Exposito.top

AUTHOR
Versión
Miércoles, 9 de Noviembre de 2022

Tabla de contenidos

Table of contents

Indice jerárquico

Jerarquía de la clase

Esta lista de herencias esta ordenada aproximadamente por orden alfabético:

Índice de clases

Lista de clases

Lista de las clases, estructuras, uniones e interfaces con una breve descripción:

Documentación de las clases

Referencia de la Clase es.ull.esit.utilities.BellmanFord

Implementacion Algoritmo Bellman Ford.

Métodos públicos

- **BellmanFord** (int[][] distanceMatrix, int nodes, ArrayList< Integer > path)
  Constructr de clase.

- int[] **getDistances** ()
  Getter, retorna la distancia.

- int **getValue** ()
- void **solve** ()

Descripción detallada

Implementacion Algoritmo Bellman Ford.

Definicion de la clase **BellmanFord** que es un algoritmo que genera el camino mas corto en un grafo dirigido

---

Documentación del constructor y destructor

es.ull.esit.utilities.BellmanFord.BellmanFord (int *distanceMatrix*[][], int *nodes*, ArrayList< Integer > *path*)

Constructr de clase.

Parámetros

distanceMatrix	-> Matriz para almacenar distances
nodes	-> numbero de nodes
path	-> ruta final

Documentación de las funciones miembro

int[] es.ull.esit.utilities.BellmanFord.getDistances ()

Getter, retorna la distancia.

Devuelve

distances

int es.ull.esit.utilities.BellmanFord.getValue ()

Getter, retorna el valor de ruta

Devuelve

value

void es.ull.esit.utilities.BellmanFord.solve ()

@ brief Método para econtrar distancia mas corta desde el inicio

---

La documentación para esta clase fue generada a partir del siguiente fichero:

- ExpositoTOP/src/es/ull/esit/utilities/BellmanFord.java

Referencia de la Clase es.ull.esit.utilities.ExpositoUtilities

Distintas utilidades auxiliares a utilizar a lo largo del proyecto.

Métodos públicos estáticos

- static void **printFile** (String file)
  Método para imprimir archivos.

- static String **simplifyString** (String string)
  Parser para simplificar las cadenas que contienen caracteres no deseados.

- static double[][] **multiplyMatrices** (double a[][], double b[][])
  Método para multiplicar 2 matrices dobles.

- static void **writeTextToFile** (String file, String text) throws IOException
- static String **getFormat** (String string)
  Método para obtener el formato de una cadena dada.

- static String **getFormat** (double value)
  Doble para formateador de cadenas.

- static String **getFormat** (double value, int zeros)
  Doble para formateador de cadenas.

- static String **getFormat** (String string, int width)
  Método auxiliar para obtener un formato de cadena.

- static String **getFormat** (String string, int width, int alignment)
  Método auxiliar para obtener un formato de cadena.

- static String **getFormat** (ArrayList< String > strings, int width)
- static String **getFormat** (ArrayList< Integer > strings)
- static String **getFormat** (String[] strings, int width)
- static String **getFormat** (String[][] matrixStrings, int width)
- static String **getFormat** (String[] strings)
- static String **getFormat** (String[] strings, int[] width)
- static String **getFormat** (String[] strings, int[] width, int[] alignment)
- static boolean **isInteger** (String str)
  Comprobar si un número dado es entero.

- static boolean **isDouble** (String str)
  Comprobar si un número dado es el doble.

- static boolean **isAcyclic** (int[][] distanceMatrix)
  Check es un gráfico es acíclico.

- static boolean **thereIsPath** (int[][] distanceMatrix, int node)
  Comprueba si un nodo determinado es accesible.

Atributos públicos estáticos

- static final int **DEFAULT_COLUMN_WIDTH** = 10
- static final int **ALIGNMENT_LEFT** = 1
- static final int **ALIGNMENT_RIGHT** = 2

Descripción detallada

Distintas utilidades auxiliares a utilizar a lo largo del proyecto.

@detalles Esta clase implementa una serie de métodos que se utilizarán en el proyecto.

Documentación de las funciones miembro

static String es.ull.esit.utilities.ExpositoUtilities.getFormat (ArrayList< Integer > *strings*) `[static]`

Parámetros

strings	-> cadenas a analizar

Devuelve

String -> cadena formateada

static String es.ull.esit.utilities.ExpositoUtilities.getFormat (ArrayList< String > *strings*, int *width*) `[static]`

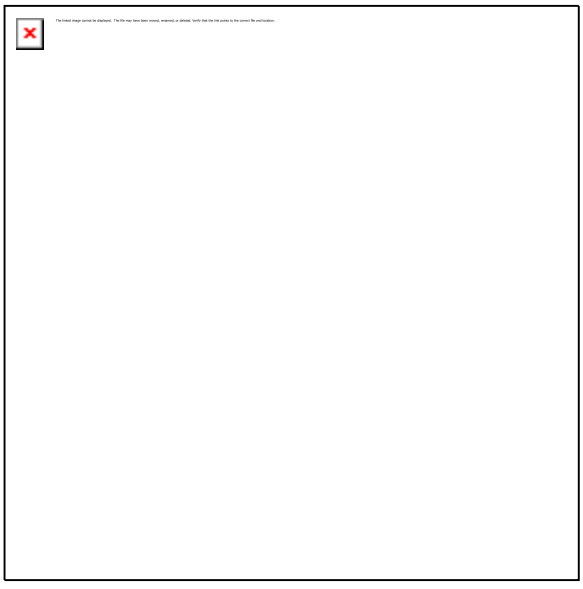
Parámetros

strings	-> cadenas a analizar
ancho	-> ancho de la cadena

Devuelve

Cadena -> formato de cadena

Gráfico de llamadas para esta función:



static String es.ull.esit.utilities.ExpositoUtilities.getFormat (double *value*) [static]

Doble para formateador de cadenas.

