TDA:Cifras

Generated by Doxygen 1.9.2

Chapter 1

Class Index

1.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 2

File Index

2.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 3

Class Documentation

3.1 Busqueda Class Reference

Public Member Functions

- **Busqueda** ()

  Constructor del árbol n-ario, construye un árbol vacío.
- ∼**Busqueda** ()

  Destructor del árbol n-ario.
- void construir (vector< int > conjunto)

  Construye el árbol con la estructura necesaria a las operaciones y al conjunto. A partir del pasarle el conjunto.
- void insertarhijoizqda (Celda *T1, Celda *n, Celda *T2)

  Insertar un hijo a la izquierda. Necesario en método construir.
- Celda * PodarHijoIzq (Celda *T, Celda *n)

  Podar hijo a la izq. Necesario para descartar las no-soluciones, de forma que cuando se encuentre la solución solo haya que listarlo en preorden.
- void **insertarhermdrcha** ()

  Insertar hermano a la derecha.
- void **ListarPreorden** (Celda *T)

  Lista el árbol por preorden.
- vector< vector< char > > busca (int num)

  Busca el número en el árbol y devuelve todas las operaciones asociadas, siguiendo los caminos del árbol desde la raíz hasta el nodo localizado (al haber quitado con otros métodos las no-soluciones anteriores al encontrar la solución llegar a la solución es tan simple como recorrer en preorden hasta llegar al número deseado, esas operaciones serán la solución a almacenar).

3.1.1 Detailed Description

Definition at line 22 of file Busqueda.h.

3.1.2 Member Function Documentation

3.1.2.1 busca()

```
vector< vector< char > > Busqueda::busca (
            int num )
```

Busca el número en el árbol y devuelve todas las operaciones asociadas, siguiendo los caminos del árbol desde la raíz hasta el nodo localizado (al haber quitado con otros métodos las no-soluciones anteriores al encontrar la solución llegar a la solución es tan simple como recorrer en preorden hasta llegar al número deseado, esas operaciones serán la solución a almacenar).

Parameters

num	El número solución

Returns

Vector de vectores de char con las operaciones que se han empleado. Ordenadas desde la primera hasta la última.

3.1.2.2 construir()

```
void Busqueda::construir (
            vector< int > conjunto )
```

Construye el árbol con la estructura necesaria a las operaciones y al conjunto. A partir del pasarle el conjunto.

Parameters

conjunto	Conjunto de números

3.1.2.3 insertarhijoizqda()

```
void Busqueda::insertarhijoizqda (
            Celda * T1,
            Celda * n,
            Celda * T2 )  [inline]
```

Insertar un hijo a la izquierda. Necesario en método construir.

Definition at line 53 of file Busqueda.h.

```
00053                                                    {
00054         if(T2 != 0)
00055         {
00056             T2->hermdrcha=n->hijoizq;
00057             T2->padre=n;
00058             T2=0;
00059         }
00060     }
```

3.1.2.4 PodarHijoIzq()

```
Celda * Busqueda::PodarHijoIzq (
            Celda * T,
            Celda * n )  [inline]
```

Podar hijo a la izq. Necesario para descartar las no-soluciones, de forma que cuando se encuentre la solución solo haya que listarlo en preorden.

Returns

Hijo podado

Definition at line 65 of file Busqueda.h.

```
00065                                           {
00066         Celda *Res=0;
00067         if(n->hijoizq !=0){
00068             Res=n->hijoizq;
00069             n->hijoizq=Res->hermdrcha;
00070             Res->padre=Res->hermdrcha=0;
00071             }
00072         return Res;
00073
00074     }
```

The documentation for this class was generated from the following file:

- include/Busqueda.h

3.2 Cifras Class Reference

T.D.A. Cifras.

```
#include <cifras.h>
```

Public Member Functions

- Cifras ()

  Constructor.
- ~Cifras ()

  Destructor.
- void resize (int tam)

  Redimensiona el vector de cifras.
- int getTam ()

  Obtiene el tamaño del vector.
- int getAtPos (int pos)

  Obtiene un valor de una posición concreta del vector.
- void in (int numero)

  Introduce un número al vector.
- vector< int > out (int cant)

  Obtener sub-conjunto aleatoriamente. Crea un subvector donde introduce elementos del otro, utiliza la función rand, para obtener una posicion de todo el tamaño del vector asi una vez saca una posición aleatoria saca un número aleatorio, se actualizan las posiciones.
- void search (int resultado, vector< int > &conjunto)

  Obtiene todas las operaciones con el subconjunto que dan el resultado pedido y las imprime por pantalla. En él, se utiliza una instancia del TDA Búsqueda (árbol n-ario) que será el soporte para realizar la búsqueda. Desde este método, se mostrarán directamente por pantalla estándar las soluciones.

