Busqueda Mediante Hashing

Analisis de Algoritmos

Universidad Tecnologica Metropolitana

Daniel Abrilot
John Lopez

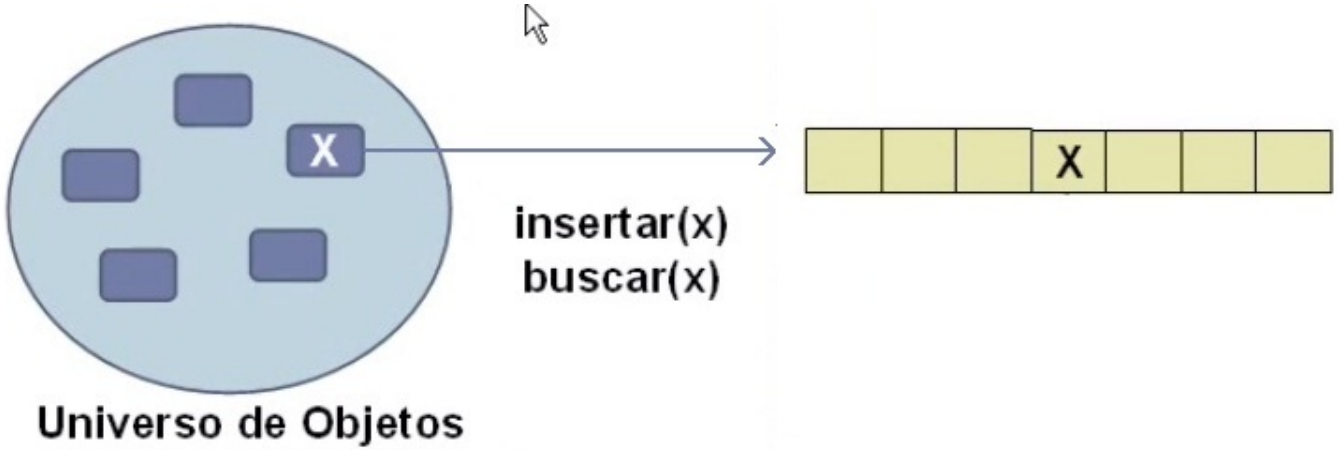
8 de noviembre de 2013

Índice

1. Introduccion y Problematica
2. Tabla Hash
3. Colisiones
4. Conclusiones

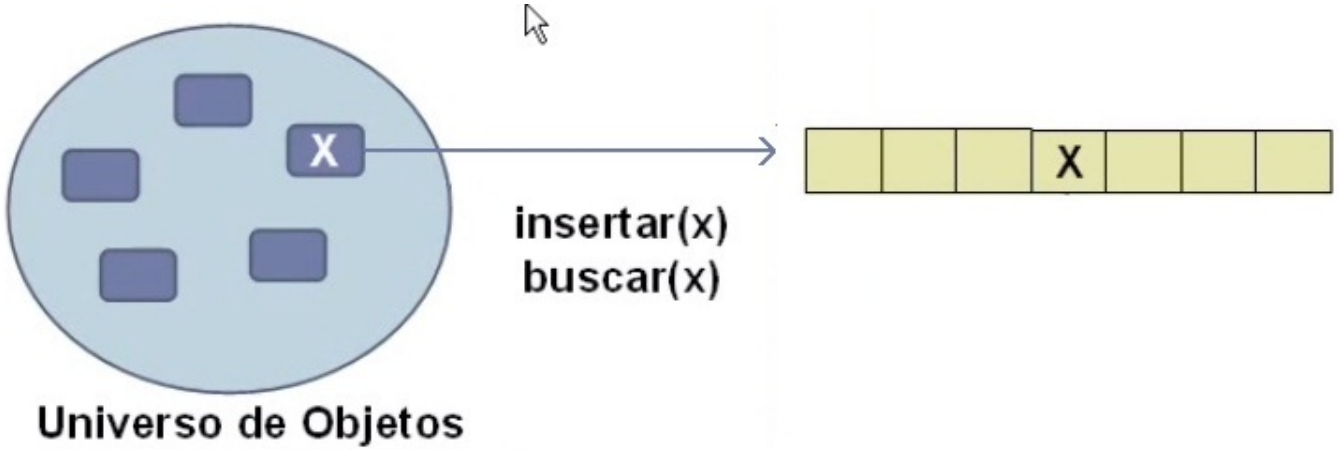
Introduccion

- Utilizar un array para almacenar un grupo de objetos requiere un coste lineal con el numero de elementos para buscar uno de ellos.
- Desconocer el punto exacto de la estructura donde se almacena el objeto buscado umplica realizar una busqueda..
- ¿Existe la forma de saber donde guardar/buscar un elemento?.



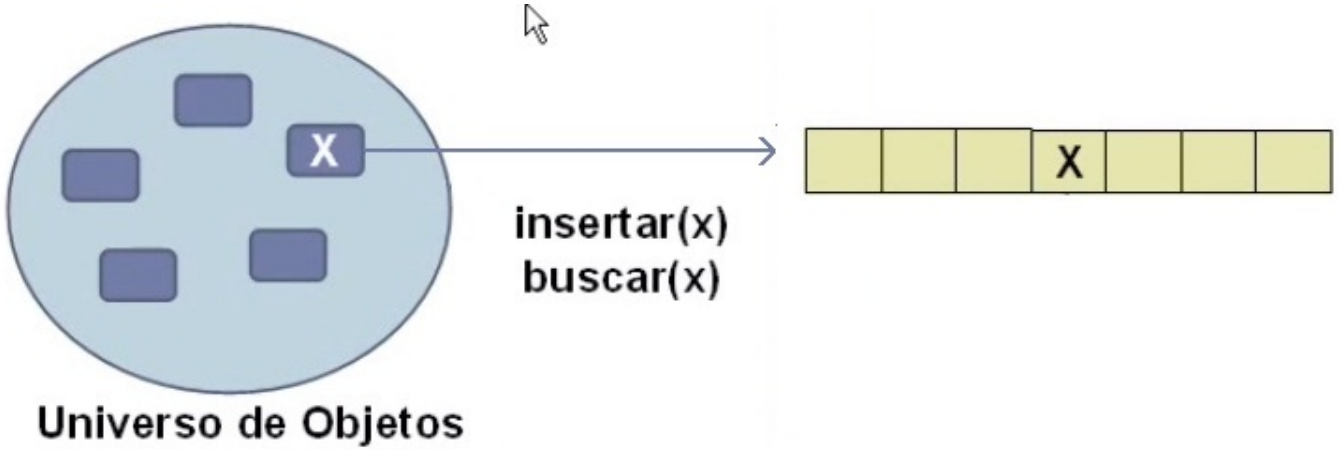
Introduccion

- Utilizar un array para almacenar un grupo de objetos requiere un coste lineal con el numero de elementos para buscar uno de ellos.
- Desconocer el punto exacto de la estructura donde se almacena el objeto buscado umplica realizar una busqueda..
- ¿Existe la forma de saber donde guardar/buscar un elemento?.



