

ESTRUCTURES DE DADES

-TAULES DE HASH-

Alumne: Carlos Martínez

Professor: Marc Ruiz

Contenido

Classes utilitzades.....	3
Fitxers creats.....	3
Aspectes a destacar.....	3
Problemes trobats durant el desenvolupament.....	3
TAD TAULA HASH	4
Creació de les excepcions pròpies	4
Mètodes destacables	5
Creació de la taula.....	5
Càlcul del hash donada una dada clau	5
Assignació d'element a taula de hash	6
Buscar un element a la taula	7
Redimensionar la taula	7
Esborrar un element de la taula de hash	8
Joc de proves.....	9
Crear llista.....	9
Inserir elements	10
Mètode buscar element	10
Cèrques fetes	11
Mètode redimensionar la taula.....	12
Col·lisions	12
Anàlisi del cost	13
ACCESSOS:.....	14
Llista.....	14
Taula de hash	14
STDEV:.....	15
Llista.....	15
Taula de hash	15
Codi Font.....	16
Taula de Hash	16
Main.....	20
Nodo	25

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.
- Nodo: representa cada node de la posició de la taula de hash. Guardo la clau clau, valor i hash de la dada que conté.
- HashTable: estructura principal d'aquesta segona fase de la pràctica. Conté tots els mètodes encarregats de calcular els hashes, comprovar si algún element existeix, etc.
- ListaDoble: llista doblement encadenada de la primera part de la pràctica

Fitxers creats

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

- LogCerques: fitxers on guardo el número de cerques que s'han fet fins que s'ha pogut trobar l'element o fins que hem recorregut tots els elements sense haver-lo trobat
- hashCodes.txt: fitxer on guardo cada element de la taula de 50000 elements per a poder provar el seu funcionament i veure si un element es troba dins de la taula.

Aspectes a destacar

Problemes trobats durant el desenvolupament

-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.

TAD TAULA HASH

```
1 package Data;
2
3 import Exceptions.ElementoNoEncontrado;
4
5 1 usage 1 implementation Carlos Martínez +1
6 public interface TADTaulaHash<K, T> {
7     1 usage 1 implementation Carlos Martínez
8     public void Crear();
9     10 usages 1 implementation Carlos Martínez García-Villarrubia
10    public void Inserir(K key, T data) throws ElementoNoEncontrado;
11    1 implementation Carlos Martínez
12    T Obtener(K key);
13    13 usages 1 implementation Carlos Martínez García-Villarrubia
14    int Buscar(K key) throws ElementoNoEncontrado;
15    1 usage 1 implementation Carlos Martínez
16    int Mida();
17    1 implementation Carlos Martínez García-Villarrubia
18    void Esborrar(K key) throws ElementoNoEncontrado;
```

Creació de les excepcions pròpies

-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

```
25      2 usages  Carlos Martínez +1  
26      public HashTable() {  
27          tablaHash=new Nodo[tableSize];  
28          nElems=0;  
29      }
```

Càlcul del hash donada una dada clau

Calculem el hash amb el toString de l'objecte

```
26  /**  
27   * Method used to return an object's hash  
28   * @param key key from which we want to obtain the hash  
29   * @return hash of the key passed by parameter  
30   */  
31  Carlos Martínez García-Villarrubia +1 *  
32  @ public int hash(K key){  
33      String str=key.toString();  
34      int hash=0;//=key.hashCode();  
35      for(int i=1;i<str.length();i++){  
36          hash=(hash*32+(int)str.charAt(i))%tablaHash.length;  
37      }  
38      hash=hash < 0 ? hash * -1 : hash;  
39      return hash;  
40  }
```


Assignació d'element a taula de hash

```
10 usages Carlos Martínez García-Villarrubia *
64 @Override
65 public void Inserir(K key, T data) {
66
67     if(factorCarga()>=0.75){
68         //printTable();
69         try{
70             resize();
71         }catch(ElementoNoEncontrado ignored){
72         }
73     }
74     int hash=hash(key);
75     //int index=getIndex(key);
76     if(tablaHash[hash]==null){
77         tablaHash[hash]=new Nodo<>(key,data,hash);
78         nElems++;
79     }
80     else{
81         try{
82             int offset=Buscar(key);
83             replace(hash,offset,data);
84         }catch (ElementoNoEncontrado e){
85             tablaHash[hash].add(key,data,hash);
86         }
87     }
88
89 }
90
91 }
```

Buscar un element a la taula

Si trobem l'element el retornem el nombre d'iteracions fetes, si no, seguim buscant.

```
5 usages Carlos Martínez García-Villarrubia +1 *
106 @Override
107 public int Buscar(K key) throws ElementoNoEncontrado{
108     int hash=hash(key);
109     int index=hash%tablaHash.length;
110     int c=0;
111     Nodo<K,T>temp=tablaHash[index];
112     if(temp!=null){
113         while(temp!=null){
114             if(temp.key.equals(key)){
115                 //System.out.println("FOUND at "+index+"
116                 return c;
117             }
118             temp=temp.nextCol;
119             c++;
120         }
121     }
122     throw new ElementoNoEncontrado(c);
123 }
124 }
```

Redimensionar la taula

```
1 usage Carlos Martínez García-Villarrubia *
199 public void resize() throws ElementoNoEncontrado {
200     //Nodo[] listaAux=new Nodo[tablaHash.length*2];
201     HashTable<K,T> tablaAux=new HashTable<>(nElems: tablaHash.length*2);
202     HashTable <K,T> aux=new HashTable<>();
203     Nodo<K,T> temp;
204     K key;
205     T data;
206     for(int i=0;i<tablaHash.length;i++){
207         temp=tablaHash[i];
208         while(temp!=null){
209             key=temp.getKey();
210             data=temp.getData();
211             tablaAux.Inserir(key,data);
212             //System.out.println("Num: "+temp.data+"\tHash: "+tablaAux.h
213             temp=temp.nextCol;
214         }
215     }
216     tablaHash=tablaAux.tablaHash;
217     tableSize=tablaHash.length;
218 }
219 }
220 }
```

