Tablas de Hash

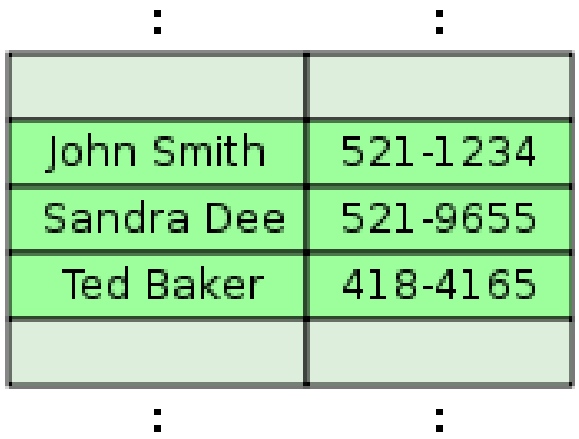
Manuel Guillermo López Buenfil

CIMAT

March 3, 2015

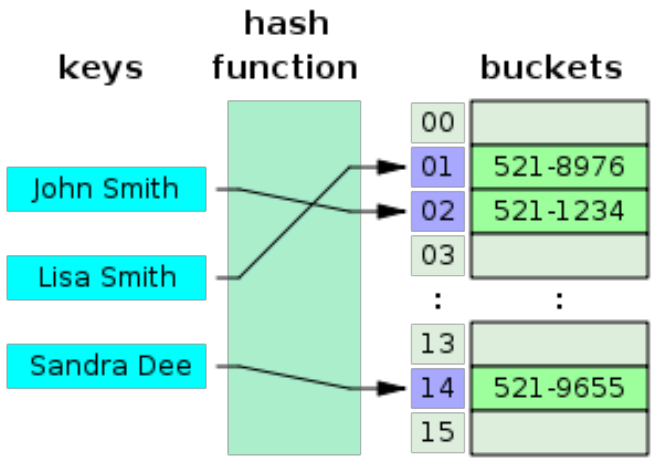
Contenedor asociativo

- Una contenedor asociativo es una estructura de datos que asocia un valor con una llave.
- Soporta dos operaciones primarias:
  1. Insertar un nuevo par en la tabla.
  2. Buscar el valor asociado a una llave:



Tablas Hash

- Si las llaves son enteros pequeños, podemos usar un arreglo para implementar el contenedor asociativo, interpretando la llave como el índice del arreglo.
- Para crear estas llaves, se usa una *función hash*, que transforma las llaves en índices del arreglo.

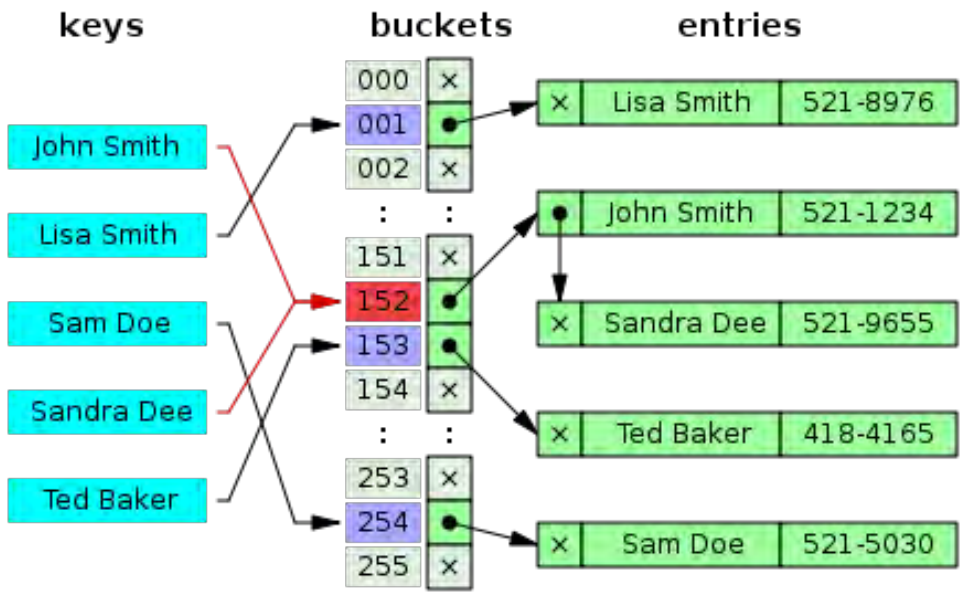


Funciones Hash

- Dado un arreglo de tamaño M, necesitamos una función que transforme una llave en un entero en el rango $[0, M-1]$.
- La función hash debe cumplir tres características:
  1. Determinista
  2. Eficiente para calcular
  3. Distribuir las llaves uniformemente
- Algunos ejemplos:
  - Para enteros positivos, podemos usar la función módulo
  - Para texto, podemos transformar cada caracter a un entero.