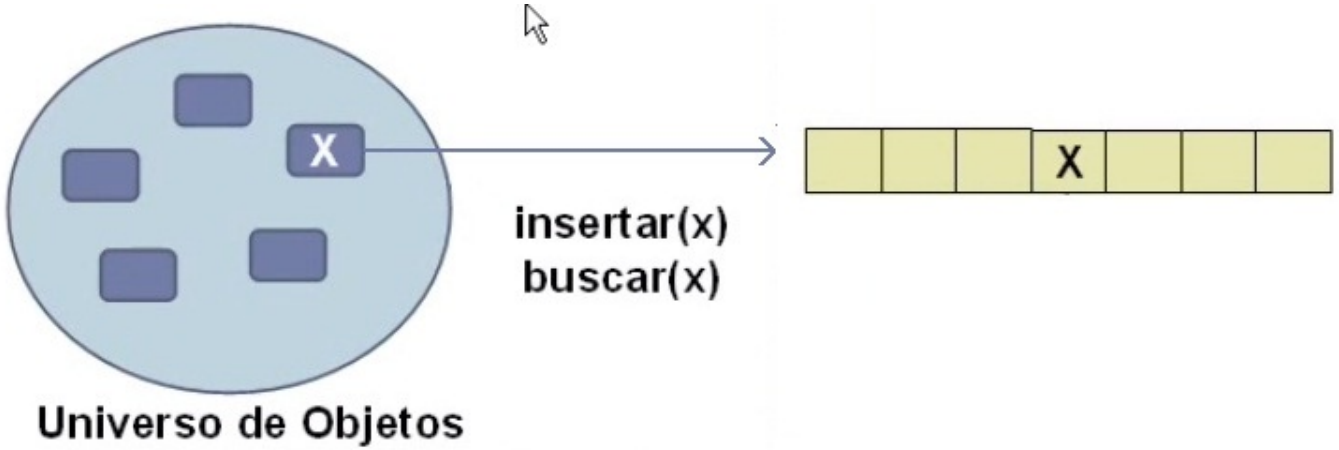
Introduccion

- Utilizar un array para almacenar un grupo de objetos requiere un coste lineal con el numero de elementos para buscar uno de ellos.
- Desconocer el punto exacto de la estructura donde se almacena el objeto buscado umplica realizar una busqueda..
- ¿Existe la forma de saber donde guardar/buscar un elemento?.



Introduccion

- Utilizar un array para almacenar un grupo de objetos requiere un coste lineal con el numero de elementos para buscar uno de ellos.
- Desconocer el punto exacto de la estructura donde se almacena el objeto buscado umplica realizar una busqueda..
- ¿Existe la forma de saber donde guardar/buscar un elemento?.



Problematica Relacionada

- ¿Como se elige la casilla en la que debe ir cada objeto?

Funcion de dispersion asigna un numero o valor hash a un objeto determinado

- $valorHash = hash(x)$.

- ¿Y si el valor hash execede el tamaño del array?

Uso del operador modulo

- $indiceHash = valorHash\%length(array)$. es un entero entre [0,length(array)-1].

- Si dos objetos reciben el mismo valor se produce una colision

Problematica Relacionada

- ¿Como se elige la casilla en la que debe ir cada objeto?

Funcion de dispersion asigna un numero o valor hash a un objeto determinado

- $valorHash = hash(x)$.

- ¿Y si el valor hash execede el tamaño del array?

Uso del operador modulo

- $indiceHash = valorHash\%length(array)$. es un entero entre [0,length(array)-1].

- Si dos objetos reciben el mismo valor se produce una colision

Problematica Relacionada

- ¿Como se elige la casilla en la que debe ir cada objeto?

Funcion de dispersion asigna un numero o valor hash a un objeto determinado

- $valorHash = hash(x)$.

- ¿Y si el valor hash execede el tamaño del array?

Uso del operador modulo

- $indiceHash = valorHash\%length(array)$. es un entero entre [0,length(array)-1].

- Si dos objetos reciben el mismo valor se produce una colision

Problematica Relacionada

- ¿Como se elige la casilla en la que debe ir cada objeto?

Funcion de dispersion asigna un numero o valor hash a un objeto determinado

- $valorHash = hash(x)$.

- ¿Y si el valor hash execede el tamaño del array?

Uso del operador modulo

- $indiceHash = valorHash\%length(array)$. es un entero entre [0,length(array)-1].

- Si dos objetos reciben el mismo valor se produce una colision

Problematica Relacionada

- ¿Como se elige la casilla en la que debe ir cada objeto?

Funcion de dispersion asigna un numero o valor hash a un objeto determinado

- $valorHash = hash(x)$.

- ¿Y si el valor hash execede el tamaño del array?

Uso del operador modulo

- $indiceHash = valorHash\%length(array)$. es un entero entre [0,length(array)-1].

- Si dos objetos reciben el mismo valor se produce una colision

Problematica Relacionada

- ¿Como se elige la casilla en la que debe ir cada objeto?

Funcion de dispersion asigna un numero o valor hash a un objeto determinado

- $valorHash = hash(x)$.

- ¿Y si el valor hash execede el tamaño del array?