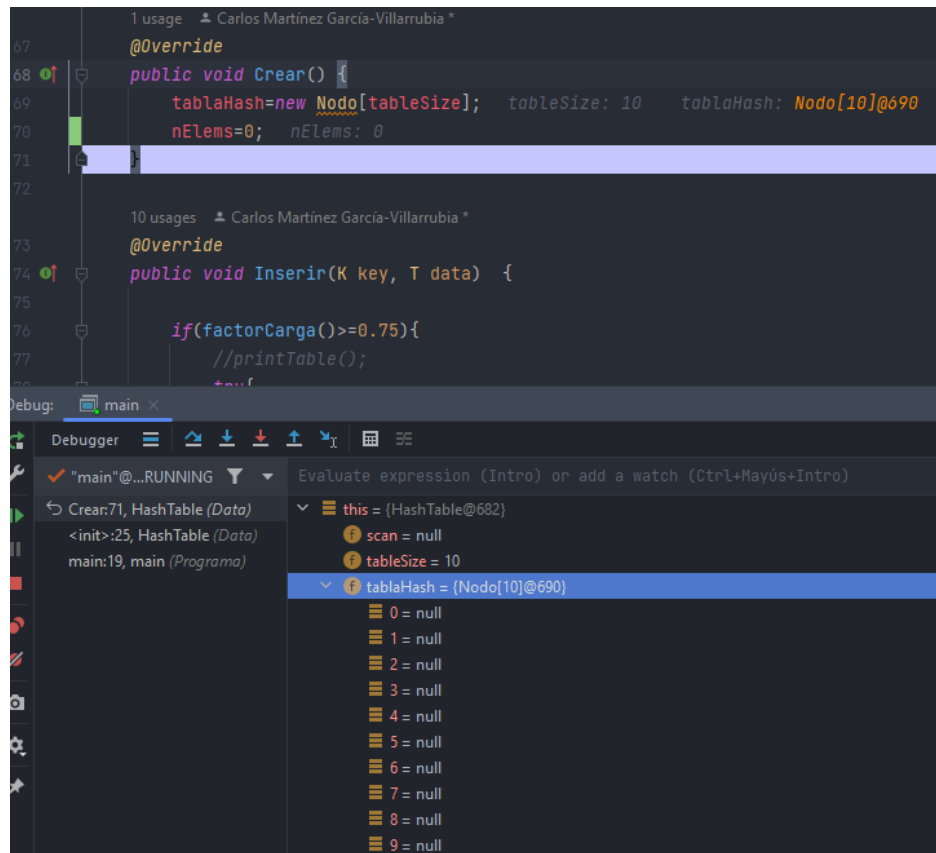
Esborrar un element de la taula de hash

```
2
3     import Exceptions.ElementoNoEncontrado;
4
5     1 usage 1 implementation Carlos Martínez +1 *
6     public interface TADTaulaHash<K, T extends Comparable<T>> {
7         1 usage 1 implementation Carlos Martínez
8         public void Crear();
9         10 usages 1 implementation Carlos Martínez García-Villarrubia
10        public void Inserir(K key, T data) throws ElementoNoEncontrado;
11        1 implementation Carlos Martínez
12        T Obtener(K key);
13        13 usages 1 implementation new *
14        public int Buscar(K key) throws ElementoNoEncontrado;
15        1 usage 1 implementation new *
16        public int Mida();
17        1 implementation new *
18        public void Esborrar(K key) throws ElementoNoEncontrado;
19        1 implementation new *
20        public ListaDoble<K,T>ObtenirValors();
21        1 implementation new *
22        public ListaDoble<K,T>ObtenirClaus();
23        1 usage 1 implementation new *
24        public float factorCarga();
25
26    }
```


Joc de proves

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

Crear llista



Inserir elements

8 ciutadans

```
Ciutada carlos=new Ciutada( nom: "Carlos", cognom: "Martinez", DNI: "49424598J"); carlos: "Ciutada [nom=Carlos, cognom=Martinez, DNI=49424598J]"
Ciutada david=new Ciutada( nom: "David", cognom: "Marti", DNI: "7771391023"); david: "Ciutada [nom=David, cognom=Marti, DNI=7771391023]"
Ciutada nil=new Ciutada( nom: "Carlos", cognom: "Martinez", DNI: "44548898T"); nil: "Ciutada [nom=Carlos, cognom=Martinez, DNI=44548898T]"
Ciutada genis=new Ciutada( nom: "Genis", cognom: "Martinez", DNI: "73981391P"); genis: "Ciutada [nom=Genis, cognom=Martinez, DNI=73981391P]"
Ciutada roger=new Ciutada( nom: "Roger", cognom: "Massana", DNI: "3731918T"); roger: "Ciutada [nom=Roger, cognom=Massana, DNI=3731918T]"
Ciutada lluis=new Ciutada( nom: "Lluis", cognom: "Gallart", DNI: "7137391890"); lluis: "Ciutada [nom=Lluis, cognom=Gallart, DNI=7137391890]"
Ciutada gerard=new Ciutada( nom: "Gerard", cognom: "Panisello", DNI: "3241233Y"); gerard: "Ciutada [nom=Gerard, cognom=Panisello, DNI=3241233Y]"
Ciutada eros=new Ciutada( nom: "Eros", cognom: "Villar", DNI: "1413133T"); eros: "Ciutada [nom=Eros, cognom=Villar, DNI=1413133T]"

try{
    tablaAux.Inserir( key: "49424598J",carlos); carlos: "Ciutada [nom=Carlos, cognom=Martinez, DNI=49424598J]"
    tablaAux.Inserir( key: "7771391023",david); david: "Ciutada [nom=David, cognom=Marti, DNI=7771391023]"
    tablaAux.Inserir( key: "44548898T",nil); nil: "Ciutada [nom=Carlos, cognom=Martinez, DNI=44548898T]"
    tablaAux.Inserir( key: "73981391P",genis); genis: "Ciutada [nom=Genis, cognom=Martinez, DNI=73981391P]"
    tablaAux.Inserir( key: "3731918T",roger); roger: "Ciutada [nom=Roger, cognom=Massana, DNI=3731918T]"
    tablaAux.Inserir( key: "7137391890",lluis); lluis: "Ciutada [nom=Lluis, cognom=Gallart, DNI=7137391890]"
    tablaAux.Inserir( key: "3241233Y",gerard); gerard: "Ciutada [nom=Gerard, cognom=Panisello, DNI=3241233Y]"
    tablaAux.Inserir( key: "1413133T",eros); eros: "Ciutada [nom=Eros, cognom=Villar, DNI=1413133T]"
}
```

```
▼ tablaHash = {Nodo[10]@870}
> 0 = {Nodo@871} "Nodo[data=Ciutada [nom=David, cognom=Marti, DNI=7771391023], nextCol=null]"
  1 = null
> 2 = {Nodo@872} "Nodo[data=Ciutada [nom=Carlos, cognom=Martinez, DNI=49424598J], nextCol=null]"
> 3 = {Nodo@873} "Nodo[data=Ciutada [nom=Roger, cognom=Massana, DNI=3731918T], nextCol=null]"
  4 = null
  5 = null
> 6 = {Nodo@874} "Nodo[data=Ciutada [nom=Carlos, cognom=Martinez, DNI=44548898T], nextCol=Nodo[data=Ciutada [nom=Lluis, cognom=Gallart, DNI=7137391890], nextCol=null]"
  7 = null
  8 = null
> 9 = {Nodo@875} "Nodo[data=Ciutada [nom=Genis, cognom=Martinez, DNI=73981391P], nextCol=Nodo[data=Ciutada [nom=Gerard, cognom=Panisello, DNI=3241233Y], nextCol=null]"
```