Parámetros

valor	-> valor a formatear

Devuelve

String -> resultado formateado

static String es.ull.esit.utilities.ExpositoUtilities.getFormat (double *value*, int *zeros*) [static]

Doble para formateador de cadenas.

Parámetros

valor	-> valor a formatear
ceros	-> precisión decimal deseada

Devuelve

String -> resultado formateado

static String es.ull.esit.utilities.ExpositoUtilities.getFormat (String *string*) [static]

Método para obtener el formato de una cadena dada.

Parámetros

string	-> cadena a analizar

Devuelve

String -> resultado del análisis

Gráfico de llamadas para esta función:

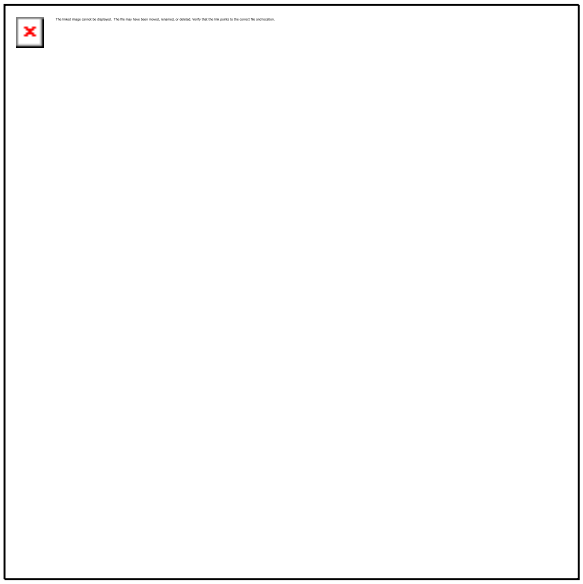
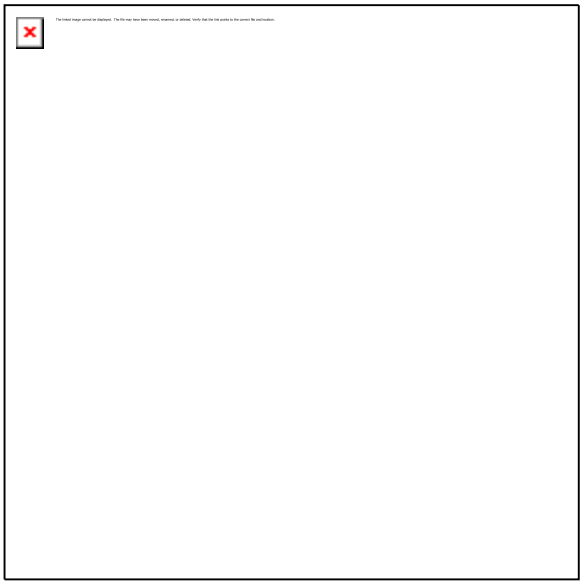


Gráfico de llamadas a esta función:



static String es.ull.esit.utilities.ExpositoUtilities.getFormat (String *string*, int *width*) `[static]`

Método auxiliar para obtener un formato de cadena.

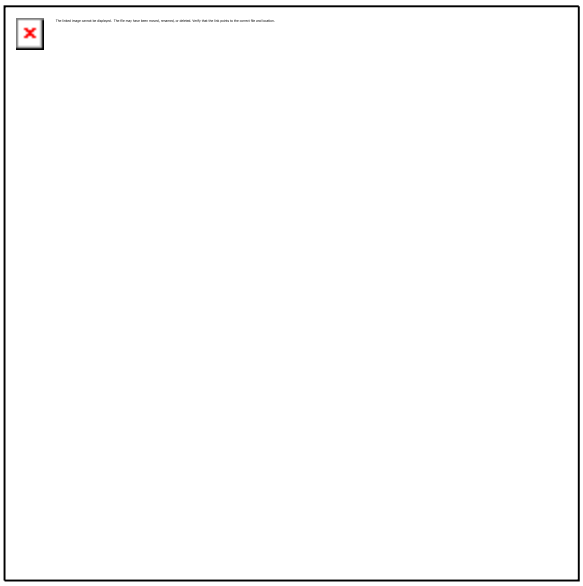
Parámetros

string	-> cadena a analizar
ancho	-> ancho de la cadena

Devuelve

Cadena -> formato de cadena

Gráfico de llamadas para esta función:



static String es.ull.esit.utilities.ExpositoUtilities.getFormat (String *string*, int *width*, int *alignment*) `[static]`

Método auxiliar para obtener un formato de cadena.