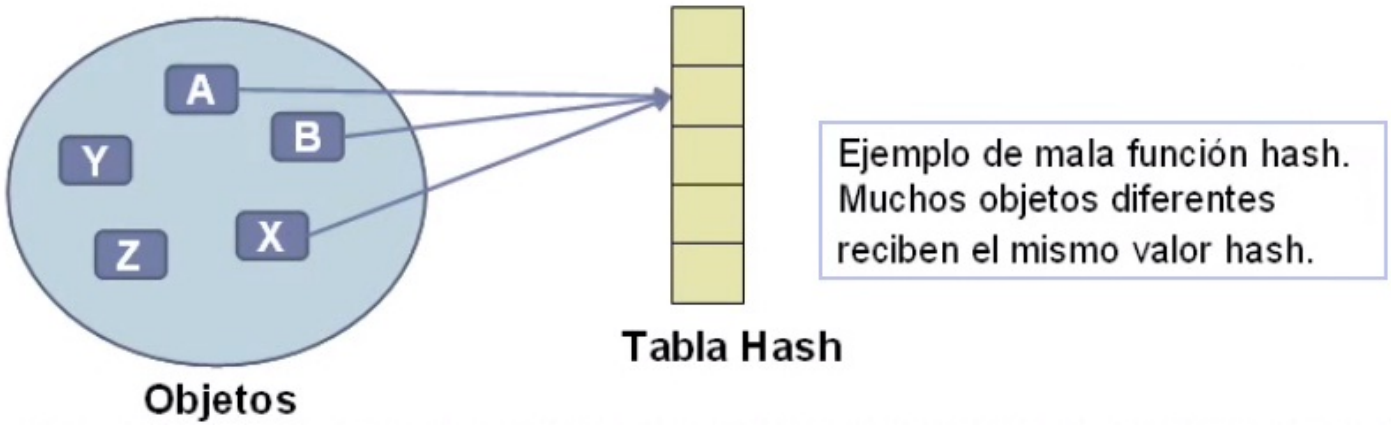
Uso del operador modulo

- $indiceHash = valorHash\%length(array)$. es un entero entre [0,length(array)-1].

- Si dos objetos reciben el mismo valor se produce una colision

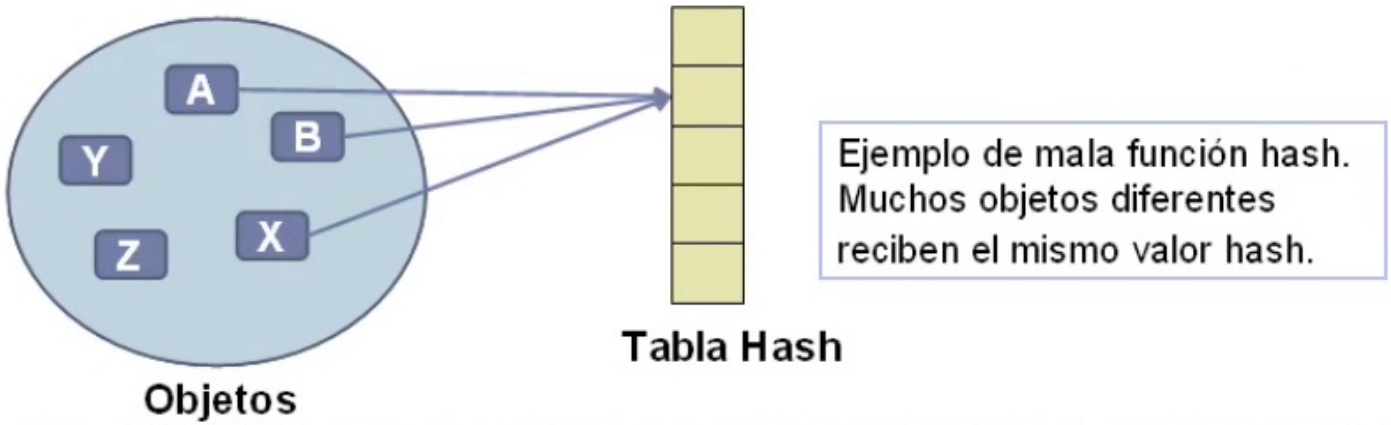
La Tabla Hash (I)

- Una tabla Hash es una estructura de datos que pretende la insercion, busqueda y borrado de elementos en un tiempo constante.
- Para ello la funcion de dispercion (hash(x)) distribuira los objetos de forma uniforma a lo largo de la tabla



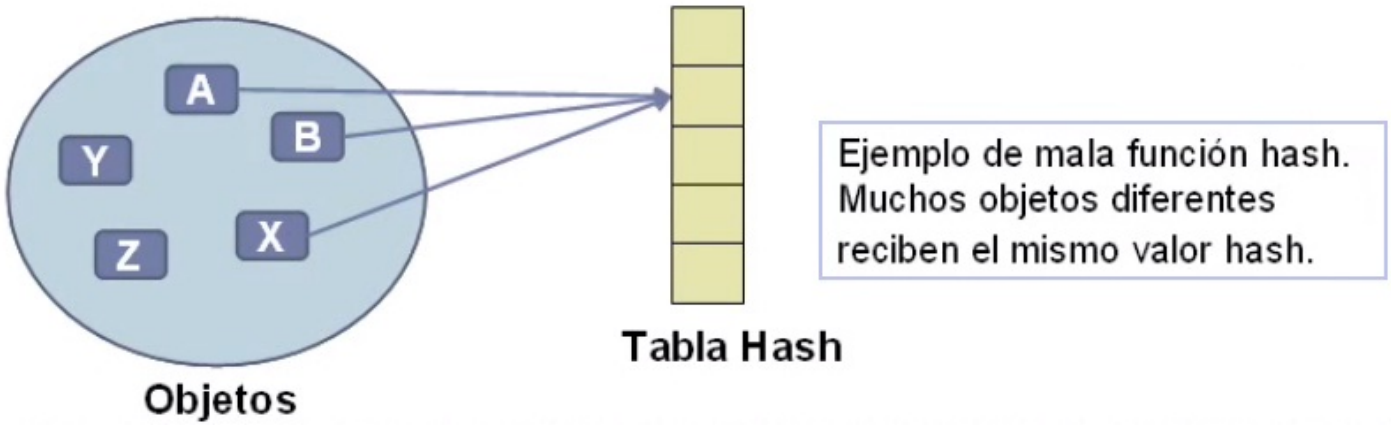
La Tabla Hash (I)