3.2.1 Detailed Description

T.D.A. Cifras.

Una instancia *c* del tipo de datos abstracto `Cifras` es un objeto que almacena el conjunto de unos números y permite operar con ellos de una forma específica.

Author

Yeray López Ramírez y Jaime Castillo Uclés

Date

Diciembre 2021

Definition at line 26 of file cifras.h.

3.2.2 Constructor & Destructor Documentation

3.2.2.1 Cifras()

```
Cifras::Cifras ( )  [inline]
```

Constructor.

Definition at line 33 of file cifras.h.

```
00033             {
00034         numeros.clear();
00035     }
```

3.2.2.2 ~Cifras()

```
Cifras::~Cifras ( )  [inline]
```

Destructor.

Definition at line 39 of file cifras.h.

```
00039              {
00040         numeros.clear();
00041     }
```

3.2.3 Member Function Documentation

3.2.3.1 getAtPos()

```
int Cifras::getAtPos (
            int pos )
```

Obtiene un valor de una posición concreta del vector.

Parameters

pos	Posición

Returns

El valor de la posición

Definition at line 26 of file cifras.cpp.

```
00026                                {
00027         return numeros[pos];
00028     }
```

3.2.3.2 getTam()

```
int Cifras::getTam ( )
```

Obtiene el tamaño del vector.

Returns

Tamaño del vector

Definition at line 23 of file cifras.cpp.

```
00023                        {
00024         return numeros.size();
00025     }
```

3.2.3.3 in()

```
void Cifras::in (
             int numero )
```

Introduce un número al vector.

Parameters

numero	Número a meter

Definition at line 29 of file cifras.cpp.

```
00029                                {
00030         numeros.push_back(numero);
00031     }
```

3.2.3.4 out()

```
vector< int > Cifras::out (
             int cant )
```

Obtener sub-conjunto aleatoriamente. Crea un subvector donde introduce elementos del otro, utiliza la función rand, para obtener una posicion de todo el tamaño del vector asi una vez saca una posición aleatoria saca un número aleatorio, se actualizan las posiciones.

Parameters

cant	Cantidad de números en el subconjunto

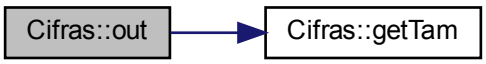
Returns

Vector de números aleatorios

Definition at line 33 of file cifras.cpp.

```
00033                                   {
00034         vector<int> los_elegidos;
00035         srand(time(NULL));
00036
00037         for(int i=0; i<cant; ++i){
00038             int posicion_elegida = rand() % getTam();
00039             int elegido = numeros[posicion_elegida];
00040             numeros.erase(numeros.begin() + posicion_elegida);
00041             los_elegidos.push_back(elegido);
00042
00043         }
00044
00045         return los_elegidos;
00046     }
```

Here is the call graph for this function:



3.2.3.5 resize()

```
void Cifras::resize (
             int tam )
```

Redimensiona el vector de cifras.

IMPLEMENTACIÓN

Definition at line 20 of file cifras.cpp.

```
00020                                 {
00021         numeros.resize(tam);
00022     }
```

3.2.3.6 search()

```
void Cifras::search (
            int resultado,
            vector< int > & conjunto )
```

Obtiene todas las operaciones con el subconjunto que dan el resultado pedido y las imprime por pantalla. En él, se utiliza una instancia del TDA Búsqueda (árbol n-ario) que será el soporte para realizar la búsqueda. Desde este método, se mostrarán directamente por pantalla estándar las soluciones.