Mètode buscar element

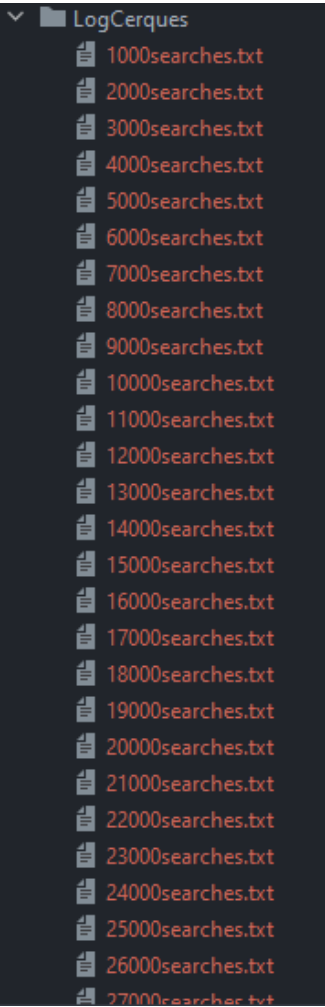
EXISTEIXEN: Agafem el tercer per a fer la prova

```
Ciutada carlos=new Ciutada( nom: "Carlos", cognom: "Martinez", DNI: "49424598J"); carlos: "Ciutada [nom=Carlos, cognom=Martinez, DNI=49424598J]"
Ciutada david=new Ciutada( nom: "David", cognom: "Marti", DNI: "7771391023"); david: "Ciutada [nom=David, cognom=Marti, DNI=7771391023]"
Ciutada nil=new Ciutada( nom: "Carlos", cognom: "Martinez", DNI: "44548898T"); nil: "Ciutada [nom=Carlos, cognom=Martinez, DNI=44548898T]"
Ciutada genis=new Ciutada( nom: "Genis", cognom: "Martinez", DNI: "73981391P"); genis: "Ciutada [nom=Genis, cognom=Martinez, DNI=73981391P]"
Ciutada roger=new Ciutada( nom: "Roger", cognom: "Massana", DNI: "3731918T"); roger: "Ciutada [nom=Roger, cognom=Massana, DNI=3731918T]"
Ciutada lluis=new Ciutada( nom: "Lluis", cognom: "Gallart", DNI: "7137391890"); lluis: "Ciutada [nom=Lluis, cognom=Gallart, DNI=7137391890]"
Ciutada gerard=new Ciutada( nom: "Gerard", cognom: "Panisello", DNI: "3241233Y"); gerard: "Ciutada [nom=Gerard, cognom=Panisello, DNI=3241233Y]"
Ciutada eros=new Ciutada( nom: "Eros", cognom: "Villar", DNI: "1413133T"); eros: "Ciutada [nom=Eros, cognom=Villar, DNI=1413133T]"
```

```
System.out.println(tablaAux.Buscar( key: "44548898T"));
```

```
13 usages  ▲ Carlos Martinez García-Villarrubia +1 *
@Override
public int Buscar(K key) throws ElementoNoEncontrado{ key: "44548898T"
    int hash=hash(key); hash: 17136
    int index=hash%tablaHash.length; hash: 17136 index: 6
    int c=0; c: 0
    Nodo<K,T>temp=tablaHash[index]; index: 6 temp: "Nodo{data=Ciutada [nom=Carlos, cognom=Martinez, DNI=44548898T], nextCol=null}"
    if(temp!=null){
        while(temp!=null){
            if(temp.key.equals(key)){ key: "44548898T" temp: "Nodo{data=Ciutada [nom=Carlos, cognom=Martinez, DNI=44548898T], nextCol=null}"
                return c; c: 0
            }
            temp=temp.nextCol;
            c++;
        }
    }
    throw new ElementoNoEncontrado(c);
}
```

Cèrques fetes



```
1 1 iteration until element has been found
2 4 iteration until element has been found
3 4 iteration until element has been found
4 1 iteration until element has been found
5 Se han recorrido 4 posiciones pero no se ha encontrado el elemento
6 1 iteration until element has been found
7 2 iteration until element has been found
8 4 iteration until element has been found
9 3 iteration until element has been found
10 4 iteration until element has been found
11 6 iteration until element has been found
12 2 iteration until element has been found
13 4 iteration until element has been found
14 4 iteration until element has been found
15 1 iteration until element has been found
16 4 iteration until element has been found
17 1 iteration until element has been found
18 3 iteration until element has been found
19 2 iteration until element has been found
20 Se han recorrido 3 posiciones pero no se ha encontrado el elemento
21 Se han recorrido 4 posiciones pero no se ha encontrado el elemento
```

1000 cerques

```
1 1 iteration until element has been found
2 2 iteration until element has been found
3 4 iteration until element has been found
4 1 iteration until element has been found
5 4 iteration until element has been found
6 1 iteration until element has been found
7 2 iteration until element has been found
8 1 iteration until element has been found
9 3 iteration until element has been found
10 3 iteration until element has been found
11 1 iteration until element has been found
12 2 iteration until element has been found
13 1 iteration until element has been found
14 5 iteration until element has been found
15 4 iteration until element has been found
16 1 iteration until element has been found
17 7 iteration until element has been found
18 Se han recorrido 11 posiciones pero no se ha encontrado el elemento
19 1 iteration until element has been found
20 7 iteration until element has been found
21 1 iteration until element has been found
22 Se han recorrido 9 posiciones pero no se ha encontrado el elemento
```

