Datu egiturak

Hash taulak

Aurkibidea

- ***Zer dira?***
  - Zergatik dira garrantzitsuak
  - Zertarako erabiltzen dira
  - Ezaugarriak
- ***Metodo nagusiak***
- ***Inplementazio adibideak***
  - Hash taula itxia
  - Hash taula irekia

Hash taulak

Zergatik dira garrantzitsuak: Helburuak

Elementu talde batean, oinarrizko eragiketak era eraginkorrenean exekutatzea dira:

- **Bilaketa**
- **Ezabaketa**
- **Txertaketa**

Hash taulen helburua eragiketa guztiak **orden konstantean** O(1)

Hash Taulak

Zer dira? Zertarako erabiltzen dira?

- Datu egitura honek elementu multzo bat indexatzen du funtzio bat erabiliz (hash funtzioa)

- Array-a da erabiltzen den datu egitura

- Adibideak:
  - Hiztegi baten hitzak gordetzeko
  - Posible diren datu kopuru handi batetik multzo murritz bat gordetzeko
    - Enpresa baten langileak
    - Banku baten kontuak

Hash taulak

Zer dira? Zertarako erabiltzen dira?

(katua,cat)
(txakurra,dog)
(etxea,house)
(kaixo,hello)

HASH FUNTZIOA

f(String)=n

HASH TAULA

1	
2	(katua,cat)
3	
4	
5	(txakurra,dog)
6	
7	(etxea,house)
8	
9	
10	(kaixo,hello)
11	
12	

Adibidea:

f(katua)=2
f(txakurra)=5
f(etxea)=7

Hash taulak
hash funtzioa

- Elementuaren gakoa erabiliz array-aren indize bat itzultzen du

  f(gakoa) = Array posizioa

- Ezaugarriak (desiragarriak)
  - Elementu bakoitza posizio desberdinean
  - Konplexutasun orden konstantea

Hash taulak

Metodo nagusiak (interfazea)

```
public interface Map<K, V>
```

Metodoak	*Esanahia*
V **put**(K key,V value)	*key* gakoa duen *value* elementua txertatzen du
V **find**(K key)	*key* gakoa duen elementua aurkitzen du
V **remove**(K key)	*key* gakoa duen elementua ezabatzen du bere balio itzuliz
void **modify**(K key, V value)	*key* elementuaren edukia aldatzen du
boolean **exists**(K key)	*true*, *key* gakoa duen elementua existitzen bada
int **numPos**()	Taularen elementu kopurua itzultzen du

Hash taulak

Metodo nagusiak (interfazea)

public interface Map<K,V>

Metodoak	***Esanahia***
V **put** (K key, V value)	Gakoa eta balio txertatzen du hash taulan.Gakoa badago, balio zaharra aldatzen du balio berriarengatik. Kasu honetan balio zaharra itzultzen du. Gakoa ez badago ***null*** itzultzen du.
V **get** (K key)	Gakoari dagokion balio itzultzen du, edo ***null*** ez badago.
V **remove**(K key)	Gakoa eta bere balioa ezabatzen du. Gakoari dagokion balioa itzultzen du edo ***null*** existitzen ez bada.
void **clear** ()	Elementu guztiak ezabatzen ditu,
boolean **isEmpty**()	True hutsa bada, edo false beste kasuan.
boolean **containsKey** (K key)	True gakoa taulan badago, false beste kasuan.
int **size** ()	Zerrendaren elementu kopurua itzultzen du

