Practica 3

Generado por Doxygen 1.12.0

Capítulo 1

Índice de espacios de nombres

1.1. Lista de paquetes

Estos son los paquetes con breves descripciones (si están disponibles):

Capítulo 2

Índice jerárquico

2.1. Jerarquía de clases

Este listado de herencia está ordenado de forma general pero no está en orden alfabético estricto:

Capítulo 3

Índice de clases

3.1. Lista de clases

Lista de clases, estructuras, uniones e interfaces con breves descripciones:

Capítulo 4

Índice de archivos

4.1. Lista de archivos

Lista de todos los archivos con breves descripciones:

Capítulo 5

Documentación de espacios de nombres

5.1. Paquete es.ull.esit.utilities

Clases

- class **BellmanFord**
- class **ExpositoUtilities**
- class **PowerSet**

5.2. Paquete es.ull.esit.utils

Clases

- class **Pair**

5.3. Paquete top

Clases

- class **mainTOPTW**

  Clase Main que comienza y dirige la ejecución del programa.
- class **TOPTW**

  Esta clase representa el modelo de un problema de ruteo con restricciones de tiempo.
- class **TOPTWEvaluator**
- class **TOPTWGRASP**

  *Clase que implementa el algoritmo GRASP para resolver el problema **TOPTW** (*pág. *20).*
- class **TOPTWReader**

  *Clase para leer archivos de entrada y cargar los datos de problemas de **TOPTW** (*pág. *20) (Team Orienteering Problem with Time Windows).*
- class **TOPTWRoute**

  *Clase que representa una ruta en el problema de **TOPTW** (*pág. *20). Cada ruta tiene un nodo predecesor, un nodo sucesor y un identificador.*
- class **TOPTWSolution**

  *Clase que representa una solución para el problema **TOPTW** (*pág. *20) (Team Orienteering Problem with Time Windows).*

Capítulo 6

Documentación de clases

6.1. Referencia de la clase es.ull.esit.utilities.BellmanFord

Métodos públicos

- **BellmanFord** (int[ ][ ] distanceMatrix, int nodes, ArrayList< Integer > path)
- int[ ] **getDistances** ()
- int **getValue** ()
- void **solve** ()

6.1.1. Documentación de constructores y destructores

6.1.1.1. BellmanFord()

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

Parámetros

distanceMatrix	
nodes	
path	

6.1.2. Documentación de funciones miembro

6.1.2.1. getDistances()

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

Devuelve

6.1.2.2. getValue()

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

Devuelve

6.1.2.3. solve()

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

La documentación de esta clase está generada del siguiente archivo:

- src/main/java/es/ull/esit/utilities/ **BellmanFord.java**

6.2. Referencia de la clase es.ull.esit.utilities.ExpositoUtilities

Métodos públicos estáticos

- static int **generateRandomNumber** (int min, int max)
- static double **generateRandomDouble** (double min, double max)
- static< T > void **shuffleArray** (T[ ] array)
- static void **printFile** (String file)
- static String **simplifyString** (String string)
- static double[ ][ ] **multiplyMatrices** (double[ ][ ] a, double[ ][ ] b)
- static void **writeTextToFile** (String file, String text) throws IOException
- static String **getFormat** (String[ ] strings, int[ ] columnWidths)
- static String **getFormat** (double value)
- static boolean **isInteger** (String str)
- static boolean **isDouble** (String str)
- static boolean **isAcyclic** (int[ ][ ] distanceMatrix)
- static boolean **thereIsPath** (int[ ][ ] distanceMatrix, int node)

Atributos públicos estáticos

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

6.2.1. Documentación de funciones miembro

6.2.1.1. generateRandomDouble()

```
static double es.ull.esit.utilities.ExpositoUtilities.generateRandomDouble (
            double min,
            double max)  [static]
```

Genera un número aleatorio de punto flotante en un rango específico.

Parámetros

min	El valor mínimo (incluido).
max	El valor máximo (excluido).