- Una tabla Hash es una estructura de datos que pretende la insercion, busqueda y borrado de elementos en un tiempo constante.
- Para ello la funcion de dispercion (hash(x)) distribuira los objetos de forma uniforma a lo largo de la tabla



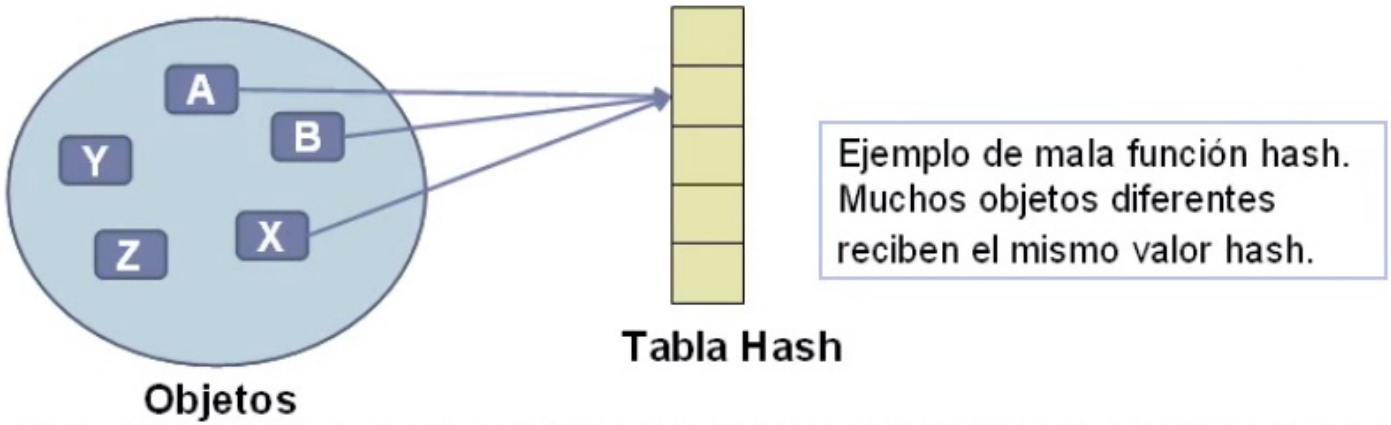
La Tabla Hash (I)

- Una tabla Hash es una estructura de datos que pretende la insercion, busqueda y borrado de elementos en un tiempo constante.
- Para ello la funcion de dispercion (hash(x)) distribuira los objetos de forma uniforma a lo largo de la tabla



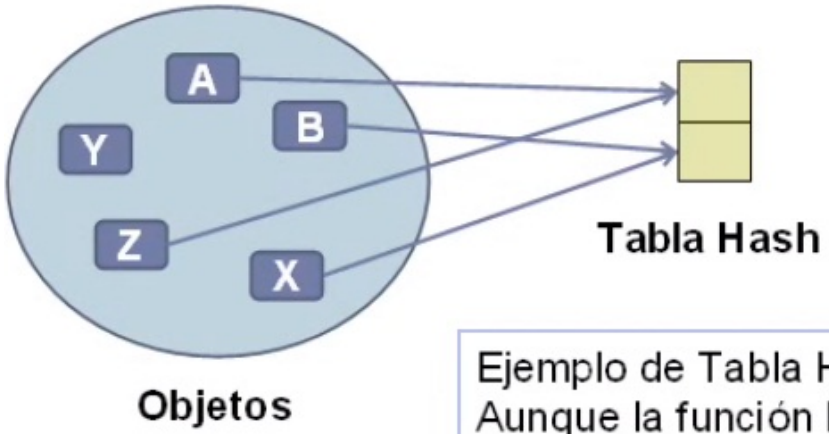
La Tabla Hash (I)

- Una tabla Hash es una estructura de datos que pretende la insercion, busqueda y borrado de elementos en un tiempo constante.
- Para ello la funcion de dispercion (hash(x)) distribuira los objetos de forma uniforma a lo largo de la tabla



La Tabla Hash (II)

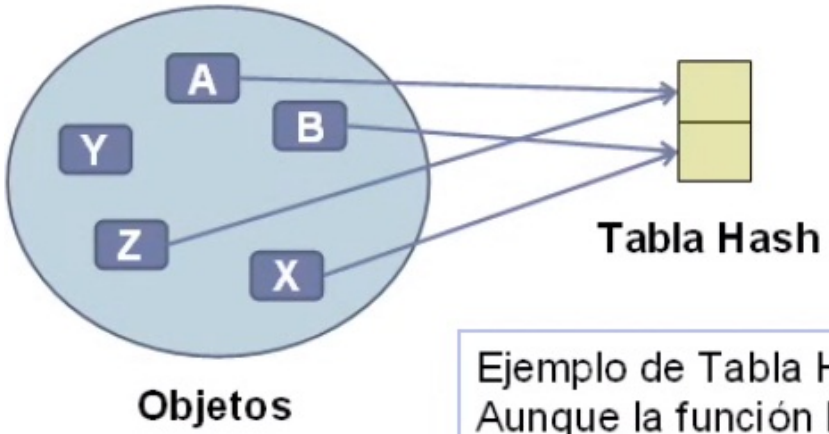
- La Tabla Hash debe tener un tamaño relativamente grande para reducir el numero de colisiones



Ejemplo de Tabla Hash de tamaño insuficiente. Aunque la función hash distribuya bien los objetos, el tamaño de la tabla afecta directamente a la probabilidad de colisión.

La Tabla Hash (II)

- La Tabla Hash debe tener un tamaño relativamente grande para reducir el numero de colisiones



Ejemplo de Tabla Hash de tamaño insuficiente. Aunque la función hash distribuya bien los objetos, el tamaño de la tabla afecta directamente a la probabilidad de colisión.