50000 cerques

Mètode redimensionar la taula

```
▼ this = {HashTable@721}
  scan = null
  tableSize = 10
  ▼ tablaHash = {Nodo@10}@723
    0 = null
    1 = {Nodo@873} "Nodo{data=118, nextCol=null}"
    2 = {Nodo@869} "Nodo{data=288, nextCol=Nodo{data=76, nextCol=null}}"
    3 = {Nodo@896} "Nodo{data=43, nextCol=Nodo{data=111, nextCol=null}}"
    4 = {Nodo@858} "Nodo{data=486, nextCol=null}"
    5 = {Nodo@725} "Nodo{data=156, nextCol=Nodo{data=77, nextCol=Nodo{data=230, nextCol=Nodo{data=57, nextCol=Nodo{data=87, nextCol=null}}}}}"
    6 = {Nodo@915} "Nodo{data=14, nextCol=null}"
    7 = null
    8 = {Nodo@893} "Nodo{data=266, nextCol=Nodo{data=103, nextCol=null}}"
    9 = {Nodo@855} "Nodo{data=231, nextCol=Nodo{data=415, nextCol=null}}"
```

Abans de redimensionar

```
▼ tablaHash = {Nodo@20}@923
  0 = null
  1 = {Nodo@924} "Nodo{data=118, nextCol=null}"
  2 = {Nodo@925} "Nodo{data=76, nextCol=null}"
  3 = {Nodo@926} "Nodo{data=43, nextCol=Nodo{data=111, nextCol=null}}"
  4 = {Nodo@927} "Nodo{data=486, nextCol=null}"
  5 = {Nodo@928} "Nodo{data=156, nextCol=null}"
  6 = null
  7 = null
  8 = null
  9 = {Nodo@929} "Nodo{data=231, nextCol=null}"
  10 = null
  11 = null
  12 = {Nodo@930} "Nodo{data=288, nextCol=null}"
  13 = null
  14 = null
  15 = {Nodo@931} "Nodo{data=77, nextCol=Nodo{data=230, nextCol=Nodo{data=57, nextCol=Nodo{data=87, nextCol=null}}}}}"
  16 = {Nodo@932} "Nodo{data=14, nextCol=null}"
  17 = null
  18 = {Nodo@933} "Nodo{data=266, nextCol=Nodo{data=103, nextCol=null}}"
```

Després de redimensionar

Col·lisions

Buscarem els elements amb la seva clau

```
28 try{
29     tablaAux.Inserir( key: "49424598J",carlos); carlos: "Ciutada [nom=Carlos, cognom=Martinez, DNI=49424598J]"
30     tablaAux.Inserir( key: "7771391023",david); david: "Ciutada [nom=David, cognom=Marti, DNI=7771391023]"
31     tablaAux.Inserir( key: "44548898T",nil); nil: "Ciutada [nom=Carlos, cognom=Martinez, DNI=44548898T]"
32     tablaAux.Inserir( key: "73981391P",genis); genis: "Ciutada [nom=Genis, cognom=Martinez, DNI=73981391P]"
33     tablaAux.Inserir( key: "3731918T",roger); roger: "Ciutada [nom=Roger, cognom=Massana, DNI=3731918T]"
34     System.out.println(tablaAux.Buscar( key: "49424598J"));
35     System.out.println(tablaAux.Buscar( key: "7771391023"));
36     System.out.println(tablaAux.Buscar( key: "44548898T"));
37     System.out.println(tablaAux.Buscar( key: "73981391P"));
38     System.out.println(tablaAux.Buscar( key: "3731918T")); tablaAux: HashTable@686
39
40 }catch(ElementoNoEncontrado e){
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
```

```
(Intro) or add a watch (Ctrl+Mayus+Intro)
main
Object = 0
83) []
ble@686)
```

```
C:\Users\carli\.jdk\az
Connected to the target
0
0
0
0
0
0
0
0
```

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, respecte a una cerca amb una llista doblement enllaçada com la de la primera part.

Fent un anàlisi basant-nos en moltes proves podem comprovar la gran diferencia d'accessos que hi ha entre una taula de hash i una llista a l'hora de trobar un element.

Com indica el guió de la pràctica, he fet diferents taules de hash i llistes de 1000 elements fins a 50000 elements, sumant de 1000 en 1000.

Per cada una d'aquestes taules he fet 1000 cerques d'elements generats aleatòriament i fent un promig de proves he comprovat la gran diferencia que hi ha. Faig un breu anàlisi però a l'excel estan tots els càlculs ben desenvolupats.

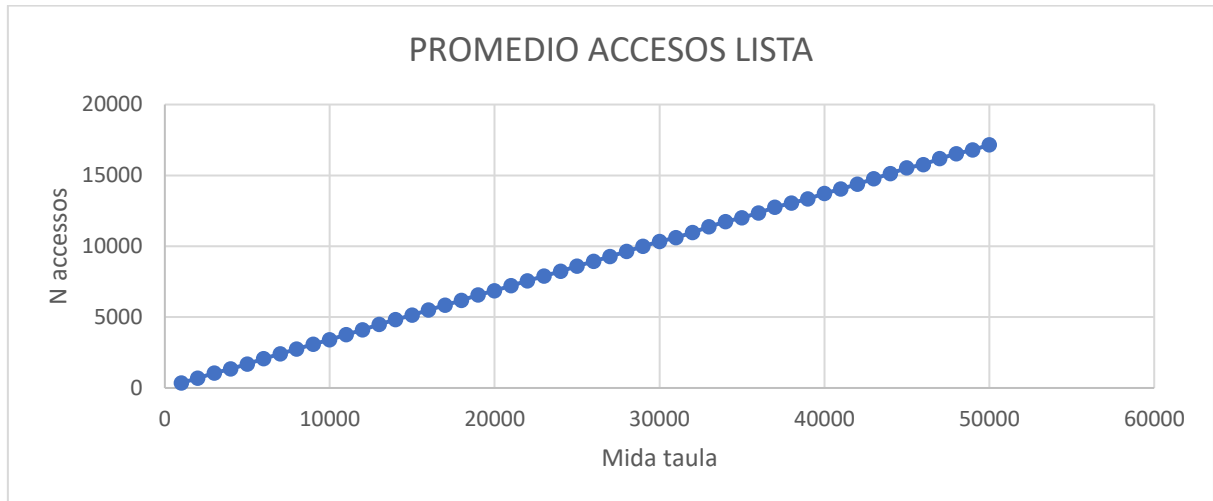
Mida	Taula de hash		Llista encadenada	
	Accessos	desvest	Accessos	desvest
1000	2,812019036	1,749255096	455,0885402	343,6008806
5000	2,716108626	2,292981763	1827,167658	1409,069324
10000	2,77880517	1,759373095	3540,456148	2714,474981
20000	4,503028618	2,438097533	6971,861079	5337,171837
50000	2,8737614	2,472879112	17149,38392	13090,67336

