Algoritmos y Estructuras de Datos

Hashing

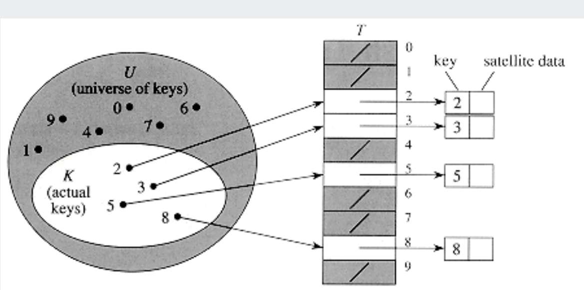
1

Hashing

- Existen otros métodos para la búsqueda no basados en comparación de claves… (ni digital).
- Cálculos aritméticos con la clave K, que arrojan una dirección f(K) del espacio que contiene los elementos
- “Desmenuzamiento”, “hashing”, “transformación de claves” o “almacenamiento disperso”.
- Mapear un espacio de claves grande sobre un espacio de almacenamiento relativamente pequeño
- Operaciones de búsqueda muy rápidas.

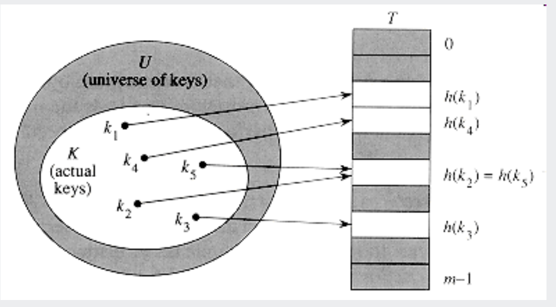
2

Directo, universo pequeño



3

Universo de claves muy grande



4

Hashing

- Dificultad principal: el conjunto de posibles valores resulta mucho mayor que el conjunto de direcciones disponibles.
  - Ejemplo: conjunto de nombres formados por hasta 16 letras, que identifican a los individuos de un grupo de 1000 personas.
  - Habrá $26^{16} = 4.36 * 10^{22}$ claves posibles, que deben mapearse en $1 * 10^3$ índices posibles.
- Las funciones de hash que no produzcan direcciones iguales para claves diferentes son muy raras

5

Hashing

- A la coincidencia **$h(K_i) = h(K_j)$** , con **K_i** <> **K_j** , se la conoce como COLISION.
- Para utilizar una tabla dispersa, se debe entonces resolver dos problemas:
  – **escoger una función de desmenuzamiento h(K) y**
  – **seleccionar un método apropiado para resolver las colisiones**.
- La elección de una función adecuada implica el considerar que las claves se distribuyan lo más uniformemente posible sobre el espacio de almacenamiento.
- La resolución de colisiones implica el obtener un lugar alternativo de almacenamiento para las claves que produzcan colisión.

6

Hashing: Elección de la función de transformación

- La distribución de claves transformadas debe ser lo más aleatoria posible.
- Debe tenerse en cuenta que las claves suelen presentar agrupamientos: por ejemplo, es común que un gran número de palabras tengan el mismo prefijo.
- La función debe ser fácil y rápida de calcular.
- Dado un espacio de almacenamiento de tamaño N, una elección obvia para la función será el llamado “esquema de división”:
  - **h(K) = K mod N**
  - esta función tiene la propiedad de que los valores de las claves están distribuidos uniformemente sobre el intervalo índice.
  - Es eficiente si N es una potencia de 2, pero este caso no arroja buenos resultados cuando las claves son palabras o secuencias de letras.
  - Una recomendación para resolver este problema es no usar N sino un número M primo, mayor que N.

7

Hashing: Elección de la función de transformación

- Esquema multiplicativo:
  - sea w el tamaño de la palabra del ordenador, y sea A una constante entera prima con w. Entonces

$$h(K) = \text{int}\left[M * \left(\left(\frac{A}{w} * K \right) \bmod 1 \right) \right]$$

- Método del centro del cuadrado:
  - Por ejemplo, claves de 10 dígitos en un ordenador decimal. Si N = 1000 hacer K*K y elegir los tres dígitos de cualquier parte cerca del centro de ese producto. Esto debería producir una dispersión suficientemente buena de valores entre 000 y 999, con baja probabilidad de colisión.
  - Este método funciona bastante bien si las claves no tienen muchos ceros al principio o al final, pero no es un buen método para generar números aleatorios.

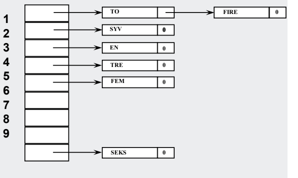
8

Resolución de colisiones por encadenamiento.