Devuelve

Un número aleatorio de punto flotante dentro del rango [min, max).

6.2.1.2. generateRandomNumber()

```
static int es.ull.esit.utilities.ExpositoUtilities.generateRandomNumber (
            int min,
            int max)  [static]
```

Genera un número aleatorio dentro de un rango específico.

Parámetros

min	El valor mínimo (incluido).
max	El valor máximo (excluido).

Devuelve

Un número aleatorio dentro del rango [min, max).

6.2.1.3. getFormat() [1/2]

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

Da formato a un número double con tres decimales.

Parámetros

value	valor a formatear

Devuelve

el string formateado

6.2.1.4. getFormat() [2/2]

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

Da formato a un string, detectando si es número entero o decimal.

Parámetros

strings	el string a formatear

Devuelve

el string formateado

6.2.1.5. isAcyclic()

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

Verifica si una matriz de distancias es acíclica.

Parámetros

distanceMatrix	matriz de distancias

Devuelve

true si no contiene ciclos, false en caso contrario

6.2.1.6. isDouble()

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

Verifica si un string representa un número decimal.

Parámetros

str	el string a verificar

Devuelve

true si es un decimal, false en caso contrario

6.2.1.7. isInteger()

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

Verifica si un string representa un número entero.

Parámetros

str	el string a verificar

Devuelve

true si es un entero, false en caso contrario

6.2.1.8. multiplyMatrices()

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

Multiplica dos matrices.

Parámetros

a	primera matriz
b	segunda matriz

Devuelve

matriz resultado o null si las dimensiones son incompatibles

6.2.1.9. printFile()

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

Imprime el contenido de un archivo línea por línea.

Parámetros

file	ruta del archivo a imprimir

6.2.1.10. shuffleArray()

```
static< T > void es.ull.esit.utilities.ExpositoUtilities.shuffleArray (
            T[] array)  [static]
```

Mezcla un arreglo de elementos de manera aleatoria.

Parámetros

array	El arreglo a mezclar.

6.2.1.11. simplifyString()

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

Simplifica un string eliminando espacios extra y tabs.

Parámetros

string	el string a simplificar

Devuelve

el string simplificado

6.2.1.12. thereIsPath()

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

Determina si existe un camino desde un nodo específico en una matriz de distancias.

Parámetros

distanceMatrix	matriz de distancias
node	nodo a verificar

Devuelve

true si existe un camino que forma un ciclo, false en caso contrario

6.2.1.13. writeTextToFile()

```
static void es.ull.esit.utilities.ExpositoUtilities.writeTextToFile (
            String file,
            String text) throws IOException  [static]
```

Escribe un texto en un archivo, usando try-with-resources para gestionar el cierre de BufferedWriter.

Parámetros

file	archivo donde escribir
text	texto a escribir

Excepciones

IOException	si ocurre un error de escritura

6.2.2. Documentación de datos miembro

6.2.2.1. ALIGNMENT_LEFT

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

6.2.2.2. ALIGNMENT_RIGHT

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

6.2.2.3. DEFAULT_COLUMN_WIDTH

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

La documentación de esta clase está generada del siguiente archivo:

- src/main/java/es/ull/esit/utilities/ **ExpositoUtilities.java**

6.3. Referencia de la clase top.mainTOPTW

Clase Main que comienza y dirige la ejecución del programa.

Métodos públicos estáticos

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

6.3.1. Descripción detallada

Clase Main que comienza y dirige la ejecución del programa.

