

# **ESTRUCTURES DE DADES**

## **-TAULES DE HASH-**

Alumne: Carlos Martínez

Professor: Albert Solé

## Classes utilitzades

- Ciutadà: no era necessari implementar l'ús de la classe ciutadà pero ho fet per a fer proves de cerca amb el DNI d'un ciutadà.
- HashElem: representa cada node de la posició de la taula de hash. Conté un comptador per a saber si està plena o no i conté les col·lisions corresponents en cas de tenir. A més té alguns mètodes propis.
- HashTable: estructura principal d'aquesta segona fase de la pràctica. Conté tots els mètodes encarregats de calcular els hashes, comprovar si algun element existeix, etc.
- Nodo: utilitzada a la classe HashElem per a representar el contingut de HashElem.

## Fitxers creats

Per a una visualització més simple he fet que el programa creei 3 fitxers en total:

- CostesTemporales.csv: és el fitxer que demana el treball, amb el numero d'elements, temps que triga en afegir-los i desviació estàndar.
- hashCodes.csv: conté els elements afegits per a que puguem comprovar manualment si ens troba un element existent a la taula.
- LogBusqueda.txt: registre de proves que fa el programa de forma automàtica, en total intenta buscar uns 40.000 elements, i a la consola no surten aquestes 40.000 línies.
- 

## Aspectes a destacar

### Problemes trobats durant el desenvolupament

-Casting entre classes, ja que a vegades operem amb dades genèriques, a vegades tipus primitius o de tipus ciutadà.

-Problemes a la hora de redimensionar la taula ja que no recalculava el hash y un cop redimensionada si volia trobar un element, no era posible ja que havia canviat el seu hash.

-Errors amb les col·lisions ja que al principi no assignava bé les col·lisions a la posició de la taula corresponent.

-Problemes de redimensionament, a vegades redimensionava la taula innecessàriament però ho vaig acabar corregint.

## Creació de les excepcions pròpies(no utilitzades)

### -ElementoNoEncontrado:

```
1 package Exceptions;
2
3 public class ElementoNoEncontrado extends Exception{
4     private static final long serialVersionUID = 1L;
5
6     public ElementoNoEncontrado(int n) {
7         super("Se han recorrido "+n+" posiciones pero no se ha encontrado el elemento");
8     }
9 }
10
```

### -NoSePuede:

```
1 package Exceptions;
2
3 public class NoSePuede extends Exception{
4     private static final long serialVersionUID = 1L;
5     public NoSePuede(int posi) {
6         super("No se puede tratar el elemento en la posición: "+posi);
7     }
8 }
9
```



# Mètodes destacables

Creació de la taula

```
18  /**
19   * Constructor principal de la clase HashTable
20   */
21  public HashTable() {
22      tablaHash=new HashElem[tableSize];
23  }
24  }
```

Càlcul del hash donada una dada T

```
25  /**
26   * Método que calcula el hash de los elementos genéricos
27   * @param data datos del cual queremos obtener el hash
28   * @return hash
29   */
30  public int hashKey(T data) {
31      int value=0;
32      String temp=new String();
33      if(data instanceof Integer||data instanceof Long)
34          temp=data.toString();
35      else if(data instanceof String)
36          temp=(String)data;
37      else if(data instanceof Ciutada) {
38          Ciutada ciud=(Ciutada)data;
39          temp=ciud.getDni();
40      }
41      else return (data.hashCode()&0x7fffffff)%tableSize;
42      int i=0;
43      while(i<temp.length()) {
44          value=(value*3+(int)(temp.charAt(i)))%tableSize;
45          i++;
46      }
47      return value;
48  }
```

Càlcul del hash donat un long

```
49  /**
50   * Returns hashKey for a long number in a range 0-tablaHash.length
51   * @param data number from which we want to get the hash
52   * @return hash key
53   */
54  public int hashKey(long data) {
55      return (int) (data%tableSize);
56  }
```

## Assignació d'element a taula de hash donat long

```
57◉  /**
58   * Method used to assign the integer to a position of the hashTable
59   * @param data integer we want to assign
60   * @return
61   */
62
63◉  int hashing(long data) {
64      int value=hashKey(data);
65      if(tablaHash[value]==null) {
66          tablaHash[value]=new HashElem();
67      }
68      if(tablaHash[value].estado!=2) {
69          tablaHash[value]=new HashElem(data);
70          firstElem=value;
71          nElems++;
72          if(factorCarga())
73              resize();
74      }
75      else if (tablaHash[value].estado==2) {
76          counter++;
77          tablaHash[value].append(new Nodo(data));
78          //nElems++;
79      }
80      return value;
81  }
```

