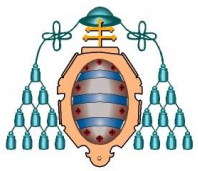
ESTRUCTURAS DICCIONARIO SESIÓN 1

Estructura de Datos

2021-2022

María del Rosario Suárez Fernández



Departamento de Informática
UNIVERSIDAD DE OVIEDO

Proyecto4 – Tablas Hash

- Crear un nuevo proyecto llamado **TablasHash**
- Crear la clase **HashNode** → implementa un elemento de una tabla hash

```
public class HashNode <T>
```

Clase HashNode - Propiedades

Propiedad	Descripción
`private T info;`	En contenido de un elemento de la tabla hash
`private int estado`	Estado del elemento de la tabla hash: borrado, lleno o vacío

Info estado

Constate	Valor
`public static final int BORRADO`	-1
`public static final int VACIO`	0
`public static final int LLENO`	1

Clase HashNode - Métodos

- `public HashNode()`
- `public T getInfo()`
- `public void setInfo(T elemento)`
- `public void remove ()`
- `public int getStatus()`
- `public String toString()`

Clase HashNode - toString

```
public String toString() {
    StringBuilder cadena = new StringBuilder("{");
    switch(getStatus()){
        case LLENO:
            cadena.append(info.toString());
            break;
        case VACIO:
            cadena.append("_E_");
            break;
        case BORRADO:
            cadena.append("_D_");
    }
    cadena.append("}");
    return cadena.toString();
}
```

Ejemplos

{8}

{_E_}

{_D_}

Proyecto4 – Tablas Hash

- Crear la clase **AbstractHash** → se definen las cabeceras de los métodos cuya implementación dependerá de la filosofía de tablas hash (abiertas o cerradas) y otro métodos comunes a cualquier implementación

```
public abstract class AbstractHash<T>
```

Clase AbstractHash – Cabeceras de métodos

- `abstract public int getNumOfElems()`
- `abstract public int getSize()`
- `abstract public int add(T elemento)`
- `abstract public T find(T elemento)`
- `abstract public int remove(T elemento)`
- `abstract public String toString()`

Clase AbstractHash – Métodos comunes

- `protected int fHash(T elemento)`

```
protected int fHash(T elemento) {
    int pos = elemento.hashCode()%getSize();
    if (pos < 0) return pos+getSize();
    else return pos;
}
```

- `protected boolean isPositivePrime(int numero)`
- `protected int nextPrimeNumber(int numero)`
- `protected int previousPrimeNumber(int numero)`

Proyecto4 – Tablas Hash

- Crear la clase **ClosedHashTable** → implementa una tabla hash cerrada

```
public class ClosedHashTable<T> extends AbstractHash<T>
```

10		12		14
LLENO	VACIO	LLENO	VACIO	BORRADO

Clase ClosedHashNode - Propiedades

Propiedad	Descripción
`private int numElementos;`	Número de elementos insertados hasta el momento
`private HashNode<T> tabla[];`	Tabla hash
`private int tipoExploracion;`	1-Lineal 2-Cuadrática 3-Dispesión doble

10		12		14
LLENO	VACIO	LLENO	VACIO	BORRADO

Clase ClosedHashTable - Métodos

- `public ClosedHashTable(int tam, int tipo)`
- `public int getNumOfElems();`
- `public int getSize();`
- `public int add(T elemento);`
- `public T find(T elemento);`
- `public int remove(T elemento);`
- `public String toString();`

Clase ClosedHashTable - toString

```
public String toString() {
    StringBuilder cadena = new StringBuilder();
    for(int i=0;i< getSize();i++){
        cadena.append(table[i].toString());
        cadena.append(";");
    }
    cadena.append("[Size: ");
    cadena.append(getSize());
    cadena.append(" Num.Elems.: ");
    cadena.append(getNumOfElements());
    cadena.append("]");
    return cadena.toString();
}
```

10		12		14
LLENO	VACIO	LLENO	VACIO	BORRADO

{10};{_E_};{12};{_E_};{_D_};[Size: 5 Num.Elems.: 2]

ClosedHashTable – Redispersión

- Añadir la clase abstracta **AbstractHash** los siguietes métodos
  - `abstract protected booelan reDispersion()`
  - `abstract protected boolean inservseRedispersion()`
- En la clase **ClosedHashTable** añadir
  - las constantes

Constate	Valor
`private static final double MINIMUN_LF`	0.16
`private static final double MAXIMUN_LF`	0.5

  - las propiedades

Propiedad	Descripción
`private double minlf;`	Factor de carga mínimo
`private double maxlf;`	Factor de carga máximo

ClosedHashTable – Redispersión

- Modificar el constructor para asignar a las propiedades minlf y maxlf, el valor de las constantes definidas.
- Añadir un nuevo constructor parametrizable al máximo
  - `public ClosedHashTable(int tam, int tipo, double  minlf, double maxlf);`
- Implementar los métodos en la clase **ClosedhashTable**

Tareas para casa (Evaluación continua)

- Terminar los métodos de la clase ClosedHashTable funcionando
- Elaborar un conjunto de baterías de test que prueben:
  - El método añadir con todos los casos posibles de error
  - El método borrar con todos los casos posibles de error
- Documentar las clases con JavaDoc