6.3.2. Documentación de funciones miembro

6.3.2.1. main()

```
static void top.mainTOPTW.main (
            String[] args)  [static]
```

La documentación de esta clase está generada del siguiente archivo:

- src/main/java/top/ **mainTOPTW.java**

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

Métodos públicos

- **Pair** (F **first**, S **second**)
- boolean **equals** (Object o)
- int **hashCode** ()

Métodos públicos estáticos

- static< A, B > **Pair**< A, B > **create** (A a, B b)

Atributos públicos

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

6.4.1. Documentación de constructores y destructores

6.4.1.1. Pair()

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

6.4.2. Documentación de funciones miembro

6.4.2.1. create()

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

6.4.2.2. equals()

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

6.4.2.3. hashCode()

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

6.4.3. Documentación de datos miembro

6.4.3.1. first

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

6.4.3.2. second

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

La documentación de esta clase está generada del siguiente archivo:

- src/main/java/es/ull/esit/utils/ **Pair.java**

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

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

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

Métodos públicos

- **PowerSet** (Set< E > set)
- boolean **hasNext** ()
- Set< E > **next** ()
- void **remove** ()
- Iterator< Set< E > > **iterator** ()

6.5.1. Documentación de constructores y destructores

6.5.1.1. PowerSet()

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

6.5.2. Documentación de funciones miembro

6.5.2.1. hasNext()

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

6.5.2.2. iterator()

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

6.5.2.3. next()

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

6.5.2.4. remove()

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

La documentación de esta clase está generada del siguiente archivo:

- src/main/java/es/ull/esit/utilities/ **PowerSet.java**

6.6. Referencia de la clase top.TOPTW

Esta clase representa el modelo de un problema de ruteo con restricciones de tiempo.

Métodos públicos

- **TOPTW** (int nodes, int routes)

  *Constructor de la clase **TOPTW** (*pág. *20)**.*
- boolean **isDepot** (int a)

  Verifica si un punto es un depósito.
- double **getDistance** (int[ ] route)

  Calcula la distancia total de una ruta.
- double **getDistance** (ArrayList< Integer > route)

  Calcula la distancia total de una ruta representada como lista.
- double **getDistance** (ArrayList< Integer >[ ] routes)

  Calcula la distancia total de varias rutas.
- void **calculateDistanceMatrix** ()

  Calcula la matriz de distancias entre puntos de interés.
- double **getMaxTimePerRoute** ()

  Obtiene el tiempo máximo permitido por ruta.
- void **setMaxTimePerRoute** (double maxTimePerRoute)

  Establece el tiempo máximo permitido por ruta.
- double **getMaxRoutes** ()

  Obtiene el número máximo de rutas.
- void **setMaxRoutes** (double maxRoutes)

  Establece el número máximo de rutas.
- int **getPOIs** ()

  Obtiene el número de puntos de interés.
- double **getDistance** (int i, int j)

  Obtiene la distancia entre dos puntos específicos.
- double **getTime** (int i, int j)

  Obtiene el tiempo necesario para viajar entre dos puntos.
- int **getNodes** ()

  Obtiene el número total de puntos de interés (nodos).
- void **setNodes** (int nodes)

  Establece el número de nodos en el problema.
- double **getX** (int index)

  Obtiene la coordenada X de un punto de interés.
- void **setX** (int index, double x)

  Establece la coordenada X de un punto de interés.
- double **getY** (int index)

  Obtiene la coordenada Y de un punto de interés.
- void **setY** (int index, double y)

  Establece la coordenada Y de un punto de interés.
- double **getScore** (int index)

  Obtiene la puntuación asociada a un punto de interés.
- double[ ] **getScore** ()

  Obtiene el arreglo de puntuaciones de todos los puntos de interés.
- void **setScore** (int index, double score)

  Establece la puntuación de un punto de interés.

- double **getReadyTime** (int index)

  Obtiene el tiempo de inicio de servicio de un punto de interés.
- void **setReadyTime** (int index, double readyTime)

  Establece el tiempo de inicio de servicio de un punto de interés.
- double **getDueTime** (int index)

  Obtiene el tiempo límite de servicio de un punto de interés.
- void **setDueTime** (int index, double dueTime)

  Establece el tiempo límite de servicio de un punto de interés.
- double **getServiceTime** (int index)

  Obtiene el tiempo de servicio de un punto de interés.
- void **setServiceTime** (int index, double serviceTime)

  Establece el tiempo de servicio de un punto de interés.
- int **getVehicles** ()

  Obtiene el número de vehículos disponibles.
- String **toString** ()

  Convierte el objeto a una representación de cadena con información detallada.
- int **addNode** ()

  Añade un nuevo nodo al conjunto de puntos de interés.
- int **addNodeDepot** ()

  Añade un nuevo depósito al conjunto de depósitos.

