Tablas de Dispersión
(Hashing Tables)

Las tablas de dispersión o hashing tables (en inglés) es una técnica que se utiliza para implementar inserciones, eliminaciones y búsquedas en un tiempo medio constante.

La estructura de datos central de esta técnica es la tabla de hashing (**tabla de dispersión**.)

Planteamiento inicial

La estructura de datos ideal para la tabla de dispersión es un arreglo de tamaño fijo que contiene las claves (elementos de la tabla). Una clave suele ser una cadena de caracteres (o de números) con un valor asociado Si el tamaño de la tabla es MAX_T, la tabla se declara entre 0 y MAX_T-1. A cada clave se le hará corresponder un número entre 0 y MAX_T-1 y se colocará en la celda correspondiente.

La relación entre la llave y la posición en la tabla es lo que se llama función de dispersión.

En la figura siguiente se muestra cómo sería una tabla de dispersión ideal (cada llave cae en una celda distinta).

0	
1	
2	Ana
3	
4	Juan
5	Eva
6	
7	Felipe
8	
9	

Función Hash (o de Dispersión)

Es la correspondencia entre la clave y un índice del arreglo. Por lo general, cuando las claves son números enteros la función de dispersión toma la forma:

h(x) = clave MOD MAX_T

El tamaño de la tabla debe ser primo. De esta forma, y si las claves son números aleatorios, esta función suele distribuir las claves uniformemente. Si tuviéramos la tabla de la figura anterior (MAX_T =10), y todas las claves terminaran en cero, la dispersión con esta función no nos valdría.

Cuando las claves son cadenas de caracteres lo usual es sumar los valores ASCII de los caracteres de la cadena.

A continuación se declaran los tipos apropiados para la tabla de dispersión y el índice, que es el que devuelve la función de dispersión.

```
TYPE
   INDICE = 0 .. MAX_T - 1;
   TablaDisp= ARRAY INDICE OF ...
```

Una implementación sencilla de la función de dispersión sería la siguiente:

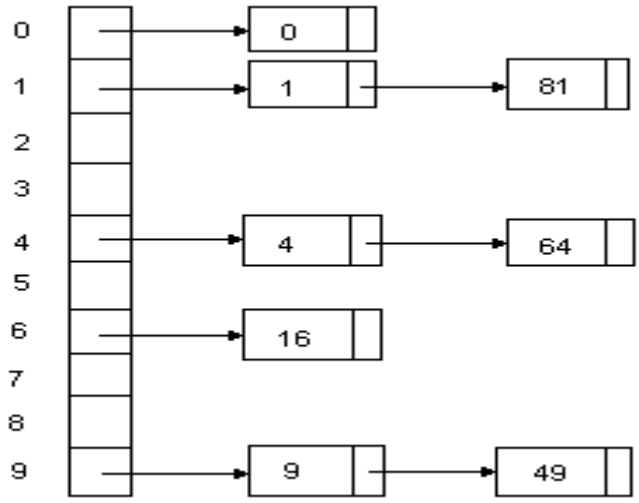
```
FUNCTION hashing (llave: TipoCadena): INDICE;
 VAR
   aux, j: integer;
 BEGIN
   aux := ord (llave[1]);
   FOR j:=2 TO Length (llave) DO
     aux := aux + ord (llave[j]);
   hashing := aux MOD MAX_T;
 END;
```

El problema de esta función es que, si la tabla es grande, no hace una distribución uniforme de las claves. Pongamos un ejemplo: tenemos una tabla con un tamaño MAX_T= 10007 (primo) y las claves tienen una longitud máxima de ocho caracteres; como ord devuelve un número entero de valor máximo 127, la función de dispersión sólo tendrá valores entre 0 y 1016 (127*8). Luego la distribución no es equitativa

A continuaciòn vamos a tratar el tema de las colisiones. Esto ocurre cuando dos elementos distintos toman el mismo valor. Para solucionar las colisiones veremos dos métodos: la dispersión abierta y la dispersión cerrada.