Desventajas de la Tabla Hash

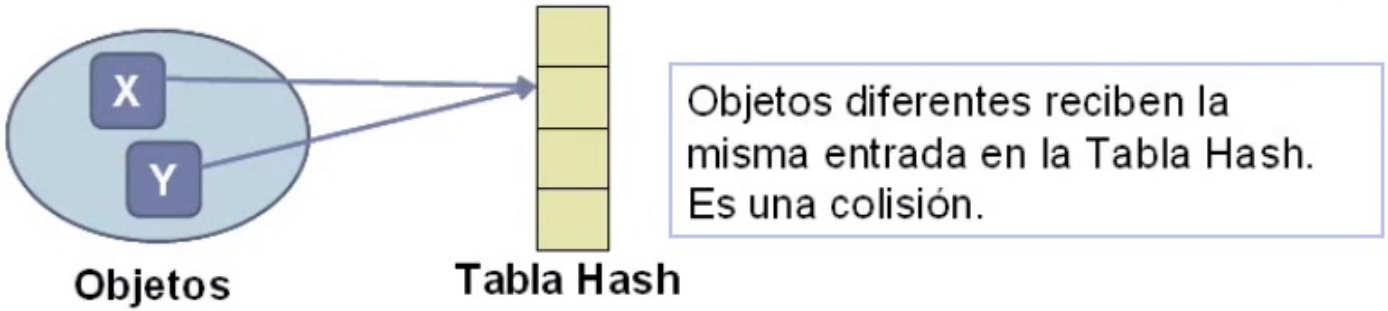
- La Tabla Hash permite realizar las operaciones de busqueda, insercion y borrado pero existen algunas desventajas.
  - Almacenar N datos requiere de una Tabla Hash de tamaño M >> N, para reducir el numero de colisiones.
  - No es posible obtener una secuencia ordenada de sus elementos sin que se genere un coste de tiempo no constante.

Desventajas de la Tabla Hash

- La Tabla Hash permite realizar las operaciones de busqueda, insercion y borrado pero existen algunas desventajas.
  - Almacenar N datos requiere de una Tabla Hash de tamaño M >> N, para reducir el numero de colisiones.
  - No es posible obtener una secuencia ordenada de sus elementos sin que se genere un coste de tiempo no constante.

Coliciones : Definicion y Gestion

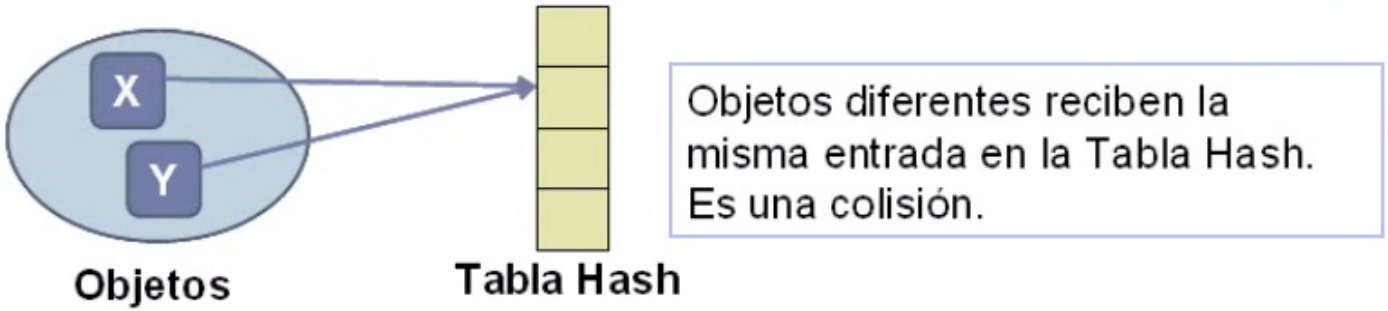
- Una colicion se produce cuando objetos diferentes dan lugar al mismo indice hash
- Objetos o1 y o2, donde: $o1 != o2 \&\& hash(o1) = hash(02)$



- Implica un sobrecoste controlar las casillas ocupadas/libres.
- Es complicado evitar colisiones.

Coliciones : Definicion y Gestion

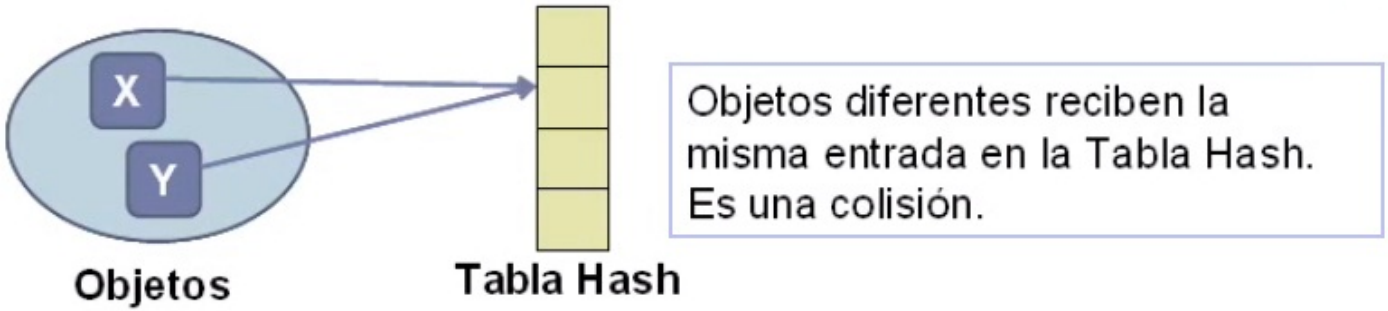
- Una colicion se produce cuando objetos diferentes dan lugar al mismo indice hash
- Objetos o1 y o2, donde: $o1 != o2 \&\& hash(o1) = hash(02)$



- Implica un sobrecoste controlar las casillas ocupadas/libres.
- Es complicado evitar colisiones.