Parámetros

string	-> cadena a analizar
ancho	-> ancho de la cadena
alineación	-> alineación de cadenas

Devuelve

Cadena -> formato de cadena

static String es.ull.esit.utilities.ExpositoUtilities.getFormat (String[] *strings*) `[static]`

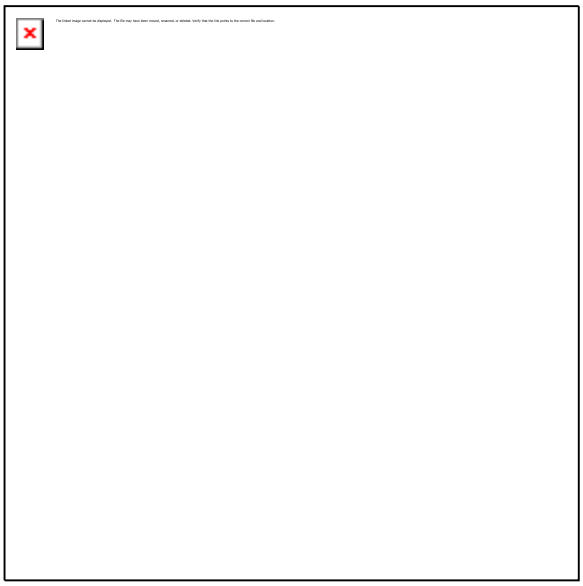
Parámetros

strings	-> Cadena de cadenas a analizar

Devuelve

Cadena -> formato de cadena

Gráfico de llamadas para esta función:



static String es.ull.esit.utilities.ExpositoUtilities.getFormat (String[] *strings*, int *width*) `[static]`

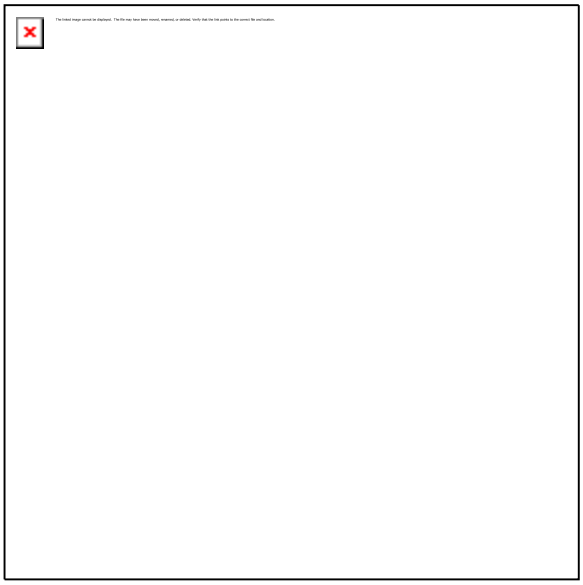
Parámetros

strings	-> cadenas a analizar
ancho	-> ancho de cadena

Devuelve

Cadena -> formato de cadena

Gráfico de llamadas para esta función:



static String es.ull.esit.utilities.ExpositoUtilities.getFormat (String[] *strings*, int[] *width*) `[static]`

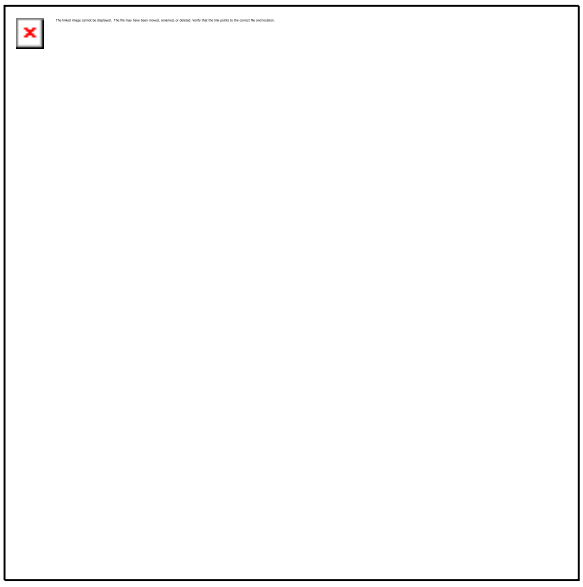
Parámetros

strings	-> Cadena de cadenas a analizar
ancho	-> ancho de cadena

Devuelve

Cadena -> formato de cadena

Gráfico de llamadas para esta función:



static String es.ull.esit.utilities.ExpositoUtilities.getFormat (String[] *strings*, int[] *width*, int[] *alignment*) `[static]`

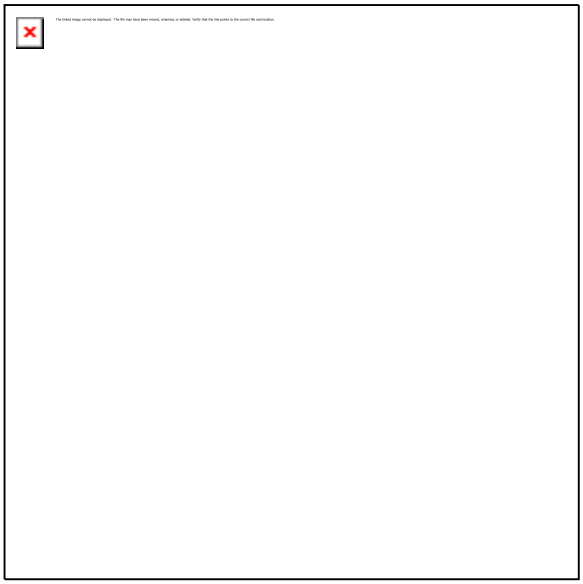
Parámetros

strings	-> Cadena de cadenas a analizar
ancho	-> ancho de cadena
alineación	-> alineación de cadenas

Devuelve

String -> formato de cadena

Gráfico de llamadas para esta función:



static boolean es.ull.esit.utilities.ExpositoUtilities.isAcyclic (int *distanceMatrix*[][]) `[static]`

Check es un gráfico es acíclico.