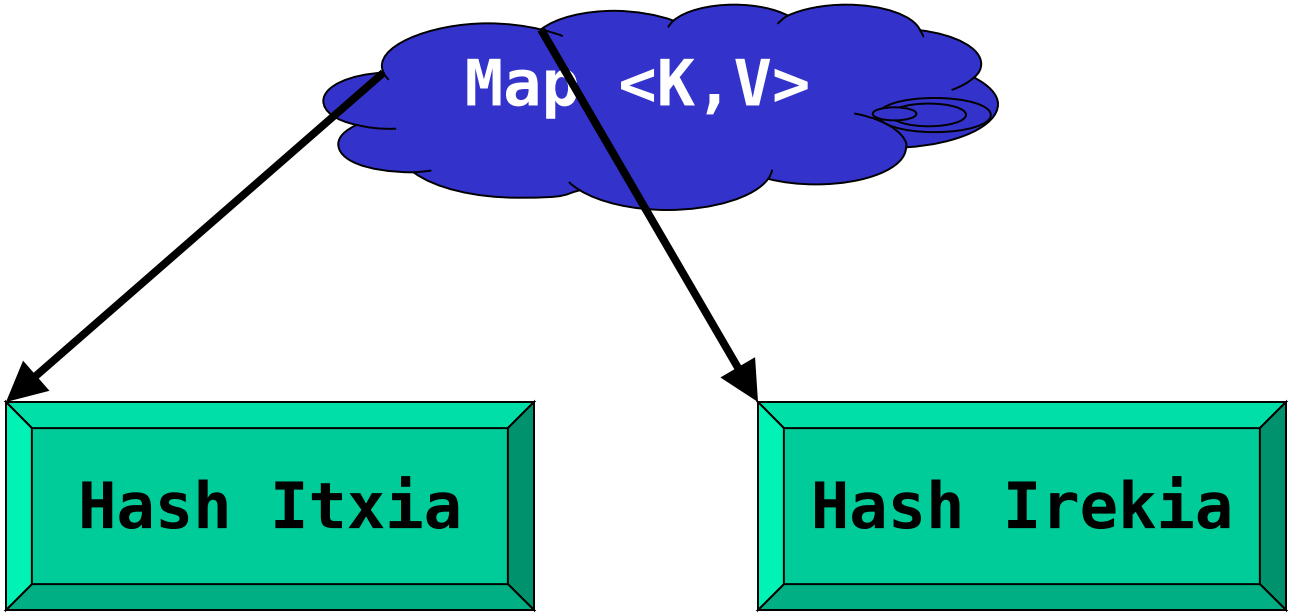
Informazio osagarria:
http://java.sun.com/j2se/1.5.0/docs/api/java/util/Map.html#get(java.lang.Object)

Hash taulak: interfazea

```
public interface Map <K,V> {
    public V put (K key, V value);
    public V get (K key);
    public V remove (K key);
    public void clear();
    public boolean isEmpty();
    public boolean containsKey(K key);
    public int size();
}
```

Hash taulak

Interfaze bat, inplementazio desberdin



Hash taulak

Hash itxia

- Hash taula bat itxia da, onartzen dituen gehienezko elementu kopurua finkatuta dagoenean (MAXPOS)
- Hash funtzioak gako bat hartuta zenbaki positibo bat lortu behar du:
  - hash(“JAVA”)=10+1+23+1=35
  - hash(“15564465”)=36
- Eraginkortasuna bermatzeko taularen tamaina gorde nahi diren baino %25 elementu osagarrirekin sortzea gomendatzen da

Hash taulak

Hash itxia

- Hash funtzioaren emaitzarekin gakoari taulan dagokion posizioa lortzen da
  - indize=hash(gakoa)%MAXPOS
  - Baldin MAXPOS=20
    - 15564465 gakoaren posizioa zein litzateke?
    - “JAVA” gakoaren posizioa zein litzateke?

Hash taulak
Hash itxia

- Zer gertatuko litzateke posizio hori beste gako sinonimo* batek beteko balu (kolisioa)? ➔
  - ***Lehenbailehen libre dagoen posizio bat aurkitu taulan!!***

- ***“CC” gakoa “ADA” gakoaren sinonimoa da ikusitako hash funtzioarekin***

Hash taulak

Kolisioa: Nola aurkitu posizio berria?