Veiem que les diferències són molt grans. Portant aquestes dades a una gràfica obtenim el següent:

ACCESSOS:

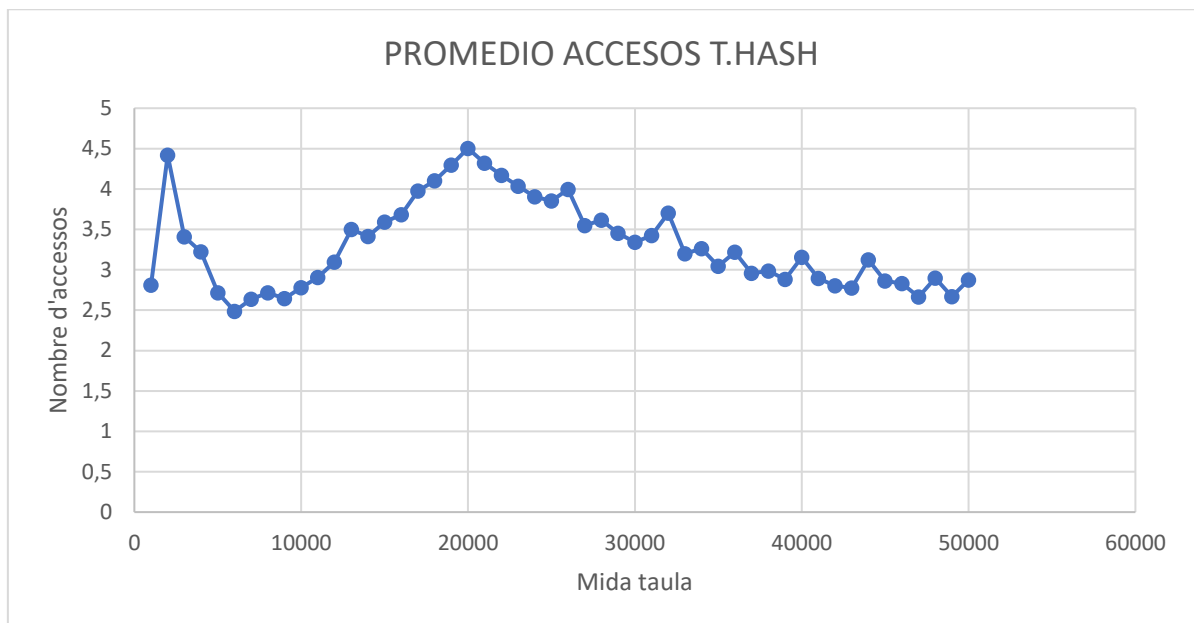
Llista

Podem veure que el cost és lineal ja que ha de recórrer element a element.



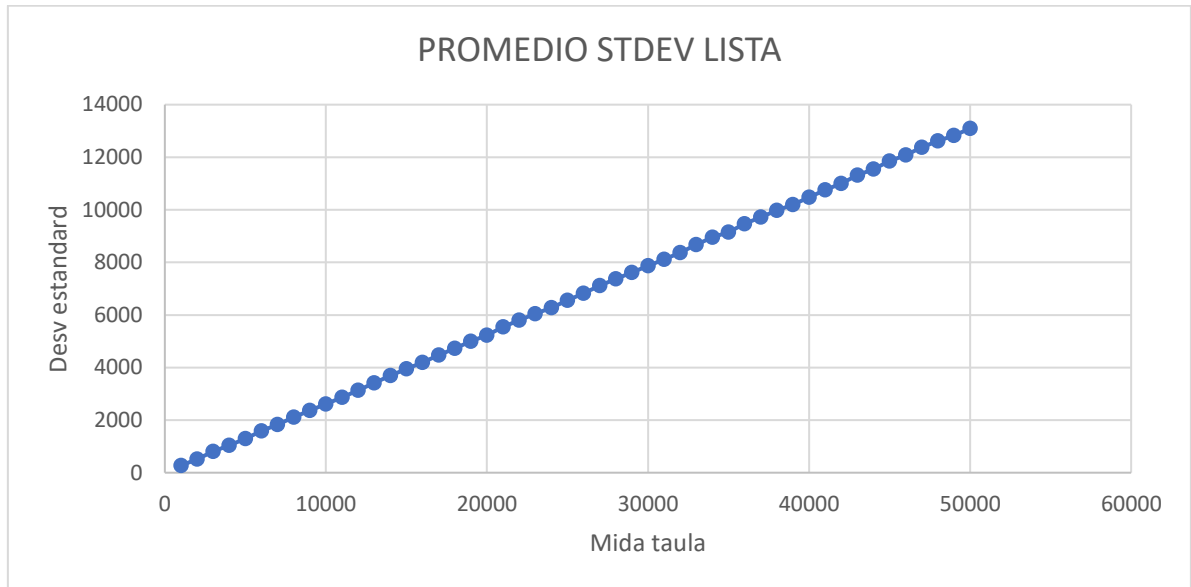
Taula de hash

El cost és constant i com a molt varia 1 o 2 accesos.