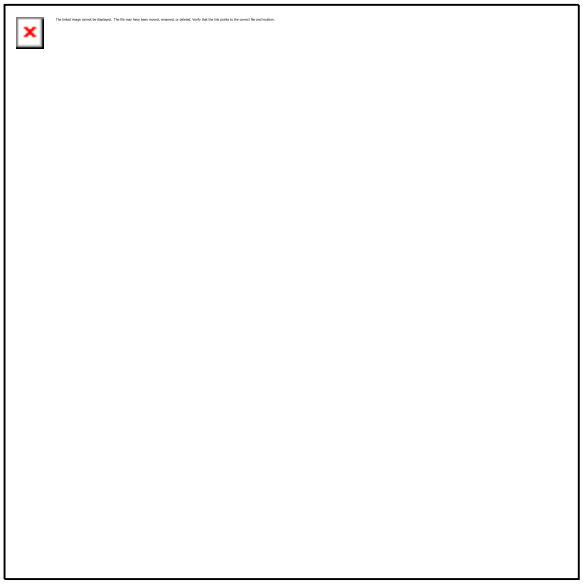
Parámetros

distanceMatrix	-> Matriz de distancias

Devuelve

booleano -> Verdadero o falso

Gráfico de llamadas para esta función:



static boolean es.ull.esit.utilities.ExpositoUtilities.isDouble (String *str*) `[static]`

Comprobar si un número dado es el doble.

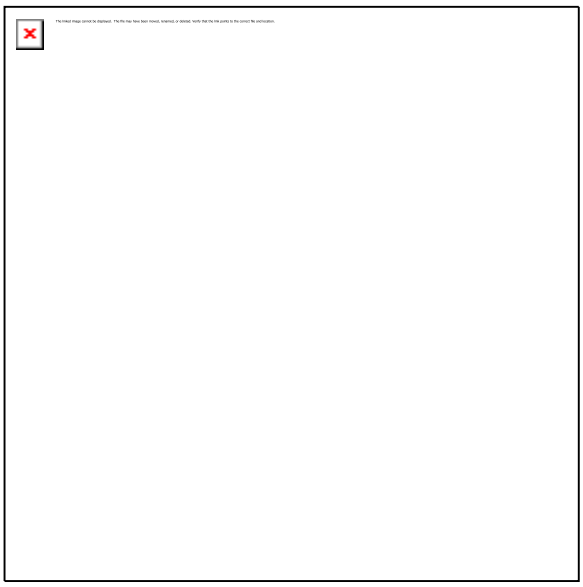
Parámetros

str	-> cadena que contiene el número

Devuelve

booleano -> Verdadero o falso

Gráfico de llamadas a esta función:



static boolean es.ull.esit.utilities.ExpositoUtilities.isInteger (String *str*) `[static]`

Comprobar si un número dado es entero.

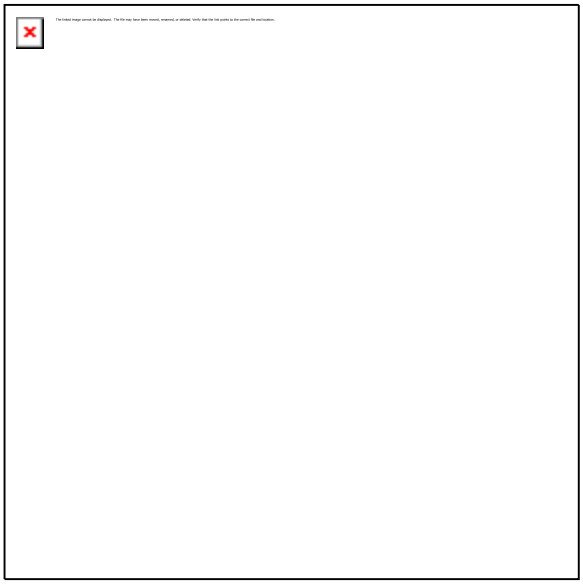
Parámetros

str	-> cadena que contiene el número

Devuelve

booleano -> Verdadero o falso

Gráfico de llamadas a esta función:



static double[][] es.ull.esit.utilities.ExpositoUtilities.multiplyMatrices (double *a*[][], double *b*[][]) `[static]`

Método para multiplicar 2 matrices dobles.

Parámetros

a	-> Matriz izquierda
b	-> Matriz derecha

Devuelve

double[][] -> Resultado del producto Matriz

static void es.ull.esit.utilities.ExpositoUtilities.printFile (String *file*) `[static]`

Método para imprimir archivos.

Parámetros

archivo	-> nombre de archivo

static String es.ull.esit.utilities.ExpositoUtilities.simplifyString (String *string*) `[static]`

Parser para simplificar las cadenas que contienen caracteres no deseados.