6.6.1. Descripción detallada

Esta clase representa el modelo de un problema de ruteo con restricciones de tiempo.

6.6.2. Documentación de constructores y destructores

6.6.2.1. TOPTW()

```
top.TOPTW.TOPTW (
            int nodes,
            int routes)
```

Constructor de la clase **TOPTW** (pág. 20).

Parámetros

nodes	Número de puntos de interés.
routes	Número de rutas (vehículos) disponibles.

6.6.3. Documentación de funciones miembro

6.6.3.1. addNode()

```
int top.TOPTW.addNode ()
```

Añade un nuevo nodo al conjunto de puntos de interés.

Devuelve

Número total de nodos tras la adición.

6.6.3.2. addNodeDepot()

```
int top.TOPTW.addNodeDepot ()
```

Añade un nuevo depósito al conjunto de depósitos.

Devuelve

Número total de depósitos tras la adición.

6.6.3.3. calculateDistanceMatrix()

```
void top.TOPTW.calculateDistanceMatrix ()
```

Calcula la matriz de distancias entre puntos de interés.

6.6.3.4. getDistance() [1/4]

```
double top.TOPTW.getDistance (
            ArrayList< Integer > route)
```

Calcula la distancia total de una ruta representada como lista.

Parámetros

route	Lista de índices que representan la ruta.

Devuelve

Distancia total de la ruta.

6.6.3.5. getDistance() [2/4]

```
double top.TOPTW.getDistance (
            ArrayList< Integer >[] routes)
```

Calcula la distancia total de varias rutas.

Parámetros

routes	Array de listas, cada una representando una ruta.

Devuelve

Distancia total de todas las rutas.

6.6.3.6. getDistance() [3/4]

```
double top.TOPTW.getDistance (
            int i,
            int j)
```

Obtiene la distancia entre dos puntos específicos.

Parámetros

i	Índice del primer punto.
j	Índice del segundo punto.

Devuelve

Distancia entre los dos puntos.

6.6.3.7. getDistance() [4/4]

```
double top.TOPTW.getDistance (
            int[] route)
```

Calcula la distancia total de una ruta.

Parámetros

route	Array de índices que representan la ruta.

Devuelve

Distancia total de la ruta.

6.6.3.8. getDueTime()

```
double top.TOPTW.getDueTime (
            int index)
```

Obtiene el tiempo límite de servicio de un punto de interés.

Parámetros

index	Índice del punto de interés.

Devuelve

Tiempo límite de servicio.

6.6.3.9. getMaxRoutes()

```
double top.TOPTW.getMaxRoutes ()
```

Obtiene el número máximo de rutas.

Devuelve

Número máximo de rutas.

6.6.3.10. getMaxTimePerRoute()

```
double top.TOPTW.getMaxTimePerRoute ()
```

Obtiene el tiempo máximo permitido por ruta.

Devuelve

Tiempo máximo por ruta.

6.6.3.11. getNodes()

```
int top.TOPTW.getNodes ()
```

Obtiene el número total de puntos de interés (nodos).

Devuelve

Número de nodos.

6.6.3.12. getPOIs()

```
int top.TOPTW.getPOIs ()
```

Obtiene el número de puntos de interés.

Devuelve

Número de puntos de interés.

6.6.3.13. getReadyTime()

```
double top.TOPTW.getReadyTime (
            int index)
```

Obtiene el tiempo de inicio de servicio de un punto de interés.

Parámetros

index	Índice del punto de interés.

Devuelve

Tiempo de inicio de servicio.

6.6.3.14. getScore() [1/2]

```
double[] top.TOPTW.getScore ()
```

Obtiene el arreglo de puntuaciones de todos los puntos de interés.

