es la probabilidad de que a una determinada direccion posea mas de un registro asignado (es decir, que contenga claves sinonimas). Entonces,

siendo $N = 1300$
$r = 1500$

prob coliciones $= (1 – p(0) – p(1))$
$= 0.3206$

y el numero esperado de desbordamientos es:

overflows $= 1300 * p(0) – (1300 – 1500)$
~ 610

2) Suponga un archivo de organización directa de **512 posiciones**. Usando una función hash con **método de compresión** y seleccionando el número primo adecuado, calcule las posiciones donde irían los siguientes registros alfanuméricos: limon, gato, vodka, ojo, ojota, caradura, caradura, oh.

Dado que se trata de un archivo de 512 posiciones, elijo el numero primo más proximo a 512 a fin de asegurar una distribucion de claves mas uniforme, cuidando ademas que dicho numero no sea potencia de 2. El menor primo mas cercano a 512 es 509, por lo que elijo ese valor.

```
clave = limon
l: 1101100
i: 1101001
m: 1101101
o: 1101111
n: 1101110
xor = 105
105 mod 509 = 105

clave = gato
g: 1100111
a: 1100001
t: 1110100
o: 1101111
xor = 29
29 mod 509 = 29

clave = vodka
v: 1110110
o: 1101111
d: 1100100
k: 1101011
a: 1100001
xor = 119
119 mod 509 = 119

clave = ojo
o: 1101111
j: 1101010
o: 1101111
xor = 106
106 mod 509 = 106

clave = ojota
o: 1101111
j: 1101010
o: 1101111
t: 1110100
a: 1100001
xor = 127
127 mod 509 = 127

clave = caradura
c: 1100011
a: 1100001
r: 1110010
a: 1100001
d: 1100100
u: 1110101
r: 1110010
a: 1100001
xor = 19
19 mod 509 = 19

clave = caradura
c: 1100011
a: 1100001
r: 1110010
```

Resultados:

clave	hash
limon	105
gato	29
vodka	119
ojo	106
ojota	127
caradura	19
caradura	19 (colision)
ah	9

```
a: 1100001
d: 1100100
u: 1110101
r: 1110010
a: 1100001
xor = 19
19 mod 509 = 19

clave = ah
a: 1100001
h: 1101000
xor = 9
9 mod 509 = 9
```

*En el archivo **ejercicio02.py** se encuentra el programa para el calculo de los valores obtenidos.*

3) ¿En qué se diferencia y qué ventajas tienen el manejo de colisiones dispersión, por distribución por almacenamiento por cubetas y por área de overflow separada?

Distribución por almacenamiento por cubetas: son asignados bloques de espacio (cubetas) para almacenar multiples registros en lugar de asignar celdas individuales a registros. Este método busca minimizar la cantidad de claves por cubeta para aumentar la velocidad de acceso. Cada cubeta tiene un puntero que apunta a una lista enlazada donde se almacenan todos los registros que tengan como dirección la calculada con la función hash. Resulta util el caso de haber muchas colisiones y dispersas, por lo que un recorrido secuencial no serviría y disminuiría la velocidad. El gasto lógico y de memoria es mayor ya que se tiene que almacenar un puntero para cada cubeta.

Área de overflow separada: se cuenta con un archivo de registros primario donde se almacenan los registro y las colisiones que puedan ocurrir son direccionadas a otro archivo distinto donde se las gestiona. Aqui se utiliza nuevamente el concepto de lista enzalada, guardando en el archivo de datos la cabeza a la lista de coliciones en el archivo de overflow, que contiene cada registro que dio lugar a una colision y un puntero al proximo registro sinonimo dentro del archivo. Este metodo, por lo tanto, implica en mantenimiento de dos archivos separados y la gestion de las listas enlazadas dentro del archivo de overflow cuya cabeza se guarda en el archivo principal.

Ejercicio de programación:

1) Suponga un sistema para la gestión de una maratón de 42Km. El cupo máximo es de 12000 corredores. En la inscripción cada corredor provee: número de DNI, nombre, sexo, edad, categoría. Hay un último campo que es el tiempo (que se carga al finalizar la carrera).

Para la solución se deberá usar un archivo directo, donde deberá diseñar una función hash que mapee número de DNI a posición en el archivo (que es el número de corredor otorgado en el momento de la inscripción).

Implemente: función hash, archivo, altas, modificaciones, bajas. Carga de tiempos. Listados de tiempos general y por categoría.

¿Es eficiente esta solución al momento de imprimir los listados de clasificaciones generales y por categorías? ¿Usaría la misma solución si la toma de tiempo se hace de forma electrónica con sensores tipo RFID en la largada, media maratón (km 21) y final (Km 42)? ¿Qué cambiaría?

*Resuelto en archivo **ejercicio01.c***

Las impresiones de los listados de tiempos no es eficiente, ya que el archivo no se encuentra ordenado ni por el tiempo ni por la categoria. Para listar clasificaciones generales habria que ordenar el archivo por tiempo de manera ascendente y luego listar recorriendo secuencialmente. De la misma manera, para listar por categoria deberia ordenarse el archivo por categoria, y dentro de cada categoria deberia ordenarse tambien por tiempo de forma descendente, y luego realizar el listado recorriendo el archivo secuencialmente. Como se ve, la estructura del archivo tal como esta no facilita dichas consultas. Estas dificultades podrian salvarse utilizando archivos adicionales: un archivo general de tiempos ordenado por tiempo de forma descendente, cuyos registros contengan el numero de corredor y su tiempo. De esta manera, a fin de generar el listado de tiempos general, bastaria con recorrer secuencialmente dicho archivo de tiempos e indexado de manera directa al archivo principal con el numero de corredor para obtener los datos del corredor.

En caso de registrar tres tiempors utilizando sensores RFID, se deberia modificar la estructura del registro para que en lugar de un solo tiempo contenga cada uno de los tres tiempos registrados por el sensor.

2) Suponga un sistema para el manejo de personal de una empresa. Los datos a almacenar son apellido y nombre, dirección, teléfono, y fecha de ingreso.

Para la solución se plantea un archivo directo donde el apellido y nombre es la clave de acceso.

Implemente una función hash, creación de archivo principal, alta, modificaciones, bajas. El manejo de colisiones se debe resolver por área separada de overflow.

Resuelto en archivo ***ejercicio02.c***