Dispersión abierta

La dispersión abierta, también llamada encadenamiento separado, consiste en tener una lista de los elementos que se dispersan en el mismo valor de la tabla. Suponiendo una tabla de tamaño 10 de números enteros y con la función de dispersión: dispersión(x) = x MOD 10, nos quedaría una tabla de dispersión abierta como la de la figura:



Las **declaraciones de los tipos** necesarios para la dispersión abierta son:

```
TYPE
  posicion =POINTER TO nodo;
  nodo = RECORD
          elem : TipoElemento;
          sig : posicion;
         END;
  TablaDisp = ARRAY INDICE OF posicion;
```

Inicialización de la tabla

Primero debemos inicializar la tabla, asignando a cada celda el valor *NIL*:

```
PROCEDURE IniciaTabla (VAR D: TablaDisp);
VAR
  i : integer;
BEGIN
  FOR i:=0 TO MAX_T-1 DO
    D[i]:= NIL;
END;
```

Búsqueda en la tabla

Para efectuar una búsqueda en una tabla, primero usamos la función de dispersión para determinar la lista a recorrer. Seguidamente, se recorre la lista hasta encontrar la clave y se devuelve un puntero a la posición de la celda que contiene la clave.

```
FUNCTION Buscar (llave: TipoElemento; D: TablaDisp): posicion;
VAR
  res, p: posicion;
BEGIN
  p := D[hashing(llave)];
  res:= NIL;
  WHILE p <> NIL DO
    IF p^.elemento = llave THEN BEGIN
      res:= p; BREAK;
    END;
    p := p^.sig;
  END;
  Buscar:= res;
END;
```

Inserción en la tabla

Para insertar un elemento en una tabla, primero buscamos en la lista que le corresponde para ver si ya está insertado; si es nuevo, se inserta al principio de la lista:

```
PROCEDURE Insertar (llave:TipoElemento; VAR D: TablaDisp);
VAR
  h: INTEGER;
  pos, lista : posicion;
  nuevo : posicion;
BEGIN
  pos := Buscar (llave, D);
  IF pos = NIL THEN BEGIN (* no encontrado *)
    h:= hashing(llave);
    NEW (nuevo);
    nuevo^.elem := llave;
    nuevo^.sig := D[h];
    D[h]:= nuevo;
  END;
END;
```

Factor de carga

Llamamos factor de carga de una tabla de dispersión, λ, a la razón entre el número de elementos en la tabla y el tamaño de la misma:

$$\lambda = \frac{\text{nº elementos tabla}}{\text{tamaño tabla}}$$

La longitud media de una lista es λ. El tiempo que se tarda en realizar una búsqueda será el tiempo en que se calcula la función de dispersión más el tiempo necesario para recorrer la lista. Si la búsqueda es infructuosa, el número medio de enlaces por recorrer es λ. Si la búsqueda tiene éxito los enlaces por recorrer son, por término medio, $1 + \lambda /2$.

De esto se deduce que el factor de carga es importante a la hora de diseñar una tabla. En el caso de la dispersión abierta la regla general es que λ **tienda a 1** (es decir, el tamaño de la tabla igual a los elementos esperados).

Tampoco debemos olvidar que el tamaño de la tabla debe ser **primo**.

Dispersión cerrada

La dispersión cerrada o direccionamiento abierto, soluciona las colisiones buscando celdas alternativas hasta encontrar una vacía (dentro de la misma tabla).

Se va buscando en las celdas: d0(x), d1(x), d2(x), ..., donde

di(x) = (hashing(x) + f(i)) MOD MAX_T.

La función f( ) es la estrategia de resolución de las colisiones, y se debe cumplir que:

$$f(i)=\begin{cases} 0 & \text{si } i=0 \\ <>0 & \text{si } i<>0 \end{cases}$$

Al estar todos los datos en la misma tabla, se necesita que ésta sea más grande; el factor de carga tiene que ser menor o igual a 0.5.

Dentro de la dispersión cerrada hay tres estrategias distintas: la exploración lineal, la exploración cuadrática y la dispersión doble.

Exploración lineal

En este tipo de estrategia la función f() es una función lineal de i, por ejemplo: f(i) = i. Esto significa que las celdas se recorren en secuencia buscando una celda vacía; es decir, si hay una colisión se prueba en la celda siguiente y así sucesivamente hasta encontrar una vacía.