Devuelve

Arreglo de puntuaciones.

6.6.3.15. getScore() [2/2]

```
double top.TOPTW.getScore (
            int index)
```

Obtiene la puntuación asociada a un punto de interés.

Parámetros

index	Índice del punto de interés.

Devuelve

Puntuación del punto de interés.

6.6.3.16. getServiceTime()

```
double top.TOPTW.getServiceTime (
            int index)
```

Obtiene el tiempo de servicio de un punto de interés.

Parámetros

index	Índice del punto de interés.

Devuelve

Tiempo de servicio.

6.6.3.17. getTime()

```
double top.TOPTW.getTime (
            int i,
            int j)
```

Obtiene el tiempo necesario para viajar entre dos puntos.

Parámetros

i	Índice del primer punto.
j	Índice del segundo punto.

Devuelve

Tiempo entre los dos puntos.

6.6.3.18. getVehicles()

```
int top.TOPTW.getVehicles ()
```

Obtiene el número de vehículos disponibles.

Devuelve

Número de vehículos.

6.6.3.19. getX()

```
double top.TOPTW.getX (
            int index)
```

Obtiene la coordenada X de un punto de interés.

Parámetros

index	Índice del punto de interés.

Devuelve

Coordenada X del punto de interés.

6.6.3.20. getY()

```
double top.TOPTW.getY (
            int index)
```

Obtiene la coordenada Y de un punto de interés.

Parámetros

index	Índice del punto de interés.

Devuelve

Coordenada Y del punto de interés.

6.6.3.21. isDepot()

```
boolean top.TOPTW.isDepot (
            int a)
```

Verifica si un punto es un depósito.

Parámetros

a	Índice del punto.

Devuelve

Verdadero si el punto es un depósito, falso en caso contrario.

6.6.3.22. setDueTime()

```
void top.TOPTW.setDueTime (
            int index,
            double dueTime)
```

Establece el tiempo límite de servicio de un punto de interés.

Parámetros

index	Índice del punto de interés.
dueTime	Tiempo límite de servicio a establecer.

6.6.3.23. setMaxRoutes()

```
void top.TOPTW.setMaxRoutes (
            double maxRoutes)
```

Establece el número máximo de rutas.

Parámetros

maxRoutes	Número máximo de rutas.

6.6.3.24. setMaxTimePerRoute()

```
void top.TOPTW.setMaxTimePerRoute (
            double maxTimePerRoute)
```

Establece el tiempo máximo permitido por ruta.

Parámetros

maxTimePerRoute	Tiempo máximo por ruta.

6.6.3.25. setNodes()

```
void top.TOPTW.setNodes (
            int nodes)
```

Establece el número de nodos en el problema.

Parámetros

nodes	Número de nodos.

6.6.3.26. setReadyTime()

```
void top.TOPTW.setReadyTime (
            int index,
            double readyTime)
```

Establece el tiempo de inicio de servicio de un punto de interés.

Parámetros

index	Índice del punto de interés.
readyTime	Tiempo de inicio de servicio a establecer.

6.6.3.27. setScore()

```
void top.TOPTW.setScore (
            int index,
            double score)
```

Establece la puntuación de un punto de interés.

Parámetros

index	Índice del punto de interés.
score	Puntuación a establecer.

6.6.3.28. setServiceTime()

```
void top.TOPTW.setServiceTime (
            int index,
            double serviceTime)
```

Establece el tiempo de servicio de un punto de interés.

Parámetros

index	Índice del punto de interés.
serviceTime	Tiempo de servicio a establecer.

6.6.3.29. setX()

```
void top.TOPTW.setX (
            int index,
            double x)
```

Establece la coordenada X de un punto de interés.

Parámetros

index	Índice del punto de interés.
x	Coordenada X a establecer.

6.6.3.30. setY()

```
void top.TOPTW.setY (
            int index,
            double y)
```

Establece la coordenada Y de un punto de interés.