Cubetas

- Cada posición del arreglo es una *cubeta*
- Si dos llaves tienen el mismo resultado de la función hash (colisión), ambos entran en la misma cubeta.
- Queremos que las cubetas tengan pocos elementos



Re-hashing

Cuando la cantidad de elementos es mayor a un múltiplo de la cantidad de cubetas (*factor de carga*), podemos hacer un *re-hashing*.

- Primero se incrementa el tamaño del arreglo
- Luego, se re-acomodan todos los valores en el nuevo arreglo

Ventajas y desventajas

Ventajas:

- Acceso promedio en $O(1)$

Desventajas:

- El cálculo de la función hash aumenta el tiempo de acceso por una constante.

STL

Se tienen las siguientes implementaciones de tablas Hash (C++11):

- Dentro de `<unordered_set>`
  - `unordered_set`
  - `unordered_multiset`
- Dentro de `<unordered_map>`
  - `unordered_map`
  - `unordered_multimap`

Ejemplo con unordered_set

```
unordered_set<string> colores = {"amarillo","verde","azul"};
colores.insert ("rojo");
colores.insert ( {"morado","naranja"} );
cout << "El conjunto contiene: \n";
for (auto& x: colores)
    cout<<"\t" << x<<"\n";

cout<<"Contiene azul: "<<colores.count("azul")<<"\n";
cout<<"Contiene blanco: "<<colores.count("blanco")<<"\n";

cout << "size = " << colores.size() << endl;
cout << "bucket_count = " << colores.bucket_count() << endl;
cout << "load_factor = " << colores.load_factor() << endl;
cout << "max_load_factor = " << colores.max_load_factor() << endl;

size_t n = colores.bucket_count();
cout << "El conjunto tiene " << n << " cubetas.\n";
for (size_t i=0; i<n; ++i) {
    cout << "\tLa cubeta #" << i << " tiene:";
    for (auto it = colores.begin(i); it!=colores.end(i); ++it)
        cout << " " << *it;
    cout << "\n";
}
```

```
El conjunto contiene:
        rojo
        morado
        azul
        amarillo
        naranja
        verde
Contiene azul: 1
Contiene blanco: 0
size = 6
bucket_count = 7
load_factor = 0.857143
max_load_factor = 1
El conjunto tiene 7 cubetas.
        La cubeta #0 tiene:
        La cubeta #1 tiene:
        La cubeta #2 tiene:
        La cubeta #3 tiene: rojo
        La cubeta #4 tiene:
        La cubeta #5 tiene: morado azul amarillo
        La cubeta #6 tiene: naranja verde
```

Ejemplo con unordered_map

```
unordered_map<string,string> frutas =
                {{"manzana","rojo"},{"limon","verde"}};
frutas.insert ( {{"uva","morado"},{"guayaba","amarillo"}} );
cout << "El mapa contiene: \n";
for (auto& x: frutas)
    cout<<"\t" << x.first << ": " << x.second << endl;

cout<<"Color de limon: "<<frutas["limon"]<<"\n";

cout << "size = " << frutas.size() << endl;
cout << "bucket_count = " << frutas.bucket_count() << endl;
cout << "load_factor = " << frutas.load_factor() << endl;
cout << "max_load_factor = " << frutas.max_load_factor() << endl;

size_t n = frutas.bucket_count();
cout << "El mapa tiene " << n << " cubetas.\n";
for (size_t i=0; i<n; ++i) {
    cout << "\tLa cubeta #" << i << " tiene:";
    for (auto it = frutas.begin(i); it!=frutas.end(i); ++it)
        cout << "[" << it->first << ":" << it->second << "] ";
    cout << "\n";
}
```

```
El mapa contiene:
        guayaba: amarillo
        uva: morado
        limon: verde
        manzana: rojo
Color de limon: verde
size = 4
bucket_count = 5
load_factor = 0.8
max_load_factor = 1
El mapa tiene 5 cubetas.
        La cubeta #0 tiene:[guayaba:amarillo] [uva:morado]
        La cubeta #1 tiene:[limon:verde]
        La cubeta #2 tiene:
        La cubeta #3 tiene:[manzana:rojo]
        La cubeta #4 tiene:
```