Lo veremos en un ejemplo: tenemos una tabla con MAX_T = 10 y la función de dispersión h(x)= x MOD 10.

(Se recuerda que MAX_T debe ser primo, aquí se ultiliza 10 por sencillez de cálculo, pero sería un grave error encontrarlo en un código real.)

En este ejemplo, que es el de la figura de abajo, la primera colisión ocurre cuando queremos insertar el 49. Como la celda 9 está ocupada se pone en la siguiente celda desocupada. Cuando el 58 entra en colisión con el 18 va a la siguiente celda y entra en colisión con el 89, y después con el 49 antes de encontrar su posición. Con el 69 pasa algo parecido.

	Tabla vacía	Después de 89	Después de 18	Después de 49	Después de 58	Después de 69
0				49	49	49
1					58	58
2						69
3						
4						
5						
6						
7						
8			18	18	18	18
9		89	89	89	89	89

Las desventajas que tiene este método son: el tiempo que se tarda en encontrar una celda vacía y la formación de bloques de celdas ocupadas, llamada efecto agrupamiento primario. A causa de este efecto, cualquier clave que se disperse en un agrupamiento realizará varios intentos para resolver la colisión, y agrandará el agrupamiento.

Para inserciones y búsquedas no exitosas el número de intentos aproximado sería: $1/2(1 + 1/(1 - \lambda)^2)$. Para búsquedas exitosas sería: $1/2(1 + 1/(1 - \lambda))$, un número menor que el anterior.

Exploración cuadrática

Este método de resolución de colisiones elimina el problema del agrupamiento primario. La función de colisiones es cuadrática: $f(i) = i^2$. En la figura que sigue se muestra la tabla de dispersión cerrada que resulta al aplicar al ejemplo anterior la función de resolución cuadrática:

	Tabla vacía	Después de 89	Después de 18	Después de 49	Después de 58	Después de 69
0				49	49	49
1						
2					58	58
3						69
4						
5						
6						
7						
8			18	18	18	18
9		89	89	89	89	89

En este caso, cuando el 49 entra en colisión con el 89, la siguiente posición es la celda siguiente (f(1) = 12 = 1).
Después, cuando el 58 entra en colisión también intenta la celda siguiente, pero está ocupada.
En su segunda colisión, el 58 intenta la celda que está 4 posiciones más allá (22 = 4), y como está libre se coloca en ella (en la celda 2).
Con el 69 ocurre lo mismo.

En este tipo de exploración no hay garantía de encontrar una celda vacía si la tabla se llena a más de la mitad, o antes si el tamaño de la tabla no es primo. Existe un Teorema que dice:

Si se usa la exploración cuadrática, y el tamaño de la tabla es primo, entonces siempre se puede insertar un elemento nuevo si la tabla está, al menos, medio vacía.

Aunque la exploración cuadrática elimina el agrupamiento primario, se produce otro fenómeno llamado agrupamiento secundario, ya que las claves que se dispersan a la misma posición intentarán las mismas celdas.

Dispersión doble

La función de colisiones para la dispersión doble es: f (i) = i * h2 (x). Lo que se hace es aplicar una segunda función de dispersión a x, y luego se prueba a distancias h2(x), 2h2(x), ...

Es muy importante la buena elección de h2(x) y, además, nunca debe ser cero. Si elegimos la función:

h2(x) = R - (x MOD R)

con R un número primo menor que MAX_T, funcionará bien.

En la figura siguiente mostramos cómo quedaría la tabla de dispersión cerrada con dispersión doble, cuando insertamos las mismas claves que en los ejemplos anteriores y para R = 7.

	Tabla vacía	Después de 89	Después de 18	Después de 49	Después de 58	Después de 69
0						69
1						
2						
3					58	58
4						
5						
6				49	49	49
7						
8			18	18	18	18
9		89	89	89	89	89

La primera colisión ocurre al insertar el 49; calculando la función de dispersión nos quedaría: h2(49) = 7 - (49 MOD7), que es igual a: h2(49) = 7 - 0 = 7; luego el 49 se insertará en la posición 6.
En la segunda colisión, al insertar el 58, la función será: h2(58) = 7 - 2 = 5, por lo tanto el 58 se colocará en la posición 3.
Finalmente intentamos insertar el 69, que con la función de dispersión iría en la posición: h2(69) = 7 - 6 = 1, por lo que la celda resultante es la cero.