## Assignació d'element a taula de hash donada T

```
94◉  public int hashing(T data) {
95
96
97      int value=0;
98      if(data instanceof Long) {
99          value=hashKey((Long)data);
100      }
101      else if(data instanceof Integer) {
102          value=hashKey((Integer)data);
103      }
104      else
105          value=hashKey(data);
106      if(tablaHash[value]==null) {
107          tablaHash[value]=new HashElem();
108      }
109      if(tablaHash[value].estado!=2) {
110          tablaHash[value]=new HashElem(data);
111          firstElem=value;
112          nElems++;
113          if(factorCarga())
114              resize();
115      }
116      else if(tablaHash[value].estado==2) {
117          int val=0;
118          //System.out.println("OCUPADO");
119          counter++;
120          //nElems++;
121          //tablaHash[value].addElems();
122
123          tablaHash[value].append(new Nodo(data));
124      }
125
126
127      return value;
128  }
```

## Buscar un element a la taula

```
227* /**
228 * Método usado para buscar un elemento pasado como parámetro
229 * @param data elemento que queremos buscar
230 * @return texto que usaremos para guardar en el fichero LogBusqueda e imprimir por pantalla
231 */
232* public String findElem(T data) {
233
234     int key=0;
235     if(data instanceof Integer) {
236         key=hashKey((Integer)data);
237     }
238     else if(data instanceof Long)
239         key=hashKey(((Long) data).longValue());
240     else if(data instanceof T)
241         key=hashKey(data);
242     int posi=0;
243     //System.out.println(key);
244     String text=new String();
245     if(tablaHash[key]!=null&&tablaHash[key].lookFor(data)==1)
246         text="El elemento "+data+" estaba en la posición "+key;
247     //System.out.println("El elemento "+data+" estaba en la posición "+key);}
248     else
249
250         text="Element "+data+" not found";
251     //System.out.println("Element not found");}
252     return text;
253 }
```

## Redimensionar la taula

```
294* /**
295 * Método encargado de redimensionar la tabla de hash y recalcular todos los hashes de nuevo
296 */
297* public void resize() {
298     tableSize*=3;
299     int size=tablaHash.length;
300     HashElem[]aux=new HashElem[tableSize];
301     for(int i=0;i<tablaHash.length;i++) {
302         aux[i]=tablaHash[i];
303
304     }
305     tablaHash=new HashElem[aux.length];
306     int i=0;
307     int mode=1;
308     int key=0;
309     for(int j=0;j<size;j++) {
310         //tablaHash[j]=aux[j];
311         if(aux[j]!=null) {
312             Nodo temp=aux[j].firstElem;
313             while(aux[j].firstElem!=null&&aux[j].firstElem.data!=null&&temp!=null) {
314                 if(temp!=null&&temp.data instanceof Long) {
315                     //System.out.println(temp.data);
316                     key=hashing((Long) temp.data);
317                 }
318                 else {
319                     key=hashing((T)temp.data);
320                 }
321                 temp=temp.nextCol;
322             }
323         }
324     }
325 }
326 }
```

Comparar si element està a posició de la taula de hash

```
57  /**
58   * Método para buscar un dato pasado como parámetro
59   * @param data dato que queremos buscar
60   * @return 1 if found
61   */
62  public int lookFor(T data) {
63      boolean found=false;
64      Nodo aux=firstElem;
65      int i=0;
66      while(aux!=null) {
67          if(aux.data instanceof Ciutada) {
68              Ciutada ciu=(Ciutada)aux.data;
69              if(ciu.getDni().equalsIgnoreCase((String)data)&&aux.data!=null)
70                  return 1;
71          }
72          else
73              try {
74                  if(aux.data.compareTo(data)==0&&aux.data!=null)
75                      return 1;}catch(ClassCastException e) {
76
77                  }
78              aux=aux.nextCol;
79              i++;
80          }
81      return -1;
82  }
83  }
84 }
```

## Joc de proves

Primer de tot he creat una taula amb 10 posicions buides.

Crear llista

numbers	HashTable<T> (id=41)
▲ counter	0
▲ fileName	null
▲ firstElem	0
▲ firstTime	true
▲ nElems	0
▲ tablaHash	HashElem[10] (id=42)
▲ [0]	null
▲ [1]	null
▲ [2]	null
▲ [3]	null
▲ [4]	null
▲ [5]	null

## Imprimir llista

```
52     numbers=new HashTable();
53     System.out.println("Llista:");
54     numbers.printList();
55     System.out.println("Fi llista");
56     //We add the strings to the string array called 's
57     separator();
58
59
60     //We add the long numbers to the long array called
61     initTime=System.nanoTime();
62     digits=generateNumber(numbers,nElems);
63     //numbers.prinHash();
64     endTime=System.nanoTime();
65
66     duration[i]=(endTime-initTime);
67     System.out.println("\tAfegir els "+nElems+" "+digi
68     writeToFile("CostesTemporales.csv", nElems, durationFi
```

Console X Problems Debug Shell

main (4) [Java Application] D:\Archivos de Programa\Java\bin\javaw.exe (20 abr 2022 18:10:52)

BENVINGUT/UDA AL PROGRAMA PRINCIPAL

A continuació s'executaran els següents mètodes de manera automàtica:

- 1- Crear taula de hash
- 2- Insertar 100/1.000/10.000 longs a la taula de hash
- 3- Insertar 100/1.000/10.000 String a la taula de hash
- 4- Buscarem alguns elements que existeixin i uns altres que no

COMENCEM...