- Proba lineala: ondoren dagoen lehen posizio libre bilatu
  - ***while (gako posizioa erabilita) do***
    gakoa := (1 + gakoa) % MAXPOS

- Zenbaki lehenaren proba: MAXPOS zatitzen EZ duen zenbaki lehen bat aukeratu (MAXPOS=15 → P = 13)
  gakoa:= (P + gakoa) % MAXPOS

HASH TAULA

1	
2	(katua,cat)
3	
4	
5	(txakurra,dog)
6	
7	(etxea,house)
8	
9	
10	(kaixo,hello)
11	
12	

Hash taulak:
Proba linealaren inplementazioa: datuak eta taula

```
class DataItem<K,V> {
      public K key;                 // data item (key)
      public V data;
      public DataItem(K k, V v) { // eraikitzailea
        key = k;
        data = v; }
   }  // end class DataItem
```

```
class HashTableItxia<K,V> implements Map<K,V> {
   DataItem<K,V>[] hashArray;        //hash taula egitura
   int arraySize;
   DataItem<K,V> nonItem;
```

Hash taulak:
Eraikitzailea eta hash funtzioa

```
public HashTableItxia(int size)  {     // eraikitzailea
      arraySize = size;
      hashArray = new DataItem<K,V>[arraySize];
      nonItem = new DataItem(null,null);   // elementu ezabatua
}
```

```
public int hashFunc(K  key) {
      return (key.hashCode() % arraysize );
                                    // Objektu bakoitzak kode desberdin bat
                                   // dauka, sistemak emandakoa
      }
```

Hash taulak:
containsKey metodoa. Proba lineala

```
public boolean containsKey (K pkey) { // find item with key
      int hashVal = hashFunc(pkey);  // hash the key
      while( (hashArray[hashVal] != null)  &&
         (!hashArray[hashVal].key.equals(pkey)) ) {// not found
               hashVal++;                 // go to next cell
               hashVal = hashVal % arraySize; // wraparound if necessary
               }
      if(hashArray[hashVal] == null)
            return false;   // ez dugu aurkitu
      else
            return true;
   }
```

Hash taulak:
remove metodoa. Proba lineala

```
public V remove  (K pKey) { // delete a DataItem
      int hashVal = hashFunc(pKey);  // hash the key

      while( (hashArray[hashVal] != null)  &&
            (!hashArray[hashVal].key.equals(pKey)) ) {
                  hashVal++;                  // proba lineala
                  hashVal = hashVal % arraySize; }

     if((hashArray[hashVal] == null) )
                   return null;
     else {
        V temp = hashArray[hashVal].data; // save item
        hashArray[hashVal] = nonItem;       // delete item
        return temp; }
}
```

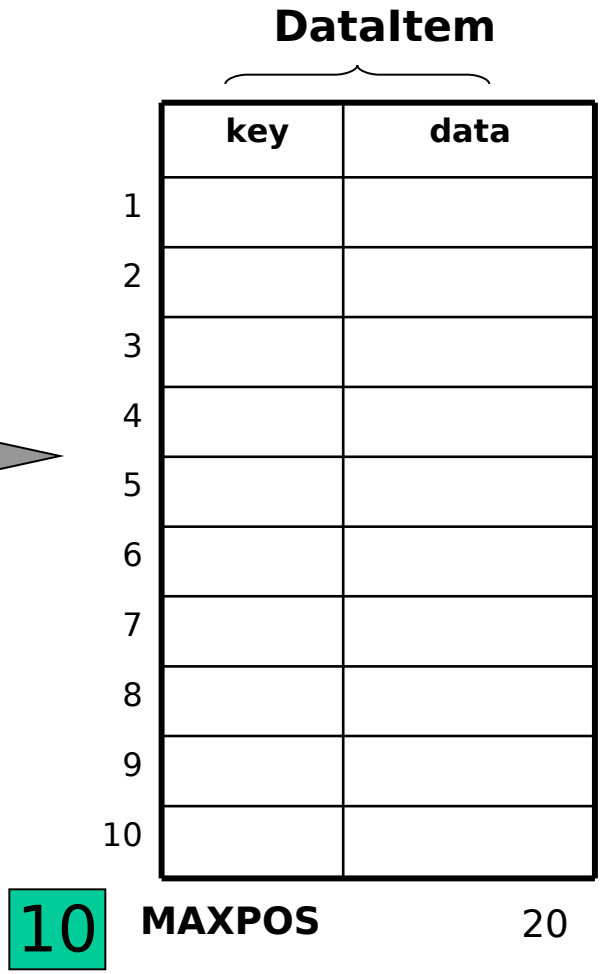
Hash Taula Itxiak:

Put metodoa (Proba lineala)

```
public V put (K pKey, V pData) {
      int hashVal = hashFunc(pKey);
      if (!this.containsKey(pKey)){
            while((hashArray[hashVal] != null) &&
                  (hashArray[hashVal] != nonItem)){
               hashVal= hashVal + 1;
                hashVal = hashVal % arraySize; }
            hashArray[hashVal] = new DataItem<K,V>(pKey,
pData);
            return null; }
            else{
            while(!hashArray[hashVal].key.equals(pKey))
            {hashVal= hashVal + 1;
              hashVal = hashVal % arraySize;}
            { // gakoa badago, balioa aldatu
            V temp = hashArray[hashVal].data;
            hashArray[hashVal].data=pData;
            return temp; }}
    }
```

Hash taula itxiaren adibidea

th=new HashTable<Integer,String>(10)



Hash taula itxiaren adibidea

Gakoek lau digitu osoak dituztela suposatuko dugu (dddd):

HASH (‘dddd’) → d+d+d+d

eta kolisioak proba linealaz ebazten direla.

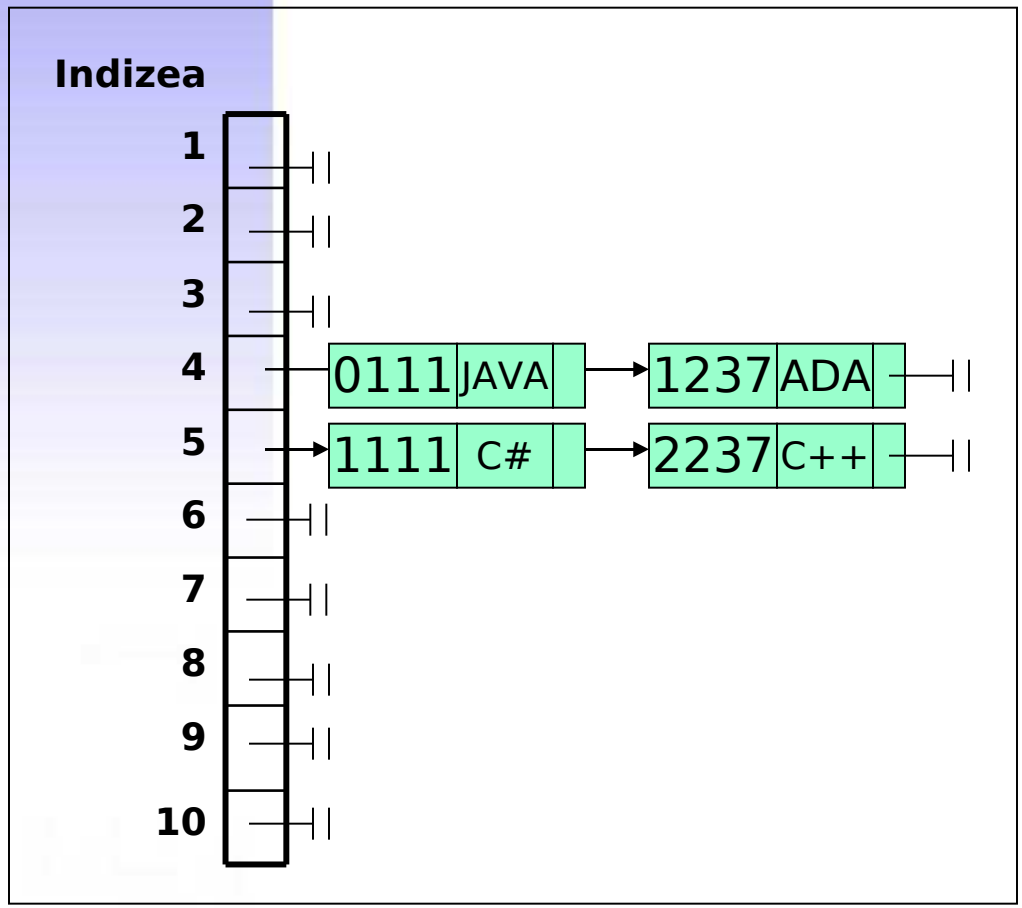
Ondorengo eragiketa egin ondoren hash taularen egoera zein den adierazi:

th.insert (new Integer(1237), ”ADA”);
th.insert (new Integer(2237), ”C”);
th.insert (new Integer(0111), ”JAVA”));
th.modify (new Integer(2237),”C++”);
th.remove (new Integer(0111));
th.insert (new Integer(1111),”C#”));
th.remove (new Integer(1237));

Hash taulak:
Hash irekia

- **Ez dago** muga logikorik elementuak txertatzeko
- Hash funtzioa erabiliz gako bateri dagokion posizioa lortzen da
- Posizio bereko sinonimoak zerrenda estekatu batean gordetzen dira
- Kolisioen kudeaketa ez da beharrezkoa

Hash taula irekiaren adibidea




```
Map th=new HashTable<Integer,String>(10);
th.put(1237,"ADA");
th.put(2237,"C");
th.put(0111,"JAVA");
th.put (2237,"C++");
th.put (1111,"C#");
```

Hash taulak:
Inplementazioa: datuak eta taula

```
class DataItem<K,V> {
      public K key;                // data item (key)
      public V data;
      public DataItem(K pKey, V pData) {// eraik.
        key=pKey;
        data=pData; }
   }  // end class DataItem
```

```
class HashTableIrekia<K,V> implements Map<K,V> {
   LinkedList<DataItem<K,V>>[] hashArray;        // array holds
                                                 //hash table
   int arraySize;
}
```

Hash taulak:
Eraikitzailea eta hash funtzioa

```
public HashTableIrekia(int size)       // eraikitzailea
      {
      arraySize = size;
      hashArray = new LinkedList<K,V>[arraySize];
      for (int j=0;j<arraySize;j++)
            hashArray[j]=new LinkedList<DataItem<K,V>>();
      }
```

```
public int hashFunc(K pKey)
      {
      return (k.hashCode() % arraysize); // hash function
      }
```

Hash taulak:
put metodoa. Proba lineala

```
public V put (K pKey, V pData) { // insert a DataItem
          // (assumes table not full)
         if (! containsKey(pKey)) {
            int hashVal = hashFunc(pKey);  // hash the key
            hashArray[hashVal].insertFirst(new DataItem(pKey, pData));
            return null;}
         else {
            int hashVal = hashFunc(pKey);
            hashArray[hashVal].goFirst();
            while (!hashArray[hashVal].getActual().key.equals(pKey)){
               hashArray[hashVal].goNext();}
            V elem;
            elem = hashArray[hashVal].getActual().data;
            hashArray[hashVal].removeActual();
            hashArray[hashVal].insertFirst(new DataItem(pKey, pData));
            return elem;
         }
        }  // end put()
```

Hash taulak:
metodoak

```
public boolean containsKey  (K pKey) {}   // find item with pKey
```

```
public V remove  (K pKey) {} // delete item with pKey
```

```
public V get  (K pKey) {} // return key associated value or null
```

Hash taulak:
Ez dira erabili behar

- Aldez aurretik elementuen kopurua ezagutzen ez denean
- Elementuak orden batean atzitu behar direnean