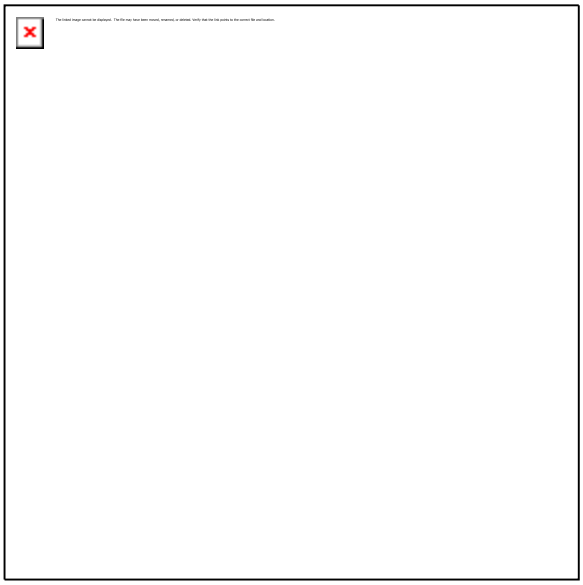
Parámetros

string	-> Cadena a simplificar

Devuelve

String -> cadena simplificada

Gráfico de llamadas a esta función:



static boolean es.ull.esit.utilities.ExpositoUtilities.thereIsPath (int *distanceMatrix*[][], int *node*) `[static]`

Comprueba si un nodo determinado es accesible.

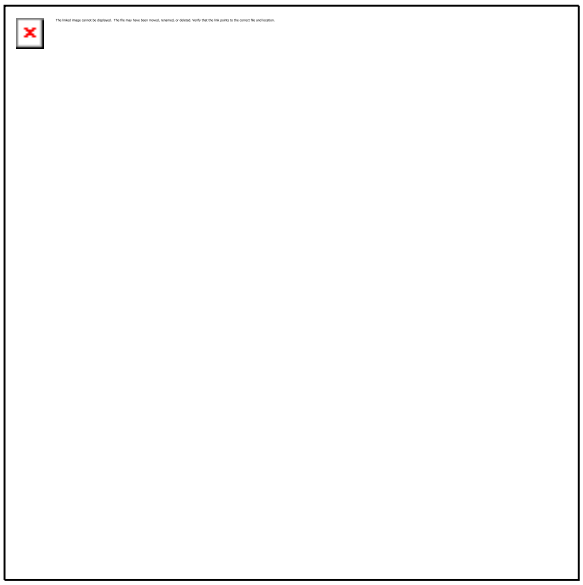
Parámetros

distanceMatrix	-> Matriz de distancias
nodo	-> nodo objetivo

Devuelve

booleano -> Verdadero o falso

Gráfico de llamadas a esta función:



Documentación de los datos miembro

final int es.ull.esit.utilities.ExpositoUtilities.ALIGNMENT_LEFT = 1 `[static]`

Constante para definir alineación izquierda

final int es.ull.esit.utilities.ExpositoUtilities.ALIGNMENT_RIGHT = 2 `[static]`

Constante para definir alineación derecha

final int es.ull.esit.utilities.ExpositoUtilities.DEFAULT_COLUMN_WIDTH = 10 `[static]`

Constante para definir ancho de columna

La documentación para esta clase fue generada a partir del siguiente fichero:

- ExpositoTOP/src/es/ull/esit/utilities/ExpositoUtilities.java

Referencia de la Clase top.mainTOPTW

Métodos públicos estáticos

- static void **main** (String[] args)

La documentación para esta clase fue generada a partir del siguiente fichero:

- ExpositoTOP/src/top/mainTOPTW.java

Referencia de la plantilla de la Clase es.ull.esit.utils.Pair< F, S >

Clase para representar un par genérico de objetos.

Métodos públicos

- **Pair** (F **first**, S **second**)
  Constructor.

- boolean **equals** (Object o)
  Comprueba si un par es igual a otro.

- int **hashCode** ()
  HashCode de la pareja.

Métodos públicos estáticos

- static< A, B > **Pair**< A, B > **create** (A a, B b)
  Crea un nuevo par.

Atributos públicos

- final F **first**
- final S **second**

Descripción detallada

Clase para representar un par genérico de objetos.

Documentación del constructor y destructor

es.ull.esit.utils.Pair< F, S >.Pair (F *first*, S *second*)

Constructor.

Parámetros

first	-> valor del primer par
second	-> valor del segundo par

Documentación de las funciones miembro

static< A, B > Pair< A, B > es.ull.esit.utils.Pair< F, S >.create (A *a*, B *b*) `[static]`

Crea un nuevo par.

Parámetros

a	-> valor del primer par
b	-> valor del segundo par

Devuelve

Pair -> pair creado

boolean es.ull.esit.utils.Pair< F, S >.equals (Object *o*)

Comprueba si un par es igual a otro.