Parámetros

index	Índice del punto de interés.
y	Coordenada Y a establecer.

6.6.3.31. toString()

```
String top.TOPTW.toString ()
```

Convierte el objeto a una representación de cadena con información detallada.

Devuelve

Representación en formato de texto del objeto.

La documentación de esta clase está generada del siguiente archivo:

- src/main/java/top/ **TOPTW.java**

6.7. Referencia de la clase top.TOPTWEvaluator

Métodos públicos

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

Atributos públicos estáticos

- static double **NO_EVALUATED** = -1.0

6.7.1. Documentación de funciones miembro

6.7.1.1. evaluate()

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

6.7.2. Documentación de datos miembro

6.7.2.1. NO_EVALUATED

```
double top.TOPTWEvaluator.NO_EVALUATED = -1.0  [static]
```

La documentación de esta clase está generada del siguiente archivo:

- src/main/java/top/ **TOPTWEvaluator.java**

6.8. Referencia de la clase top.TOPTWGRASP

Clase que implementa el algoritmo GRASP para resolver el problema **TOPTW** (pág. 20).

Métodos públicos

- **TOPTWGRASP** (**TOPTWSolution** sol)
  *Constructor de la clase **TOPTWGRASP** (pág. 30).*
- void **GRASP** (int maxIterations, int maxSizeRCL)
  Método principal de GRASP que ejecuta iteraciones del algoritmo.
- int **aleatorySelectionRCL** (int maxTRCL)
  Selecciona aleatoriamente un índice de la RCL.
- int **fuzzySelectionBestFDRCL** (ArrayList< double[ ]> rcl)
  Realiza una selección difusa de la mejor opción en la RCL.
- int **fuzzySelectionAlphaCutRCL** (ArrayList< double[ ]> rcl, double alpha)
  Selecciona un candidato usando corte alfa en la RCL.
- void **computeGreedySolution** (int maxSizeRCL)
  Método principal de construcción de una solución greedy.
- void **updateSolution** (double[ ] candidateSelected, ArrayList< ArrayList< Double > > departureTimes)
  Actualiza la solución con el candidato seleccionado.
- ArrayList< double[ ]> **comprehensiveEvaluation** (ArrayList< Integer > customers, ArrayList< ArrayList< Double > > departureTimes)
- double **getMaxScore** ()
  Obtiene el puntaje máximo del problema.

Atributos públicos estáticos

- static double **NO_EVALUATED** = -1.0

6.8.1. Descripción detallada

Clase que implementa el algoritmo GRASP para resolver el problema **TOPTW** (pág. 20).

6.8.2. Documentación de constructores y destructores

6.8.2.1. TOPTWGRASP()

```
top.TOPTWGRASP.TOPTWGRASP (
              TOPTWSolution sol)
```

Constructor de la clase **TOPTWGRASP** (pág. 30).

Parámetros

sol	Solución inicial para el problema.

6.8.3. Documentación de funciones miembro

6.8.3.1. aleatorySelectionRCL()

```
int top.TOPTWGRASP.aleatorySelectionRCL (
            int maxTRCL)
```

Selecciona aleatoriamente un índice de la RCL.

Parámetros

maxTRCL	Tamaño máximo de la lista RCL.

Devuelve

Índice seleccionado aleatoriamente de la RCL.

6.8.3.2. comprehensiveEvaluation()

```
ArrayList< double[]> top.TOPTWGRASP.comprehensiveEvaluation (
            ArrayList< Integer > customers,
            ArrayList< ArrayList< Double > > departureTimes)
```

6.8.3.3. computeGreedySolution()

```
void top.TOPTWGRASP.computeGreedySolution (
            int maxSizeRCL)
```

Método principal de construcción de una solución greedy.

Parámetros

maxSizeRCL	Tamaño máximo de la Lista de Candidatos Restringida.

6.8.3.4. fuzzySelectionAlphaCutRCL()

```
int top.TOPTWGRASP.fuzzySelectionAlphaCutRCL (
            ArrayList< double[]> rcl,
            double alpha)
```

Selecciona un candidato usando corte alfa en la RCL.