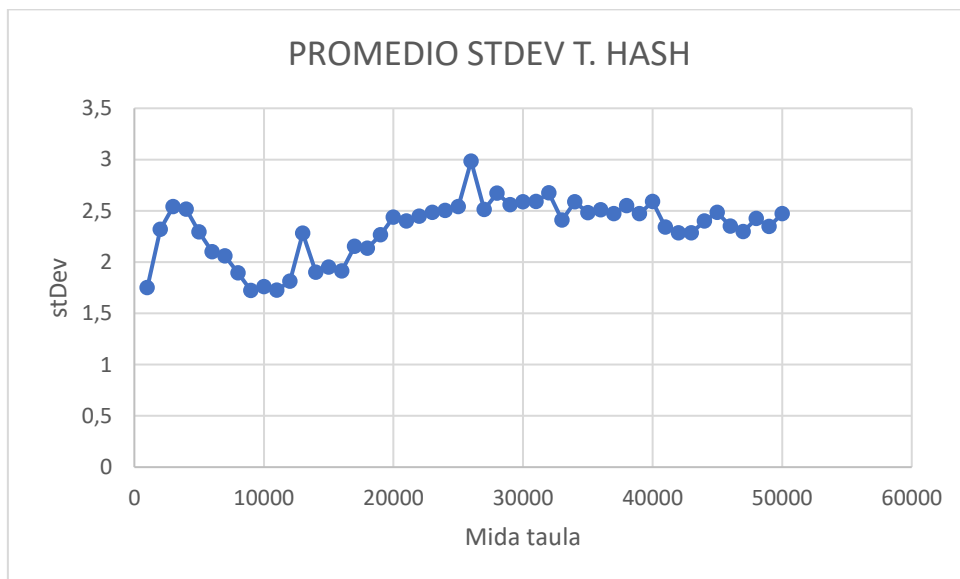


STDEV:

Llista



Taula de hash



Codi Font

Taula de Hash

```
package Data;
import java.io.*;

import Exceptions.*;
public class HashTable <K,T extends Comparable<T>>implements
TADTaulaHash<K,T>{
    int tableSize=10;
    Nodo<K,T>[] tablaHash;
    int nElems;
    public HashTable(int nElems) {
        tablaHash=new Nodo[nElems];
        tableSize=nElems;
    }
    /*
     * Constructor principal de la clase HashTable
     */
    public HashTable() {
        Crear();
    }
    public int getIndex(K key) {
        int hash=hash(key);
        return hash%tablaHash.length;
    }

    /**
     * Method used to return an object's hash
     * @param key key from which we want to obtain the hash
     * @return hash of the key passed by parameter
     */
    public int hash(K key) {

        String str=key.toString();
        int hash=0;//=key.hashCode();

        for(int i=1;i<str.length();i++){

            hash+=((int)str.charAt(i))*(32+i);
        }
        hash=hash < 0 ? hash * -1 : hash;
        return hash;
    }

    /**
     * Alternative method used to return an object's hash
     * @param key object from we want to obtain the hash
     * @return hash of the object passed by parameter
     */
    public int hashKey(K key) {
        String str=key.toString();
        int res=0;
        for(int i=0;i<str.length();i++){
            res+=str.charAt(i)*Math.pow(32,i);
        }
        res=res < 0 ? res * -1 : res;
    }
}
```

```

        return res;
    }

    @Override
    public void Crear() {
        tablaHash=new Nodo[tableSize];
        nElems=0;
    }

    @Override
    public void Inserir(K key, T data) {

        if(factorCarga()>=0.75){
            //printTable();
            try{
                resize();
            }catch(ElementoNoEncontrado ignored){
            }
        }
        int hash=hash(key);
        int index=getIndex(key);
        if(tablaHash[index]==null){
            tablaHash[index]=new Nodo<>(key,data,hash);
            nElems++;
        }
        else{
            try{
                int offset=Buscar(key);
                replace(index,offset,data);
            }catch (ElementoNoEncontrado e){
                tablaHash[index].add(key,data,hash);
            }
        }

    }

    @Override
    public T Obtener(K key) {
        return null;
    }

    @Override
    public int Buscar(K key) throws ElementoNoEncontrado{
        int hash=hash(key);
        int index=hash%tablaHash.length;
        int c=0;
        Nodo<K,T>temp=tablaHash[index];
        if(temp!=null){
            while(temp!=null){
                if(temp.key.equals(key)){
                    return c;
                }
                temp=temp.nextCol;
                c++;
            }
        }
        throw new ElementoNoEncontrado(c);
    }
}

```

```

@Override
public int Mida() {
    return tablaHash.length;
}

/**
 * Mètode que esborra un element en cas que existeixi
 * @param key clau de l'element
 * @throws ElementoNoEncontrado throws exception if element isn't
found
 */
@Override
public void Esborrar(K key) throws ElementoNoEncontrado {
    int hash=hash(key);
    int index=getIndex(key);
    Nodo<K, T> elem=tablaHash[index];
    if(elem!=null){
        try {
            int offset = Buscar(key);
            if (offset == 0) {
                elem=elem.nextCol;
            } else {
                while (elem.nextCol.hash != hash) {
                    elem = elem.nextCol;
                }
                elem.nextCol = elem.nextCol.nextCol;
            }
        } catch (ElementoNoEncontrado e) {
            System.out.println(e.getMessage());
        }
    }
    else{
        throw new ElementoNoEncontrado(0);
    }
}

public ListaDoble<K,T>ObtenirValors(){
    Nodo<K,T> aux;
    ListaDoble<K,T> listaAux=new ListaDoble<>();
    for (Nodo<K, T> hash : tablaHash) {
        aux = hash;
        while (aux != null) {
            listaAux.Inserir(aux.getData());
            aux = aux.nextCol;
        }
    }
    return listaAux;
}

public ListaDoble<K,T>ObtenirClaus(){
    Nodo<K,T>aux;

    ListaDoble<K,T> listaAux=new ListaDoble<>();
    for (Nodo<K, T> hash : tablaHash) {
        aux = hash;
        while (aux != null) {

            listaAux.Inserir((T) aux.key);
            aux = aux.nextCol;
        }
    }
}