- Mantener N listas enlazadas,
- N cabeceras de listas.
- Inserción en listas múltiples.

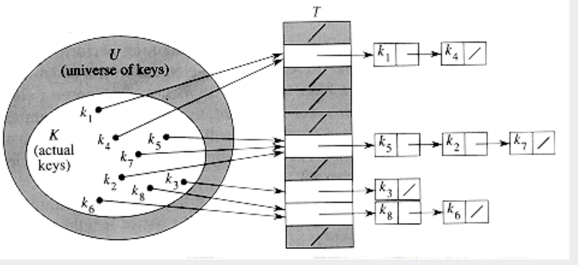
Ejemplo: N = 9, K = EN, TO, TRE, FIRE, FEM, SEKS, SYV

h(K) +1 =3, 1, 4, 1, 5, 9, 2



9

encadenamiento



Algoritmos y Estructuras de Datos 10

10

Resolución de colisiones por encadenamiento abierto

- Eliminar completamente los enlaces, mirando varias entradas de la tabla en forma consecutiva, hasta encontrar la clave K o una posición vacía.
- Formular algunas normas por las que cada clave determine una secuencia de intentos
- Si encontramos una posición vacía mientras buscamos K, podemos concluir que K no está en la tabla.

Algoritmos y Estructuras de Datos 13

13

Búsqueda

- Resolución de colisiones por encadenamiento abierto.
  - El esquema de direccionamiento abierto más simple es conocido como "Intento o exploración lineal":
    - $h_0 = H(K)$
    - $h_i = (h_0 + i) \text{ MOD } N$, $i = 1.. N-1$
  - La exploración cuadrática es una variante que disminuye el agrupamiento de las entradas alrededor de las claves primarias.
    - $h_0 = H(K)$
    - $hi = (h_0 + i^2) \text{ MOD } N$, $i > 0$
  - Otra opción para la anterior es sumar un valor c primo con N.
    - $h_i = (h_0 + c) \text{ MOD } N$

Algoritmos y Estructuras de Datos 14

14

Hashing, inserción en encadenamiento abierto simple

```
TTablaHASH.Insertar(clave k)
T indica el vector de claves
i = 0
 repetir
   j = h(k,i)
   si vacio(T[j]) entonces
     T[j] = k, devolver j, salir
    sino, i =  i + 1
 hasta que i = m
error "sobrecarga de tabla de hash"
```


15

Hashing, búsqueda en encadenamiento abierto simple

```
TTablaHASH.Buscar(clave k)
// T indica el vector de claves
i = 0
 repetir
   j = h(k,i)
   si T[j] = k entonces
      devolver j, salir
    sino, i =  i + 1
hasta que vacio(T[j]) o i = m
devolver nulo
```


16

Búsqueda

- Resolución de colisiones por encadenamiento abierto lineal, ejemplo.

  N = 9, K = EN, TO, TRE, FIRE, FEM, SEKS, SYV

  H(K) = 2,7,1,8,2,8,1

h_0 = H(K)

h_i = (h_0 - i)

0	FEM
1	TRE
2	EN
3	
4	
5	SYV
6	SEKS
7	TO
8	FIRE

17

```
hash ( 89, 10 ) = 9
hash ( 18, 10 ) = 8
hash ( 49, 10 ) = 9
hash ( 58, 10 ) = 8
hash (  9, 10 ) = 9
```

	89	18	49	58	9
0			49	49	49
1				58	58
2					9
3					
4					
5					
6					
7					
8		18	18	18	18
9	89	89	89	89	89

18

Resolución de colisiones por doble desmenuzamiento

- Aunque un valor fijo de **c** reduce el fenómeno de amontonamiento, podemos mejorar la situación haciendo depender **C** de **K**.
- La idea es utilizar una segunda función de desmenuzamiento **h_2(K),** para calcular el c correspondiente.
- Funciones sugeridas:
  - Si N es primo y **h_1(K)** = **K mod N:**
    - **$h_2(K) = 1+(K \bmod (N-1))$**
    - **$h_2(K) = 1+(K \bmod (N-2))$**

    (N y N-2 parejas de primos 1019 1021)
  - **$h_2(K) = 1+(int(K/N) \bmod (N-2))$**

    (int(K/M) puede estar disponible como subproducto del cálculo de **h_1**

19

Análisis de los métodos de hashing

- Inserción y recuperación: peor caso muy malo.
- Diseñar algoritmos para asegurar que en promedio tengan muy buen rendimiento.
- Todas las claves son equiprobables, y la función de transformación distribuye uniformemente.
- Insertar clave en tabla de tamaño **N** con **k** elementos.
- Probabilidad de encontrar una posición libre la primera vez : **(N-k) / N**.
- Esta es también la probabilidad **p_1** de que sea necesaria una sóla comparación.