Parameters

resultado	Resultado pedido
conjunto	Subconjunto de números

Definition at line 48 of file cifras.cpp.

```
00048                                                   {
00049     }
```

The documentation for this class was generated from the following files:

- include/cifras.h
- src/cifras.cpp

Chapter 4

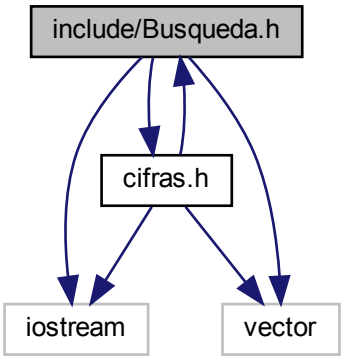
File Documentation

4.1 include/Busqueda.h File Reference

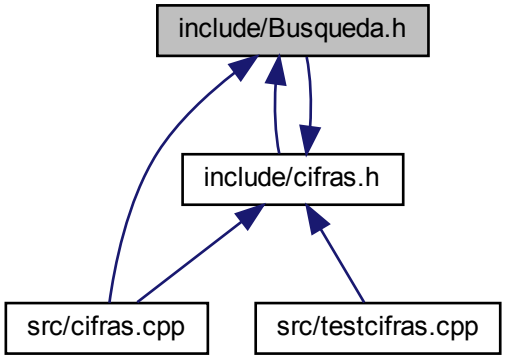
Cabecera del TDA Busqueda.

```
#include <iostream>
#include <cifras.h>
#include <vector>
```

Include dependency graph for Busqueda.h:



This graph shows which files directly or indirectly include this file:



Classes

- class Busqueda

4.1.1 Detailed Description

Cabecera del TDA Busqueda.

Definition in file Busqueda.h.

4.2 Busqueda.h

Go to the documentation of this file.

```
00001
00005 #ifndef _BUSQUEDA_H
00006 #define _BUSQUEDA_H
00007
00008 #include <iostream>
00009 #include <cifras.h>
00010 #include <vector>
00011
00012 using namespace std;
00013
00014
00022 class Busqueda{
00023 private:
00028     struct Celda{
00029         int etiqueta;
00030         Celda *padre, *hijoizq, *hermdrcha;
00031     };
00032
00033
00034 public:
00038     Busqueda();
00042     ~Busqueda();
00047     void construir(vector<int> conjunto);
00048
00049     // Funciones de uso //
00053     void insertarhijoizqda(Celda *T1, Celda *n, Celda *T2){
00054         if(T2 != 0)
```

```
00055         {
00056             T2->hermdrcha=n->hijoizq;
00057             T2->padre=n;
00058             T2=0;
00059         }
00060     }
00065     Celda *PodarHijoIzq(Celda *T, Celda *n){
00066         Celda *Res=0;
00067         if(n->hijoizq !=0){
00068             Res=n->hijoizq;
00069             n->hijoizq=Res->hermdrcha;
00070             Res->padre=Res->hermdrcha=0;
00071             }
00072         return Res;
00073
00074     }
00079     void insertarhermdrcha();
00083     void ListarPreorden(Celda *T);
00090     vector<vector<char» busca(int num);
00091 };
00092 #endif
```

4.3 cifras.h

```
00001
00005 #ifndef _CIFRAS_H
00006 #define _CIFRAS_H
00007
00008 #include <iostream>
00009 #include <vector>
00010 #include "Busqueda.h"
00011 using namespace std;
00012
00013
00026 class Cifras{
00027 private:
00028     vector<int> numeros;
00029 public:
00033      Cifras(){
00034         numeros.clear();
00035      }
00039      ~Cifras(){
00040         numeros.clear();
00041      }
00045     void resize(int tam);
00050     int getTam();
00056     int getAtPos(int pos);
00061     void  in(int numero);
00068     vector<int> out(int cant);
00074     void search(int resultado, vector<int> &conjunto);
00075 };
00076 #endif
```

4.4 cifras.cpp

```
00001
00007 #include <iostream>
00008 #include <vector>
00009 #include <random>
00010 #include "cifras.h"
00011 #include "Busqueda.h"
00012
00013 using namespace std;
00014
00015
00020     void Cifras::resize(int tam){
00021         numeros.resize(tam);
00022     }
00023     int Cifras::getTam(){
00024         return numeros.size();
00025     }
00026     int Cifras::getAtPos(int pos){
00027         return numeros[pos];
00028     }
00029     void Cifras::in(int numero){
00030         numeros.push_back(numero);
00031     }
00032
00033     vector<int> Cifras::out(int cant){
```