Parámetros

o	-> comparación

Devuelve

boolean -> Verdadero o falso

Gráfico de llamadas para esta función:

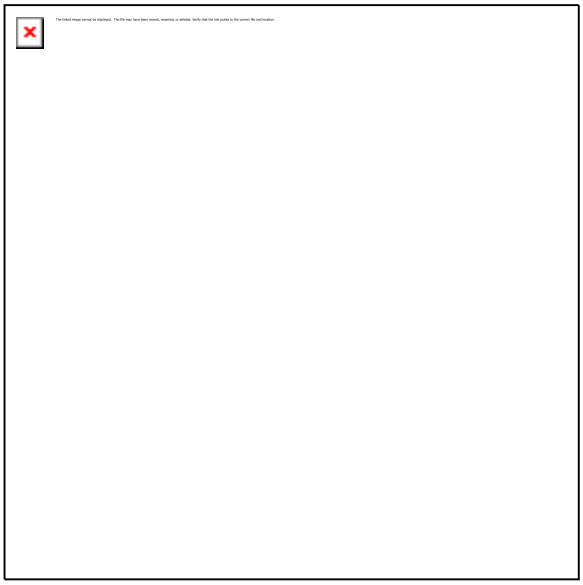
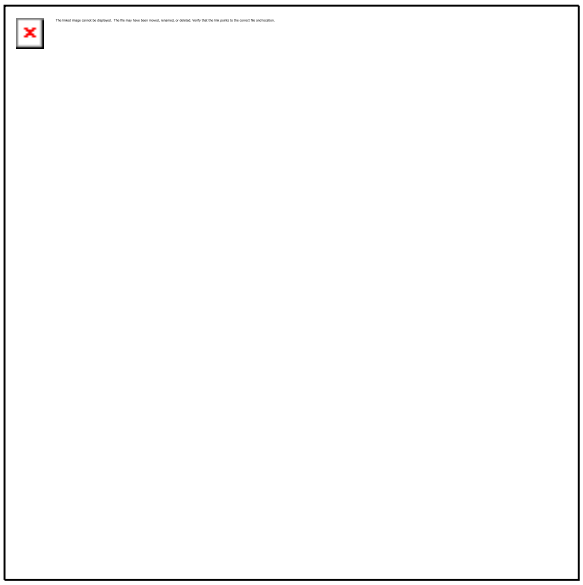


Gráfico de llamadas a esta función:



int es.ull.esit.utils.Pair< F, S >.hashCode ()

HashCode de la pareja.

Devuelve

int -> código hash

Documentación de los datos miembro

final F es.ull.esit.utils.Pair< F, S >.first

Valor del primer par.

final S es.ull.esit.utils.Pair< F, S >.second

Valor del segundo par.

La documentación para esta clase fue generada a partir del siguiente fichero:

- ExpositoTOP/src/es/ull/esit/utils/Pair.java

Referencia de la plantilla de la Clase es.ull.esit.utilities.PowerSet< E >

Clase para calcular cada subconjunto de un conjunto dado.

Diagrama de herencias de es.ull.esit.utilities.PowerSet< E >

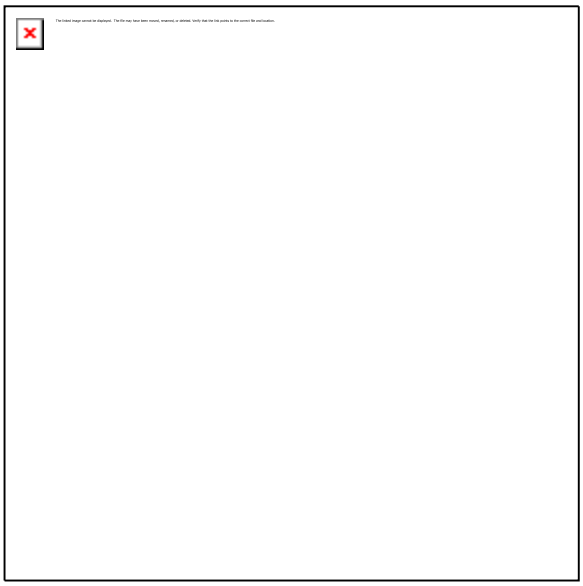
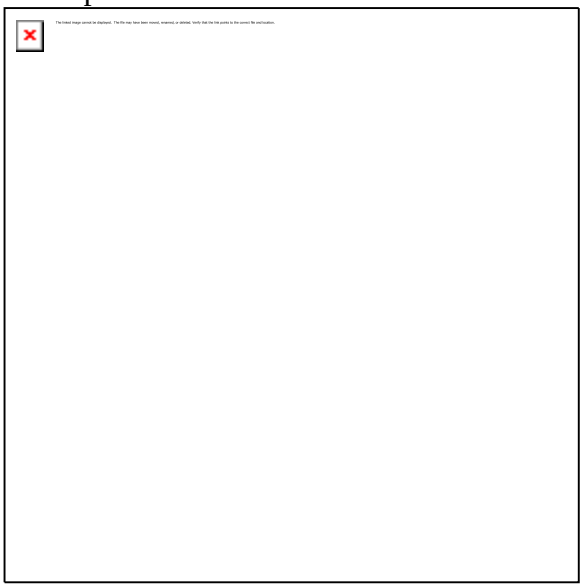


Diagrama de colaboración para es.ull.esit.utilities.PowerSet< E >:



Métodos públicos

- **PowerSet** (Set< E > set)
  constructor de clase

- boolean **hasNext** ()
  Comprueba si un subconjunto tiene un subconjunto siguiente.