20

Búsqueda

- Análisis de los métodos de hashing.
  - La probabilidad de que se requiera una segunda comparación es igual a la probabilidad de una colisión en el primer intento, multiplicada por la probabilidad de hallar una posición libre la siguiente vez.
  - Para obtener la probabilidad p_i de una inserción que requiere **i** exploraciones, operamos como sigue:

$$p_1 = \frac{(N-k)}{N}$$
$$p_2 = \frac{k}{N} * \frac{(N-k)}{N-1}$$
$$p_3 = \frac{k}{N} * \frac{(k-1)}{N-1} * \frac{(N-k)}{N-2}$$
$$\ldots$$
$$p_i = \frac{k}{N} * \frac{(k-1)}{N-1} * \frac{(k-2)}{N-2} * \ldots * \frac{1}{N-k+1}$$

21

Búsqueda

- Análisis de los métodos de hashing.
  - El número esperado **E** de exploraciones requeridas después de insertar la clave **k+1** -ésima será:

$$E_{k+1} = \sum_{i=1}^{k+1} i * p_i = 1 * \frac{(N-k)}{N} + 2 * \frac{k}{N} * \frac{(N-k)}{N-1} + \ldots + (k+1) * \frac{k}{N} * \frac{(k-1)}{N-1} * \frac{(k-2)}{N-2} * \ldots * \frac{1}{N-k+1} = \frac{(N+1)}{N-k+1}$$

El número de búsquedas necesarias para recuperar un elemento es igual al necesario para insertarlo. Calculamos ahora el número promedio **E** de exploraciones necesarias para encontrar una clave aleatoria en la tabla de tamaño **N**, con **M** claves presentes en ella:

$$E = \frac{1}{M} * \sum_{k=1}^{M} E_k = \frac{(N+1)}{M} * \sum_{k=1}^{M} \frac{1}{N-k+2} = \frac{(N+1) * \left(H_{N+1} - H_{N-M+1}\right)}{M}$$

22

Búsqueda

- Análisis de los métodos de hashing.
  - En la fórmula anterior, **H** indica la serie armónica. **H** puede ser aproximada como $H_n = \ln(N) + g$, donde **g** es la constante de Euler. Si hacemos

a = M / (N+1), obtenemos:

$$E = \frac{\left(\ln(N+1) - \ln(N-M+1)\right)}{a} = \frac{\ln\left(\frac{N+1}{N-M+1}\right)}{a} = \frac{-\ln(1-a)}{a}$$

a = **"Factor de Carga",** cociente aproximado de las posiciones ocupadas y libres.
a = 0 indica tabla vacía; **a = N/(N+1)** indica tabla llena.

23

Análisis de los métodos de hashing

- El análisis anterior asumió un método que distribuye las claves uniformemente. En el caso de búsqueda lineal, el número previsto de búsquedas da:

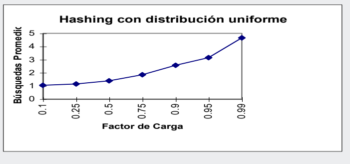
$$E = \frac{\left(1 - \frac{a}{2}\right)}{1 - a}$$

- En la siguiente transparencia podemos observar algunos valores de E y a, para las dos fórmulas vistas.
- En el primer caso, vemos que aún cuando la tabla esté llena en un **90 %**, se precisarán **2.56** búsquedas para encontrar la clave o una posición vacía.
- En el segundo caso, observamos también que los resultados son muy buenos, superiores a la organización en árbol más ajustada.

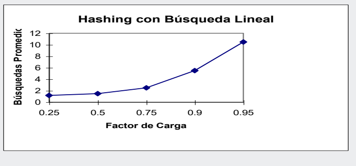
24

Rendimientos del Hashing en función del Factor de Carga.

a	E
0.1	1.05
0.25	1.15
0.5	1.39
0.75	1.85
0.9	2.56
0.95	3.15
0.99	4.66



a	E
0.1	1.06
0.25	1.17
0.5	1.5
0.75	2.5
0.9	5.5
0.95	10.5



25

Hashing, desventajas

- Tamaño fijo de la tabla: es necesaria una buena estimación "a priori" del número de elementos a clasificar.
- En caso de que se conozca el tamaño del conjunto, para lograr un buen rendimiento normalmente se dimensiona la tabla un 10% más grande de lo necesario.
- Si además de insertar y buscar, también es necesario eliminar, estas estructuras son muy ineficientes. La eliminación es un proceso muy difícil, a menos que se utilice encadenamiento directo en un área de desbordamiento independiente. Algunos sistemas operativos utilizan variantes de este método para clasificación externa.

26