Aunque en el ejemplo el tamaño de la tabla no es primo, por hacer más fáciles los cálculos, sería conveniente que fuera primo siempre. De esta manera hay más posiciones alternativas que cuando no es primo.

La conclusión es que con la dispersión doble se consiguen buenos resultados, aunque es más compleja y, por lo tanto, más lenta que otras soluciones (como la exploración cuadrática).

Declaraciones de tipos

Las declaraciones de los tipos para implantar la dispersión cerrada son las siguientes:

```
TYPE
  ClaseEntrada = (legal, vacia, eliminada);
  Entrada_disp = RECORD
                   elem : TipoElemento;
                   info : ClaseEntrada;
                 END;
  posicion = INDICE;
  TablaDisp = ARRAY [INDICE] OF Entrada_disp;
```

Inicialización de la tabla

La puesta a valores iniciales de la tabla se realiza poniendo el campo info a vacío.

```
PROCEDURE IniciarTabla (VAR D: TablaDisp);
VAR
  i: integer;
BEGIN
  FOR i:=0 TO MAX_T-1 DO
    D[i].info := vacia;
END;
```

Búsqueda para dispersión cerrada con exploración cuadrática

La rutina de búsqueda devolverá la posición de la llave en la tabla de dispersión. Si no se encuentra, devuelve la celda donde estaría insertada la llave, que estará marcada como vacía.
Suponemos que la tabla de dispersión es al menos el doble de grande que el número de elementos de la tabla, ya que así la resolución cuadrática funcionará. La implementación es:

```
FUNCTION Buscar (llave: TipoElemento; D:TablaDisp): INDICE;
VAR
  i, pos : posicion;
BEGIN
  i := 0;
  pos := hashing (llave);
```

```
  WHILE D[pos].info <> vacia DO BEGIN
    IF D[pos].elem = llave THEN (* encontrado *)
       BREAK;
    END;
    INC (i);
    pos := (pos + 2 * i - 1);
    IF pos > MAX_T THEN
       pos := pos - MAX_T;
  END;
  Buscar:= pos;
END;
```

Aquí se utiliza la forma rápida de hacer la resolución cuadrática. Siendo la definición de la función cuadrática: f(i) = f(i-1) + 2i - 1, se deduce que se puede implementar con una multiplicación por dos y un decremento. Si la posición resultante se sale del array, se le resta el tamaño de la tabla.

Por otro lado, es trivial modificar esta función para que reaproveche huecos eliminados en caso de no encontrar el elemento:

```
FUNCTION BuscarMejor (llave: TipoElemento; D:TablaDisp): INDICE;
VAR
  i, pos, hueco : posicion;
BEGIN
  i := 0;
  hueco := -1; (* todavia no hay hueco *)
  pos := hashing (llave);
  WHILE D[pos].info <> vacia DO BEGIN
    IF D[pos].info = eliminada THEN BEGIN (* hay hueco *)
       hueco := pos; (* no acabamos porque quizá "llave" este más adelante *)
    END;
    IF D[pos].elem = llave THEN (* encontrado *)
       BREAK;
    END;
    INC (i);
    pos := (pos + 2 * i - 1);
    IF pos > MAX_T THEN
       pos := pos - MAX_T;
  END;
  Buscar:= pos;
END;
```

Inserción en tablas de dispersión cerrada

En el caso de la inserción, cuando la clave está ya en la tabla no se hace nada; si no, se coloca en la posición que resulte de la rutina buscar:

```
PROCEDURE Insertar (llave: TipoElemento; VAR D: TablaDisp);
VAR
  p; pos: INDICE;
BEGIN
  pos := Buscar (llave, D);
  IF D[pos].info <> legal THEN BEGIN (* celda apropiada para insertar *)
    D[pos].elem := llave;
    D[pos].info := legal;
  END;
END;
```