```

```

        return listaAux;
    }

    /**
     * Método que calcula el factor de carga para saber si hay que
     * redimensionar la tabla de hash
     * @return true si hay que redimensionar
     */
    public float factorCarga() {

        return (float)nElems/tablaHash.length;
    }

    /**
     * Método encargado de redimensionar la tabla de hash y recalcular
     * todos los hashes de nuevo
     */
    public void resize() throws ElementoNoEncontrado {
        //Nodo[] listaAux=new Nodo[tabelaHash.length*2];
        HashTable<K,T> tablaAux=new HashTable<>(tablaHash.length*2);
        Nodo<K,T> temp;
        K key;
        T data;
        for (Nodo<K, T> hash : tablaHash) {
            temp = hash;
            while (temp != null) {
                key = temp.getKey();
                data = temp.getData();
                tablaAux.Inserir(key, data);
                temp = temp.nextCol;
            }

            }
        tablaHash=tablaAux.tablaHash;
        tableSize=tablaHash.length;

    }

    /**
     * Mètode auxiliar per a sobreesciure un valor en cas de que la
     * clau ja existeixi
     * @param index index de la taula de hahs
     * @param offset número de la col·lisió
     * @param data nou valor a assignar
     */
    public void replace(int index,int offset,T data){
        Nodo node=tablaHash[index];
        int i=0;
        while(i!=offset&&node!=null) {
            node=node.nextCol;
            i++;
        }
        try{
            node.setData(data);
        }catch(NullPointerException e){

        }

    }

}

```

```

/**
 * METHOD USED TO WRITE A FILE WITH ALL THE VALUES THAT CONTAINS
 * THE HASH TABLE
 */
public void writeFile () {

    String fileName;
    FileWriter escribir=null;
    int nElems=tablaHash.length;
    Nodo temp=null;
    try {

        fileName="hashCodes.txt";
        escribir=new FileWriter(fileName);
    } catch (IOException e) {
        System.out.println(e.getMessage());
    }
    // TODO Auto-generated catch block
    for(int i=0;i<nElems;i++) {
        try {
            if(tablaHash[i]!=null) {
                temp=tablaHash[i];
                while(temp!=null) {
                    escribir.write("key= "+i+"; "+temp.data+";
hash= "+temp.hash+"\n");
                    //escribir.write();
                    escribir.flush();
                    temp=temp.nextCol;
                }
            }
        } catch (NullPointerException e) {
            System.out.println(e.getMessage());
        } catch (IOException e) {
            throw new RuntimeException(e);
        }
    }
    try {
        escribir.close();
    } catch (IOException e) {
        throw new RuntimeException(e);
    }

}

}

```

Main

```

package Programa;

import java.io.*;
import java.util.*;

import Data.*;
import Exceptions.ElementoNoEncontrado;

public class main {
    public static FileWriter file;
    public static Scanner scan;

```



```

    public static void main(String[] args) throws InterruptedException,
IOException {
        // PROGRAMA PRINCIPAL PART HASHINGS

        HashTable<String,Ciutada>tablaAux=new HashTable<>();

        Ciutada carlos=new Ciutada("Carlos","Martinez","49424598J");
        Ciutada david=new Ciutada("David","Marti","7771391023");
        Ciutada nil=new Ciutada("Carlos","Martinez","44548898T");
        Ciutada genis=new Ciutada("Genis","Martinez","73981391P");
        Ciutada roger=new Ciutada("Roger","Massana","3731918T");
        Ciutada lluis=new Ciutada("Lluis","Gallart","7137391890");
        Ciutada gerard=new Ciutada("Gerard","Panisello","3241233Y");
        Ciutada eros=new Ciutada("Eros","Villar","1413133T");

        try{
            tablaAux.Inserir("49424598J",carlos);
            tablaAux.Inserir("7771391023",david);
            tablaAux.Inserir("44548898T",nil);
            tablaAux.Inserir("73981391P",genis);
            tablaAux.Inserir("3731918T",roger);
            tablaAux.Inserir("7137391890",lluis);
            tablaAux.Inserir("3241233Y",gerard);
            tablaAux.Inserir("1413133",eros);

            System.out.println(tablaAux.Buscar("49424598J"));
            System.out.println(tablaAux.Buscar("7771391023"));
            System.out.println(tablaAux.Buscar("44548898T"));
            System.out.println(tablaAux.Buscar("73981391P"));
            System.out.println(tablaAux.Buscar("3731918T"));
            System.out.println(tablaAux.Buscar("7137391890"));
            System.out.println(tablaAux.Buscar("3241233Y"));
            System.out.println(tablaAux.Buscar("1413133"));

        }catch(ElementoNoEncontrado e){

        }

        //Joc de proves de la taula de hash
        mostrarMenu();
        System.out.println("TABLE DONE");
        //Anàlisi de la llista doblement encadenada
        JocProvesLlista();
        System.out.println("LIST DONE");
    }

    public static void mostrarMenu() throws IOException {

        int searchElems;

        System.out.println("BENVINGUT/UDA AL PROGRAMA PRINCIPAL");
        System.out.println("A continuació s'executaran els següents
mètodes de manera automàtica:");
        System.out.println("-Després de finalitzar cada inerció dels

```

```

elements a la taula de hash anirem buscant" +
    " números generats aleatoriament");
    System.out.println("-Un cop generades totes les taules de hash i
calculats els seus costos d'accès," +
    "generarem un fitxer que recollirà un anàlisi del cost mig
i desviació estandard tenint en compte" +
    "els factors mencionats previament.");
    System.out.println("COMENCEM...\n");
    //t.sleep(3500);

    int nElems=1000;
    HashTable<Integer,Integer> numbers;
    int[] digits;
    int i=0;
    ArrayList<ArrayList<Integer>> llistaAux=new ArrayList<>();
    do {
        System.out.println("nElems= "+nElems);
        llistaAux.add(new ArrayList<>());
        numbers=new HashTable<>(nElems);
        digits = new int[nElems];

        for(int k=0;k<nElems;k++) {
            digits[k] = randomInt(nElems/2);

            numbers.Inserir(digits[k],digits[k]);
        }

        String fileName="Analisi/LogCerques/"+nElems+"searches.txt";
        PrintStream output = null;
        try {
            output = new PrintStream(new FileOutputStream(fileName));
        } catch (FileNotFoundException e) {
            throw new RuntimeException(e);
        }
        System.setOut(output);
        //Escribimos en un fichero los elementos que encuentra y
        los que no, para poder visualizarlo más comodamente

        for(int j=0;j<nElems;j++) {

            try{
                searchElems=numbers.Buscar(randomInt(numbers.Mida()/2));
                llistaAux.get(i).add(searchElems);
                //totalSearches[i]+=searchElems;
                System.out.println(searchElems+" iteration until
element has been found");

            } catch (ElementoNoEncontrado e) {
                System.out.println(e.getMessage());
            }

        }
        System.setOut(new PrintStream(new
FileOutputStream(FileDescriptor.out)));
        nElems+=1000;i++;
    }while(nElems<=50000);
    numbers.writeFile();
    scan=new Scanner(System.in);
    System.out.println("Introduce el numero a buscar");