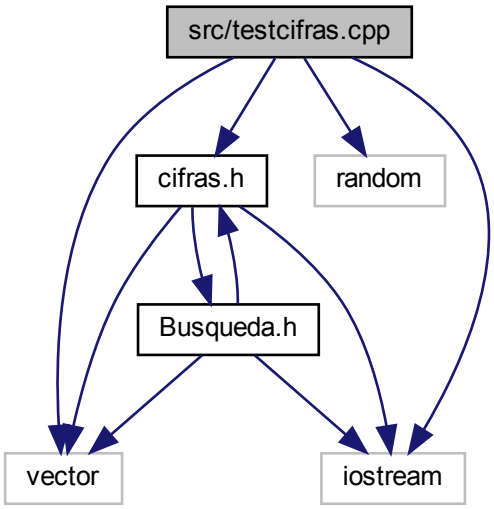
```
00034         vector<int> los_elegidos;
00035         srand(time(NULL));
00036 
00037         for(int i=0; i<cant; ++i){
00038             int posicion_elegida = rand() % getTam();
00039             int elegido = numeros[posicion_elegida];
00040             numeros.erase(numeros.begin() + posicion_elegida);
00041             los_elegidos.push_back(elegido);
00042 
00043         }
00044 
00045         return los_elegidos;
00046     }
00047 
00048     void Cifras::search(int resultado, Cifras &conjunto){
00049     }
```

4.5 src/testcifras.cpp File Reference

Test para probar el programa cifras.

```
#include <iostream>
#include <vector>
#include <random>
#include "cifras.h"
```

Include dependency graph for testcifras.cpp:



Functions

- int main (int argc, char ∗argv[ ])

4.5.1 Detailed Description

Test para probar el programa cifras.

Definition in file testcifras.cpp.

4.5.2 Function Documentation

4.5.2.1 main()

```
int main (
              int argc,
              char * argv[] )
```

Definition at line 14 of file testcifras.cpp.

```
00014                                   {
00015     int cantidad;
00016     cout«"Introduce la cantidad de números en el conjunto"«endl;
00017     cin»cantidad;
00018
00019     Cifras numeros;
00020
00021     for(int a=0; a<numeros.getTam(); ++a){
00022         cout«numeros.getAtPos(a)«endl;
00023     }
00024     for(int i=0; i<cantidad; ++i){
00025         int num = 0;
00026         cout«"Introduce el "«i«" número del conjunto"«endl;
00027         cin»num;
00028         numeros.in(num);
00029     }
00030     cout«" Comienza el juego   "«endl;
00031     cout«" Se va a obtener un subconjunto...   "«endl;
00032     int tam;
00033     int conseguir;
00034     cout«"Dime la cantidad de números que tendrá"«endl;
00035     cin»tam;
00036
00037     vector<int> los_elegidos;
00038     los_elegidos=numeros.out(tam);
00039
00040     for(int i=0; i<tam; ++i){
00041         cout«"El "«i«"º número elegido es: "« los_elegidos[i]«endl;
00042     }
00043
00044     conseguir = 100 + rand() % 900; //Número entre 100 y 999.
00045     cout«"El número que se quiere conseguir es "«conseguir«endl;
00046
00047     cout«" Procesando...       "«endl;
00048     numeros.search(conseguir,numeros);
00049     cout«" Fin del juego       "«endl;
00050
00051 }
```

4.6 testcifras.cpp

Go to the documentation of this file.

```
00001
00007 #include <iostream>
00008 #include <vector>
00009 #include <random>
00010 #include "cifras.h"
00011
00012 using namespace std;
00013
00014 int main( int argc, char *argv[]){
00015     int cantidad;
00016     cout«"Introduce la cantidad de números en el conjunto"«endl;
00017     cin»cantidad;
00018
00019     Cifras numeros;
00020
00021     for(int a=0; a<numeros.getTam(); ++a){
00022         cout«numeros.getAtPos(a)«endl;
00023     }
00024     for(int i=0; i<cantidad; ++i){
00025         int num = 0;
```

```
00026         cout«"Introduce el "«i«" número del conjunto"«endl;
00027         cin»num;
00028         numeros.in(num);
00029     }
00030     cout«" Comienza el juego   "«endl;
00031     cout«" Se va a obtener un subconjunto...   "«endl;
00032     int tam;
00033     int conseguir;
00034     cout«"Dime la cantidad de números que tendrá"«endl;
00035     cin»tam;
00036 
00037     vector<int> los_elegidos;
00038     los_elegidos=numeros.out(tam);
00039 
00040     for(int i=0; i<tam; ++i){
00041         cout«"El "«i«"° número elegido es: "« los_elegidos[i]«endl;
00042     }
00043 
00044     conseguir = 100 + rand() % 900; //Número entre 100 y 999.
00045     cout«"El número que se quiere conseguir es "«conseguir«endl;
00046 
00047     cout«" Procesando...       "«endl;
00048     numeros.search(conseguir,numeros);
00049     cout«" Fin del juego       "«endl;
00050 
00051 }
```

Index