Coliciones : Definicion y Gestion

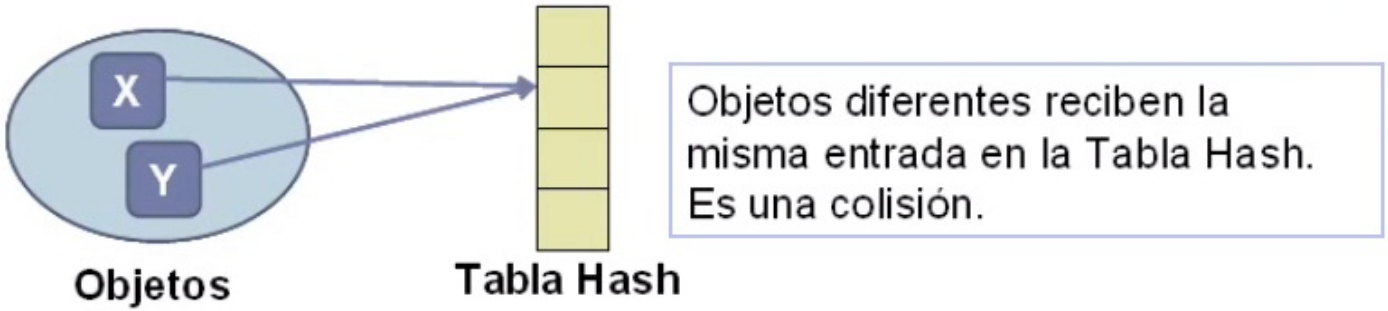
- Una colicion se produce cuando objetos diferentes dan lugar al mismo indice hash
- Objetos o1 y o2, donde: $o1 != o2 \&\& hash(o1) = hash(02)$



- Implica un sobrecoste controlar las casillas ocupadas/libres.
- Es complicado evitar colisiones.

Coliciones : Definicion y Gestion

- Una colicion se produce cuando objetos diferentes dan lugar al mismo indice hash
- Objetos o1 y o2, donde: $o1 != o2 \&\& hash(o1) = hash(02)$



- Implica un sobrecoste controlar las casillas ocupadas/libres.
- Es complicado evitar colisiones.

Formas de Resolver las Colisiones

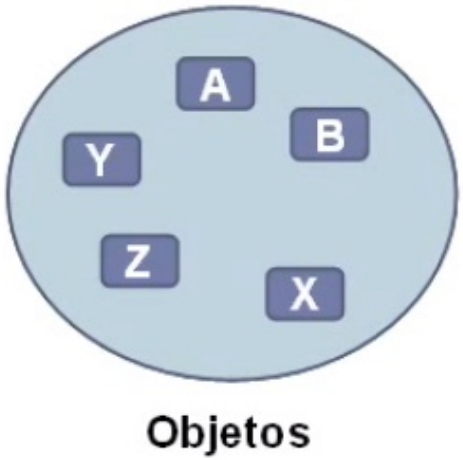
- Resolucion Mediante Exploracion : Trata de buscar otra posicion libre en la tabla para albergar la insercion del elemento.
  - Exploracion Lineal : visita la siguiente casilla, si esta libre realiza la insercion sino visita la siguiente.
  - Exploracion Cuadratica : visita la casilla i^2 posiciones.si esta libre realiza la insercion sino visita la siguiente.
- Resolucion Mediante Encadenamiento Enlazado: En cada posicion de la tabla se mantien una lista enlazada en la que se van insertando los elementos cuyo valor hash les asigna la misma posicion.

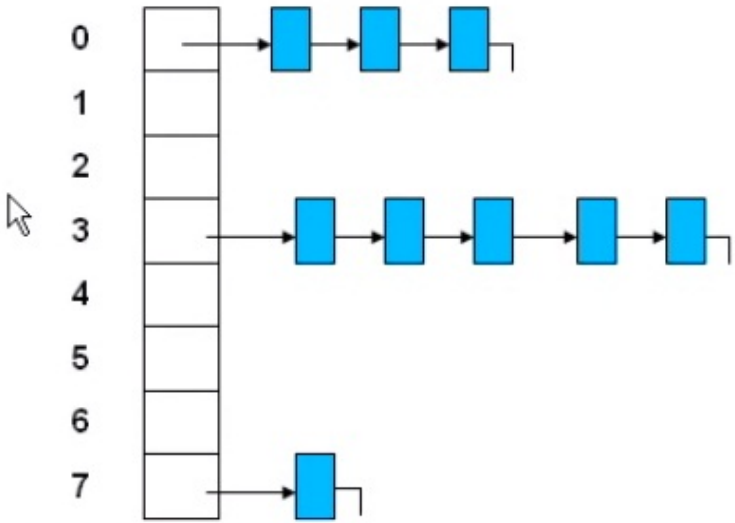
Formas de Resolver las Colisiones

- Resolucion Mediante Exploracion : Trata de buscar otra posicion libre en la tabla para albergar la insercion del elemento.
  - Exploracion Lineal : visita la siguiente casilla, si esta libre realiza la insercion sino visita la siguiente.
  - Exploracion Cuadratica : visita la casilla i^2 posiciones.si esta libre realiza la insercion sino visita la siguiente.
- Resolucion Mediante Encadenamiento Enlazado: En cada posicion de la tabla se mantien una lista enlazada en la que se van insertando los elementos cuyo valor hash les asigna la misma posicion.

Hashing Enlazado

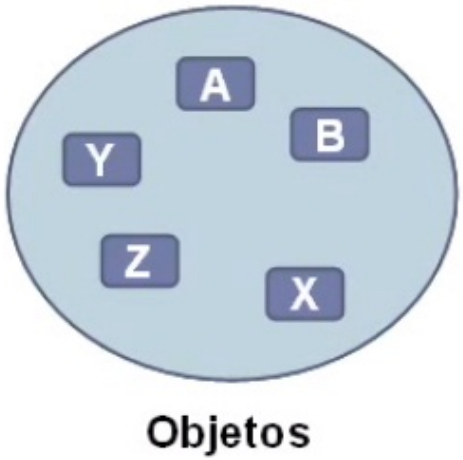
- El Hashing Enlazado usa un vector de Listas Enlazadas.
  - Aquellos objetos que reciban un determinado valor de Hash, se insertaran en la lista enlazada correspondiente.