Llista:  
Fi llista

## Inserir elements

100 elements

▲ tablaHash	HashElem<T> [270] (id=46)
▲ [0..99]	
▲ [0]	null
▲ [1]	null
> ▲ [2]	HashElem<T> (id=50)
▲ [3]	null
▲ [4]	null
▲ [5]	null
▲ [6]	null
▲ [7]	null
> ▲ [8]	HashElem<T> (id=51)
> ▲ [9]	HashElem<T> (id=52)
▲ [10]	null
▲ [11]	null
> ▲ [12]	HashElem<T> (id=53)
> ▲ [13]	HashElem<T> (id=54)
▲ [14]	null
▲ [15]	null
▲ [16]	null
▲ [17]	null
▲ [18]	null
> ▲ [19]	HashElem<T> (id=55)
▲ [20]	null
▲ [21]	null
▲ [22]	null
▲ [23]	null
▲ [24]	null
▲ [25]	null
▲ [26]	null
▲ [27]	null
▲ [28]	null
> ▲ [29]	HashElem<T> (id=56)
> ▲ [30]	HashElem<T> (id=57)
▲ [31]	null
▲ [32]	null
▲ [33]	null
▲ [34]	null
▲ [35]	null
▲ [36]	null



## Escritura del fitxer LogBusqueda.txt

```
*****  
Afegir els 100 int[] ha trigat 735901600 ns  
*****
```

```
1El elemento 1189493218 estaba en la posicion 118  
2Element 1338335257 not found  
3El elemento 1315360622 estaba en la posicion 2  
4Element 1237360373 not found  
5El elemento 1144511447 estaba en la posicion 77  
6Element 1267962888 not found  
7El elemento 1286492290 estaba en la posicion 70  
8Element 1394626982 not found  
9El elemento 1116249580 estaba en la posicion 190  
10Element 1151899425 not found  
11El elemento 1022826982 estaba en la posicion 22  
12Element 1101447035 not found  
13El elemento 1120602729 estaba en la posicion 129  
14Element 1301570716 not found  
15El elemento 1403719789 estaba en la posicion 49  
16Element 1109543820 not found  
17El elemento 1211465563 estaba en la posicion 133  
18Element 1098686903 not found  
19El elemento 1297716845 estaba en la posicion 185
```

## Escritura del fitxer hashCodes.csv

```
79759 key= 156825;1093547475  
79760 key= 156829;1113624139  
79761 key= 156838;1039222408  
79762 key= 156839;1350804299  
79763 key= 156842;1368912662  
79764 key= 156850;1099452400  
79765 key= 156851;1089807731  
79766 key= 156854;1360055324  
79767 key= 156855;1276993065  
79768 key= 156855;1257310065  
79769 key= 156856;1098074596  
79770 key= 156857;1043355857  
79771 key= 156862;1136062792  
79772 key= 156862;1261049842  
79773 key= 156863;1098271433  
79774 key= 156865;1223258485  
79775 key= 156865;1272859645  
79776 key= 156869;1050441749  
79777 key= 156875;1219912385
```

## Mètode buscar element

EXISTEIXEN: Agafem el tercer per a fer la prova

```
Pots utilitzar el fitxer hashCodes.csv per a fer proves
Escriu -1 per a sortir del programa
Escriu l'element que vulguis buscar:
1077801910
El elemento 1077801910 estaba en la posicion 157660
Escriu l'element que vulguis buscar:
```

```
79763 key= 157658;1404933368
79764 key= 157660;1020721210
79765 key= 157660;1077801910
79766 key= 157665;1302384945
79767 key= 157668;1133111148
79768 key= 157669;1361237119
79769 key= 157671;1389186981
79770 key= 157671;1160076861
```

NÚMERO INVENTAT:

1. Número invàlid que se surt del rang d'un enter

```
Pots utilitzar el fitxer hashCodes.csv per a fer proves
Escriu -1 per a sortir del programa
Escriu l'element que vulguis buscar:
1237198237918274239187432
El número introduit no és vàlid
Escriu l'element que vulguis buscar:
```

2. Número que no existeix, inventat.

```
Escriu l'element que vulguis buscar:
287312332
Element 287312332 not found
Escriu l'element que vulguis buscar:
```