- Set< E > **next** ()

Calcula el siguiente subconjunto.

- void **remove** ()
  Mensaje de error.

- Iterator< Set< E > > **iterator** ()
  Iterador.

Descripción detallada

Clase para calcular cada subconjunto de un conjunto dado.

Documentación del constructor y destructor

es.ull.esit.utilities.PowerSet< E >.PowerSet (Set< E > *set*)

constructor de clase

Parámetros

set	-> set para calcular sus subconjuntos

Documentación de las funciones miembro

boolean es.ull.esit.utilities.PowerSet< E >.hasNext ()

Comprueba si un subconjunto tiene un subconjunto siguiente.

Devuelve

bool -> Verdadero o falso

Iterator< Set< E > > es.ull.esit.utilities.PowerSet< E >.iterator ()

Iterador.

Devuelve

this

Set< E > es.ull.esit.utilities.PowerSet< E >.next ()

Calcula el siguiente subconjunto.

Devuelve

set -> set resultado

void es.ull.esit.utilities.PowerSet< E >.remove ()

Mensaje de error.

Devuelve

Not Supported

La documentación para esta clase fue generada a partir del siguiente fichero:

- ExpositoTOP/src/es/ull/esit/utilities/PowerSet.java

Referencia de la Clase top.TOPTW

Métodos públicos

- **TOPTW** (int nodes, int routes)
- boolean **isDepot** (int a)
- double **getDistance** (int[] route)
- double **getDistance** (ArrayList< Integer > route)
- double **getDistance** (ArrayList< Integer >[] routes)
- void **calculateDistanceMatrix** ()
- double **getMaxTimePerRoute** ()
- void **setMaxTimePerRoute** (double maxTimePerRoute)
- double **getMaxRoutes** ()
- void **setMaxRoutes** (double maxRoutes)
- int **getPOIs** ()
- double **getDistance** (int i, int j)
- double **getTime** (int i, int j)
- int **getNodes** ()
- void **setNodes** (int nodes)
- double **getX** (int index)
- void **setX** (int index, double x)
- double **getY** (int index)
- void **setY** (int index, double y)
- double **getScore** (int index)
- double[] **getScore** ()
- void **setScore** (int index, double score)
- double **getReadyTime** (int index)
- void **setReadyTime** (int index, double readyTime)
- double **getDueTime** (int index)
- void **setDueTime** (int index, double dueTime)
- double **getServiceTime** (int index)
- void **setServiceTime** (int index, double serviceTime)
- int **getVehicles** ()
- String **toString** ()
- int **addNode** ()
- int **addNodeDepot** ()

La documentación para esta clase fue generada a partir del siguiente fichero:

- ExpositoTOP/src/top/TOPTW.java

Referencia de la Clase top.TOPTWEvaluator

Métodos públicos

- void **evaluate** (**TOPTWSolution** solution)

Atributos públicos estáticos

- static double **NO_EVALUATED** = -1.0

Documentación de las funciones miembro

void top.TOPTWEvaluator.evaluate (TOPTWSolution *solution*)

CumulativeCVRP problem = solution.getProblem(); double objectiveFunctionValue = 0.0; for (int i = 0; i < solution.getIndexDepot().size(); i++) { double cumulative = 0; int depot = solution.getAnIndexDepot(i); int actual = depot; actual = solution.getSuccessor(actual); cumulative += problem.getDistanceMatrix(0, actual); objectiveFunctionValue += problem.getDistanceMatrix(0, actual); System.out.println("Desde " + 0 + " a " + actual + " = " + cumulative); while (actual != depot) { int ant = actual; actual = solution.getSuccessor(actual); if (actual != depot) { cumulative += problem.getDistanceMatrix(ant, actual); objectiveFunctionValue += cumulative; System.out.println("Desde " + ant + " a " + actual + " = " + cumulative); } else { cumulative += problem.getDistanceMatrix(ant, 0); objectiveFunctionValue += cumulative; System.out.println("Desde " + ant + " a " + 0 + " = " + cumulative); } } System.out.println(""); } solution.setObjectiveFunctionValue(objectiveFunctionValue);

La documentación para esta clase fue generada a partir del siguiente fichero:

- ExpositoTOP/src/top/TOPTWEvaluator.java

Referencia de la Clase top.TOPTWGRASP

Métodos públicos

- **TOPTWGRASP** (**TOPTWSolution** sol)
- void **GRASP** (int maxIterations, int maxSizeRCL)
- int **aleatorySelectionRCL** (int maxTRCL)
- int **fuzzySelectionBestFDRCL** (ArrayList< double[] > rcl)
- int **fuzzySelectionAlphaCutRCL** (ArrayList< double[] > rcl, double alpha)
- void **computeGreedySolution** (int maxSizeRCL)
- void **updateSolution** (double[] candidateSelected, ArrayList< ArrayList< Double > > departureTimes)
- ArrayList< double[] > **comprehensiveEvaluation** (ArrayList< Integer > customers, ArrayList< ArrayList< Double > > departureTimes)
- **TOPTWSolution getSolution** ()
- void **setSolution** (**TOPTWSolution** solution)
- int **getSolutionTime** ()
- void **setSolutionTime** (int solutionTime)
- double **getMaxScore** ()

Atributos públicos estáticos

- static double **NO_EVALUATED** = -1.0

Documentación de las funciones miembro

void top.TOPTWGRASP.GRASP (int *maxIterations*, int *maxSizeRCL*)

procedure GRASP(Max Iterations,Seed) 1 Read Input(); 2 for k = 1, . . . , Max Iterations do 3 Solution â†• Greedy Randomized Construction(Seed); 4 Solution â†• Local Search(Solution); 5 Update Solution(Solution,Best Solution); 6 end; 7 return Best Solution; end GRASP

La documentación para esta clase fue generada a partir del siguiente fichero:

- ExpositoTOP/src/top/TOPTWGRASP.java

Referencia de la Clase top.TOPTWReader

Esta Clase lee un problema **TOPTW**.