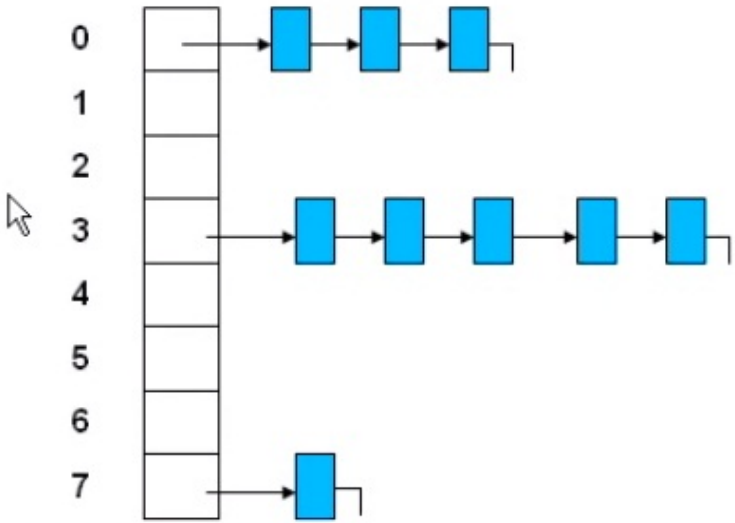




Hashing Enlazado

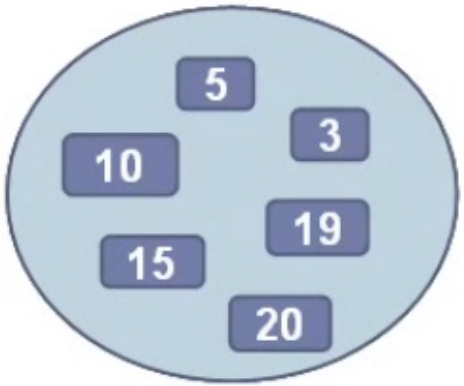
- El Hashing Enlazado usa un vector de Listas Enlazadas.
  - Aquellos objetos que reciban un determinado valor de Hash, se insertaran en la lista enlazada correspondiente.





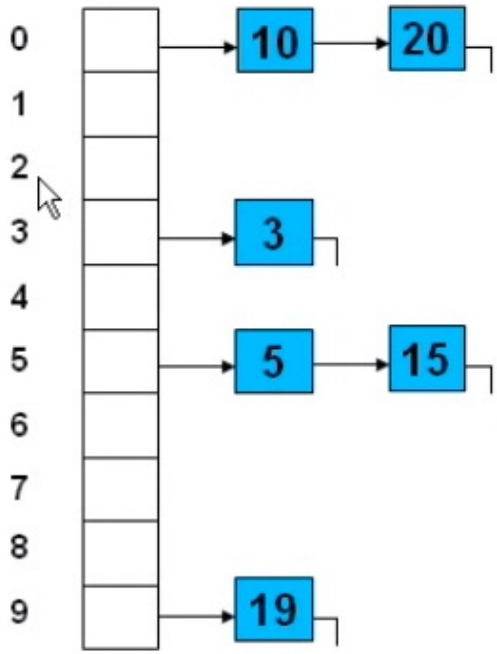
Ejemplo de uso de la Tabla Hash

- Insertar los elementos {5,10,3,15,20,19} en una Tabla Hash de tamaño 10 y con funcion hash:
  - hash (x) = x % 10



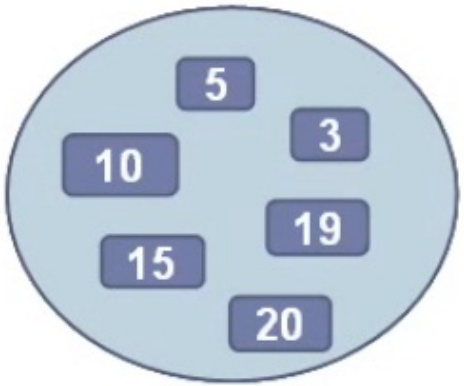
Objetos

- Factor de Carga
  - Elementos / Capacidad = 6/10



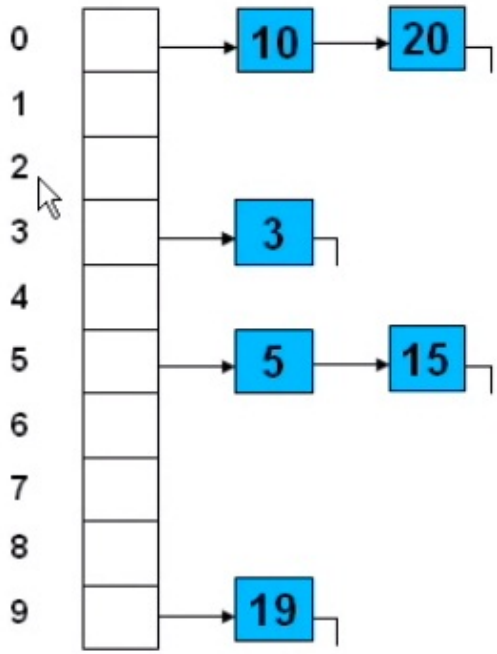
Ejemplo de uso de la Tabla Hash

- Insertar los elementos {5,10,3,15,20,19} en una Tabla Hash de tamaño 10 y con funcion hash:
  - hash (x) = x % 10



Objetos

- Factor de Carga
  - Elementos / Capacidad = 6/10



Rehashing en una Tabla Hash

- Si el Factor de Carga (grado de ocupacion) de una Tabla Hash es alto (>75%), el numero de colisiones aumenta significativamente.
- Solucion : Rehashing
  - Aumentar el tamaño de la tabla para reducir el factor de carga.
- Ventajas
  - Permite mantener un reducido factor de carga para que las principales operaciones (insertar, buscar, eliminar) se realicen en tiempo constante.
- Inconvenientes
  - Requiere construir un nuevo array e insertar nuevamente todos los datos

Rehashing en una Tabla Hash

- Si el Factor de Carga (grado de ocupacion) de una Tabla Hash es alto (>75%), el numero de colisiones aumenta significativamente.
- Solucion : Rehashing
  - Aumentar el tamaño de la tabla para reducir el factor de carga.
- Ventajas
  - Permite mantener un reducido factor de carga para que las principales operaciones (insertar, buscar, eliminar) se realicen en tiempo constante.
- Inconvenientes
  - Requiere construir un nuevo array e insertar nuevamente todos los datos