```

```

        int n=scan.nextInt();
        try{
            System.out.println(numbers.Buscar(n));

        }catch(ElementoNoEncontrado e){

        }

        //Escritura del fitxer
        FileWriter analisis=new
FileWriter("Analisi/CostCompuTaula.csv");
        analisis.write("MIDA;"+"N ACCESSOS;"+"DESV EST\n");
        int Elems=1000;
        for (ArrayList<Integer> lista : llistaAux) {
            analisis.write(Elems + ";" + mean(lista) + ";" +
stDev(lista)+"\n");
            Elems += 1000;
        }
        analisis.close();

    }

    public static void JocProvesLlista() {
        ListaDoble<Integer,Integer>lista=new ListaDoble<>();
        int nElems=1000;
        int[] digits;
        int searchElems=0;
        ArrayList<ArrayList<Integer>>llistaAux=new ArrayList<>();
        int i=0;
        FileWriter analisis=null;
        try {
            analisis = new FileWriter("Analisi/CostCompuLlista.csv");
            analisis.write("MIDA;"+"N ACCESSOS;"+"DESV EST\n");
        }catch(IOException e) {
        }
        do{
            lista=new ListaDoble<>();
            System.out.println("nElems= "+nElems);
            llistaAux.add(new ArrayList<>());
            digits=new int[nElems];
            for(int j=0; j<nElems; j++){
                digits[j]=randomInt(nElems/2);
                lista.Inserir(digits[j]);
            }
            for(int k=0;k<nElems;k++){
                try{
                    searchElems=lista.Buscar(randomInt(nElems/2));
                    llistaAux.get(i).add(searchElems);
                }catch(ElementoNoEncontrado e){
                    //System.out.println(e.getMessage());
                }
            }

            try {

                analisis.write(nElems + ";" + mean(llistaAux.get(i)) + ";" +
+ stDev(llistaAux.get(i)) + "\n");
            }
            catch(NullPointerException e){
                System.out.println(e.getMessage());
            }
        }
    }

```

```

        } catch (IOException e) {
            System.out.println("FILE ERROR");
        }
        i++;
        nElems+=1000;
    }while(nElems<=50000);
    try{
        analisis.close();

    }catch(IOException e){

    }

}

public static double mean(ArrayList<Integer> lista){
    int sum=0;
    for(int temp: lista){
        sum+=temp;
    }
    return (double)sum/ lista.size();
}

public static double stDev(ArrayList<Integer>lista) {
    int[] nums=new int[lista.size()];
    for(int i=0;i<lista.size();i++){
        nums[i]=lista.get(i);
    }
    double stDev=0.0,sum=0.0;
    for(int i=0;i<nums.length&&nums[i]!=0;i++) {
        sum+=nums[i];
    }
    double media=sum/nums.length;
    for(int j=0;j<nums.length&&nums[j]!=0;j++) {
        stDev+=Math.pow(nums[j]-media, 2);
    }
    double sq = stDev / nums.length;
    stDev = Math.sqrt(sq);
    return stDev;
}

public static void separator() {

System.out.println("*****
*****");
    }

    /**
     * Método que se encarga de pedir un elemento y buscarlo en la
     * lista pasada como parámetro
     * @param table tabla en la que buscaremos el elemento que se
     * introduzca
     */
    public static void searchElement(HashTable table) {
        String data= "";
        do {
            System.out.println("Escriu l'element que vulguis buscar: ");
            scan=new Scanner(System.in);
            try {
                data=scan.next();
            }

```

```

        //else
        System.out.println(table.Buscar(data));
    } catch (NumberFormatException e) {
        System.out.println("EL número introduit no és vàlid");
    }
    catch (ElementoNoEncontrado e) {
        System.out.println(e.getMessage());
    }
    } while (!data.equalsIgnoreCase("-1"));
}

/**
 * Genera un número entero aleatorio
 * @return random int
 */
public static int randomInt(int rightLimit) {
    int leftLimit=1;
    //int rightLimit;
    int number;

    //rightLimit=Integer.MAX_VALUE;
    //rightLimit=(leftLimit*10)-1;
    number=leftLimit+(int) (Math.random()*(rightLimit-leftLimit));
    return number;
}
}

```

Nodo

```

package Data;

public class Nodo<K,T extends Comparable<T>> implements Comparable<T>
{
    T data;
    K key;
    public Nodo<K,T> nextCol;
    public Nodo<K,T> prev;
    int hash=0;

    public Nodo(Nodo<K,T> col,T data) {
        this.data=data;
        nextCol=col;
        prev=null;
    }
    public Nodo(K key,T data,int hash){
        this.data=data;
        this.key=key;
        this.hash=hash;
        nextCol=null;
        prev=null;
    }

    public Nodo(Nodo<K,T> sig, Nodo<K,T> prev, T data) {
        this.data = data;
        this.nextCol = sig;
        this.prev = prev;
    }
}

```

```

public Nodo(T data2) {
    this(null,data2);
}
public void add(K key,T data,int hash) {
    Nodo<K,T> temp=this;
    if(temp.data==null){
        temp.data=data;
        temp.hash=hash;
    }
    else {
        while (temp.nextCol != null) {
            temp = temp.nextCol;
        }
        temp.setNextCol(new Nodo<>(key,data,hash));
    }
}

public K getKey() {
    return key;
}

public T getData() {
    return this.data;
}
public void setData(T data) {
    this.data=data;
}

public void setNextCol(Nodo<K,T> col) {
    this.nextCol=col;
}

@Override
public String toString() {
    return "Nodo{" +
        "data=" + data +
        ", nextCol=" + nextCol +
        '}';
}

@Override
public int hashCode() {
    return hash;
}

@Override
public int compareTo(T o) {
    if(o.hashCode()==hashCode()) return 0;
    return -1;
}
}

```