Métodos públicos estáticos

- static **TOPTW readProblem** (String filePath)
  *Lee un problema **TOPTW** del archivo.*

Descripción detallada

Esta Clase lee un problema **TOPTW**.

Documentación de las funciones miembro

static TOPTW top.TOPTWReader.readProblem (String *filePath*) `[static]`

Lee un problema **TOPTW** del archivo.

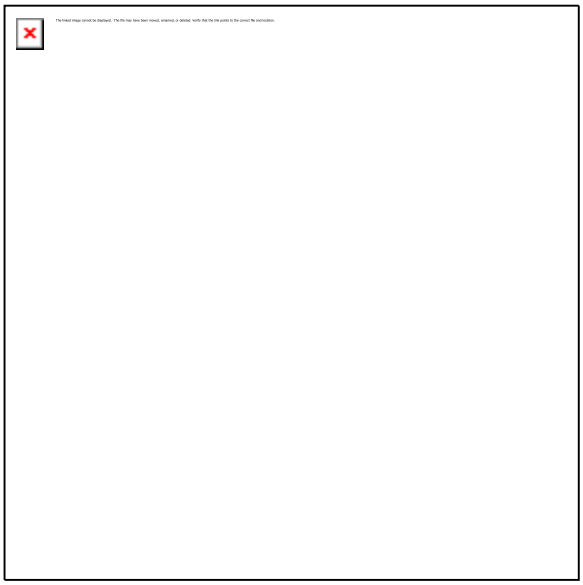
Parámetros

filePath	-> ruta del archivo

Devuelve

TOPTW -> Objeto problemático **TOPTW**

Gráfico de llamadas para esta función:



La documentación para esta clase fue generada a partir del siguiente fichero:

- ExpositoTOP/src/top/TOPTWReader.java

Referencia de la Clase top.TOPTWRoute

Class para representar la ruta.

Métodos públicos

- int **getPredeccesor** ()
  Getter.

- int **getSuccesor** ()
  Getter.

- int **getId** ()
  Getter.

- void **setPredeccesor** (int pre)
  Setter.

- void **setSuccesor** (int suc)
  Setter.

- void **setId** (int id)
  Setter.

Descripción detallada

Class para representar la ruta.

Documentación de las funciones miembro

int top.TOPTWRoute.getId ()

Getter.

Devuelve
int -> ruta id

int top.TOPTWRoute.getPredeccesor ()

Getter.

Devuelve
int -> predecessor

int top.TOPTWRoute.getSuccesor ()

Getter.

Devuelve

int -> sucessor

void top.TOPTWRoute.setId (int *id*)

Setter.

Parámetros

id	-> id de ruta

void top.TOPTWRoute.setPredeccesor (int *pre*)

Setter.

Parámetros

pre	-> predecesor

void top.TOPTWRoute.setSuccesor (int *suc*)

Setter.

Parámetros

suc	-> sucesor

La documentación para esta clase fue generada a partir del siguiente fichero:

- ExpositoTOP/src/top/TOPTWRoute.java

Referencia de la Clase top.TOPTWSolution

Clase para representar la solución del problema **TOPTW**.

Métodos públicos

- **TOPTWSolution** (**TOPTW** problem)
- void **initSolution** ()
- boolean **isDepot** (int c)
- boolean **equals** (**TOPTWSolution** otherSolution)
- int **getAvailableVehicles** ()
- int **getCreatedRoutes** ()
- double **getDistance** (int x, int y)
- void **setAvailableVehicles** (int availableVehicles)
- int **getPredecessor** (int customer)
- int[] **getPredecessors** ()
- **TOPTW getProblem** ()
- double **getObjectiveFunctionValue** ()
- int **getPositionInRoute** (int customer)
- int **getSuccessor** (int customer)
- int[] **getSuccessors** ()
- int **getIndexRoute** (int index)
- double **getWaitingTime** (int customer)
- void **setObjectiveFunctionValue** (double objectiveFunctionValue)
- void **setPositionInRoute** (int customer, int position)
- void **setPredecessor** (int customer, int predecessor)
- void **setSuccessor** (int customer, int succesor)
- void **setWaitingTime** (int customer, int waitingTime)
- String **getInfoSolution** ()
- double **evaluateFitness** ()
- int **addRoute** ()
- double **printSolution** ()

Atributos públicos estáticos

- static final int **NO_INITIALIZED** = -1

Descripción detallada

Clase para representar la solución del problema **TOPTW**.

La documentación para esta clase fue generada a partir del siguiente fichero:

- ExpositoTOP/src/top/TOPTWSolution.java

Índice

INDEX