Rehashing en una Tabla Hash

- Si el Factor de Carga (grado de ocupacion) de una Tabla Hash es alto (>75%), el numero de colisiones aumenta significativamente.
- Solucion : Rehashing
  - Aumentar el tamaño de la tabla para reducir el factor de carga.
- Ventajas
  - Permite mantener un reducido factor de carga para que las principales operaciones (insertar, buscar, eliminar) se realicen en tiempo constante.
- Inconvenientes
  - Requiere construir un nuevo array e insertar nuevamente todos los datos

Rehashing en una Tabla Hash

- Si el Factor de Carga (grado de ocupacion) de una Tabla Hash es alto (>75%), el numero de colisiones aumenta significativamente.
- Solucion : Rehashing
  - Aumentar el tamaño de la tabla para reducir el factor de carga.
- Ventajas
  - Permite mantener un reducido factor de carga para que las principales operaciones (insertar, buscar, eliminar) se realicen en tiempo constante.
- Inconvenientes
  - Requiere construir un nuevo array e insertar nuevamente todos los datos

Rehashing en una Tabla Hash

- Si el Factor de Carga (grado de ocupacion) de una Tabla Hash es alto (>75%), el numero de colisiones aumenta significativamente.
- Solucion : Rehashing
  - Aumentar el tamaño de la tabla para reducir el factor de carga.
- Ventajas
  - Permite mantener un reducido factor de carga para que las principales operaciones (insertar, buscar, eliminar) se realicen en tiempo constante.
- Inconvenientes
  - Requiere construir un nuevo array e insertar nuevamente todos los datos

Busqueda Mediante Hashing

- Permite aumentar la velocidad de búsqueda sin necesidad de tener los elementos ordenados.
- Cuenta también con la ventaja de que el tiempo de búsqueda es prácticamente independiente del número de componentes del arreglo.
- Para realizar la busqueda mediante este metodo bastara con conocer la clave asignada, por la funcion hash, a cada elemento.
- Al contrario de una busqueda convencional no debera recorrer el arreglo casilla por casilla, obtendremos el elemento directamente ingresando la clave.

Busqueda Mediante Hashing

- Permite aumentar la velocidad de búsqueda sin necesidad de tener los elementos ordenados.
- Cuenta también con la ventaja de que el tiempo de búsqueda es prácticamente independiente del número de componentes del arreglo.
- Para realizar la busqueda mediante este metodo bastara con conocer la clave asignada, por la funcion hash, a cada elemento.
- Al contrario de una busqueda convencional no debera recorrer el arreglo casilla por casilla, obtendremos el elemento directamente ingresando la clave.

Busqueda Mediante Hashing

- Permite aumentar la velocidad de búsqueda sin necesidad de tener los elementos ordenados.
- Cuenta también con la ventaja de que el tiempo de búsqueda es prácticamente independiente del número de componentes del arreglo.
- Para realizar la busqueda mediante este metodo bastara con conocer la clave asignada, por la funcion hash, a cada elemento.
- Al contrario de una busqueda convencional no debera recorrer el arreglo casilla por casilla, obtendremos el elemento directamente ingresando la clave.

Busqueda Mediante Hashing

- Permite aumentar la velocidad de búsqueda sin necesidad de tener los elementos ordenados.
- Cuenta también con la ventaja de que el tiempo de búsqueda es prácticamente independiente del número de componentes del arreglo.
- Para realizar la busqueda mediante este metodo bastara con conocer la clave asignada, por la funcion hash, a cada elemento.
- Al contrario de una busqueda convencional no debera recorrer el arreglo casilla por casilla, obtendremos el elemento directamente ingresando la clave.

Busqueda Mediante Hashing

- Permite aumentar la velocidad de búsqueda sin necesidad de tener los elementos ordenados.
- Cuenta también con la ventaja de que el tiempo de búsqueda es prácticamente independiente del número de componentes del arreglo.
- Para realizar la busqueda mediante este metodo bastara con conocer la clave asignada, por la funcion hash, a cada elemento.
- Al contrario de una busqueda convencional no debera recorrer el arreglo casilla por casilla, obtendremos el elemento directamente ingresando la clave.

Conclusiones

- La Tabla Hash permite que el coste medio de las operaciones insertar, buscar y eliminar sea constante.
- Hay que elegir correctamente la funcione de hash:
  - Debe ser facilmente calculable, sin costosas operaciones.
  - Debe tener una buena distribucion de valores entre todas las componentes de la tabla.
- Ejemplos de Aplicacion de Tablas de Dispersion:
  - Corrector Ortografico.
  - Tablas de Simbolos de un Compilador.
  - Diccionarios.
  - etc.

Conclusiones

- La Tabla Hash permite que el coste medio de las operaciones insertar, buscar y eliminar sea constante.
- Hay que elegir correctamente la funcione de hash:
  - Debe ser facilmente calculable, sin costosas operaciones.
  - Debe tener una buena distribucion de valores entre todas las componentes de la tabla.
- Ejemplos de Aplicacion de Tablas de Dispersion:
  - Corrector Ortografico.
  - Tablas de Simbolos de un Compilador.
  - Diccionarios.
  - etc.

Conclusiones

- La Tabla Hash permite que el coste medio de las operaciones insertar, buscar y eliminar sea constante.
- Hay que elegir correctamente la funcione de hash:
  - Debe ser facilmente calculable, sin costosas operaciones.
  - Debe tener una buena distribucion de valores entre todas las componentes de la tabla.
- Ejemplos de Aplicacion de Tablas de Dispersion:
  - Corrector Ortografico.
  - Tablas de Simbolos de un Compilador.
  - Diccionarios.
  - etc.

Conclusiones

- La Tabla Hash permite que el coste medio de las operaciones insertar, buscar y eliminar sea constante.
- Hay que elegir correctamente la funcione de hash:
  - Debe ser facilmente calculable, sin costosas operaciones.
  - Debe tener una buena distribucion de valores entre todas las componentes de la tabla.
- Ejemplos de Aplicacion de Tablas de Dispersion:
  - Corrector Ortografico.
  - Tablas de Simbolos de un Compilador.
  - Diccionarios.
  - etc.

Bibliografia I


Paginas de Referencia.


http://www.youtube.com/watch?v=WnLdu8OHA3Q


http://www.hci.uniovi.es/Products/DSTool/hash/hash-queSon.html


http://juan-pato.blogspot.com/2009/05/metodo-de-busqueda-hashing.html