3. No dona error perquè vaig fer proves amb String i vaig haver d'adaptar el programa per a que llegís strings i comprovés si existien a la taula.

```
Escriu l'element que vulguis buscar:
oijifjrnfs
Element oijifjrnfs not found
Escriu l'element que vulguis buscar:
```

Mètode factor de càrrega i comprovar si s'ha de redimensionar

Nelems = 8 i la mida de la taula és 10, per la qual cosa el factor de càrrega supera el 0,75.

```

283 public boolean factorCarga() {
284     //System.out.println("nElems= "+nElems);
285     float res=(float)nElems/tablaHash.length;
286     //System.out.println(res);
287     if(res>0.75) {
288         return true;
289     }
290     return false;
291 }
292 /**

```

Name	Value
no method return value	
this	HashTable<T> (id=21)
counter	4
fileName	null
firstElem	7
firstTime	true
nElems	8
tablaHash	HashElem<T>[10] (id=22)
tableSize	10
writer	null
res	0.8

Mètode redimensionar la taula

Name	Value
no method return value	
this	HashTable<T> (id=21)
counter	4
fileName	null
firstElem	7
firstTime	true
nElems	8
tablaHash	HashElem<T>[10] (id=22)
tableSize	10
writer	null

Abans de redimensionar

this	HashTable<T> (id=21)
counter	6
fileName	null
firstElem	28
firstTime	true
nElems	18
tablaHash	HashElem<T>[30] (id=34)
tableSize	30
writer	null
size	10
aux	HashElem<T>[30] (id=33)
i	0
mode	1
key	28

Després de redimensionar

Col·lisions

Buscarem els elements amb la clau 156120

[156120]	HashElem<T> (id=58)
estado	2
firstElem	Nodo<T> (id=60)
data	Integer (id=66)
value	1285062360
nextCol	Nodo<T> (id=73)
data	Integer (id=75)
value	1366943640
nextCol	Nodo<T> (id=76)
data	Integer (id=77)
value	1070714490
nextCol	Nodo<T> (id=62)
data	Integer (id=78)
value	1217549670

```

79224 key= 156120;1285062360
79225 key= 156120;1366943640
79226 key= 156120;1070714490
79227 key= 156120;1217549670
79228 key= 156121;1337615971
79229 key= 156122;1345686002

```

Mètode generar números aleatoris per les proves

```
Int: 1266745696
Long: 8794722359
Int: 1364016506
Long: 7816813191
Int: 1179589060
Long: 1722320136
Int: 1301699622
Long: 8625739028
Int: 1295593452
Long: 5016347952
Int: 1403852876
Long: 5642199495
Int: 1173842681
Long: 8766960131
Int: 1262366339
Long: 1461077504
Int: 1006498603
Long: 2058119847
Int: 1064532190
Long: 8822845728
Int: 1175148497
Long: 3615604721
Int: 1348103739
Long: 4039636759
Int: 1073820543
Long: 3380963118
Int: 1191585159
Long: 3017691425
Int: 1273604740
Long: 2825171929
```

```
for(int i=0;i<15;i++) {
    System.out.println("Int: "+randomInt());
    System.out.println("Long: "+randomLong());
}
```



## Anàlisi del cost

Fent moltes més proves de les que he inclòs en aquest document m'he adonat del potencial de les taules de hash. La velocitat de cerca és molt superior a la que podria tenir una llista estàtica i m'ha interessat molt aquest fet.

Com només aplicant una operació podem reduir de gran forma les iteracions a fer per a poder trobar un element. A més la velocitat de cerca és extremadament ràpida com hem pogut comprovar.

En definitiva, les taules de hash son una forma molt recomanable d'indexar la informació per a després accedir de manera molt ràpida i poden ser utilitzades en molts àmbits

Per a concluir, puc dir que m'ha suposat un repte aquesta pràctica i buscaré i investigaré més sobre les taules de hash ja que m'han semblat molt interessants.

```
1 LIST SIZE; TIME TAKEN; STDEV
2 100; 0.001126501s; 1.2684969519599797E12
3 100; 3.39E-5s; 1.1487503619684E9
4 100; 2.25E-5s; 5.0604752025E8
5 100; 2.2E-5s; 4.8380641935999995E8
6 100; 2.47E-5s; 6.098459884036001E8
7 100; 2.3E-5s; 5.287884211600001E8
8 100; 2.31E-5s; 5.3339657734440005E8
9 100; 2.67E-5s; 7.126048725156E8
10 100; 2.52E-5s; 6.347860094016E8
11 100; 2.24E-5s; 5.015593160704E8
12 100; 2.02E-5s; 4.0787680032159996E8
13 100; 2.28E-5s; 5.1963208479359996E8
14 100; 2.02E-5s; 4.0787680032159996E8
15 100; 2.4199E-5s; 5.853573877832639E8
```