Parámetros

rcl	Lista de Candidatos Restringida (RCL).
alpha	Valor de corte alfa.

Devuelve

Posición del candidato seleccionado en la RCL.

6.8.3.5. fuzzySelectionBestFDRCL()

```
int top.TOPTWGRASP.fuzzySelectionBestFDRCL (
            ArrayList< double[]> rcl)
```

Realiza una selección difusa de la mejor opción en la RCL.

Parámetros

rcl	Lista de Candidatos Restringida (RCL).

Devuelve

Posición del candidato seleccionado en la RCL.

6.8.3.6. getMaxScore()

```
double top.TOPTWGRASP.getMaxScore ()
```

Obtiene el puntaje máximo del problema.

Devuelve

Puntaje máximo.

6.8.3.7. GRASP()

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

Método principal de GRASP que ejecuta iteraciones del algoritmo.

Parámetros

maxIterations	Número máximo de iteraciones del algoritmo.
maxSizeRCL	Tamaño máximo de la Lista de Candidatos Restringida (RCL).

6.8.3.8. updateSolution()

```
void top.TOPTWGRASP.updateSolution (
            double[] candidateSelected,
            ArrayList< ArrayList< Double > > departureTimes)
```

Actualiza la solución con el candidato seleccionado.

Parámetros

candidateSelected	Candidato seleccionado.
departureTimes	Tiempos de salida de los clientes.

6.8.4. Documentación de datos miembro

6.8.4.1. NO_EVALUATED

```
double top.TOPTWGRASP.NO_EVALUATED = -1.0  [static]
```

Valor utilizado para indicar que una solución no ha sido evaluada.

La documentación de esta clase está generada del siguiente archivo:

- src/main/java/top/ **TOPTWGRASP.java**

6.9. Referencia de la clase top.TOPTWReader

Clase para leer archivos de entrada y cargar los datos de problemas de **TOPTW** (pág. 20) (Team Orienteering Problem with Time Windows).

Métodos públicos estáticos

- static **TOPTW readProblem** (String filePath)

6.9.1. Descripción detallada

Clase para leer archivos de entrada y cargar los datos de problemas de **TOPTW** (pág. 20) (Team Orienteering Problem with Time Windows).

6.9.2. Documentación de funciones miembro

6.9.2.1. readProblem()

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

Lee el archivo de entrada y crea una instancia del problema **TOPTW** (pág. 20).

Parámetros

filePath	La ruta del archivo que contiene los datos del problema.

Devuelve

Una instancia de **TOPTW** (pág. 20) con los datos cargados desde el archivo.

La documentación de esta clase está generada del siguiente archivo:

- src/main/java/top/ **TOPTWReader.java**

6.10. Referencia de la clase top.TOPTWRoute

Clase que representa una ruta en el problema de **TOPTW** (pág. 20). Cada ruta tiene un nodo predecesor, un nodo sucesor y un identificador.

Métodos públicos

- int **getPredeccesor** ()

  Obtiene el nodo predecesor de la ruta.
- int **getSuccesor** ()

  Obtiene el nodo sucesor de la ruta.
- int **getId** ()

  Obtiene el identificador de la ruta.
- void **setPredeccesor** (int pre)

  Establece el nodo predecesor de la ruta.
- void **setSuccesor** (int suc)

  Establece el nodo sucesor de la ruta.
- void **setId** (int id)

  Establece el identificador de la ruta.

6.10.1. Descripción detallada

Clase que representa una ruta en el problema de **TOPTW** (pág. 20). Cada ruta tiene un nodo predecesor, un nodo sucesor y un identificador.

6.10.2. Documentación de funciones miembro

6.10.2.1. getId()

```
int top.TOPTWRoute.getId ()
```

Obtiene el identificador de la ruta.

Devuelve

Identificador de la ruta.

6.10.2.2. getPredeccesor()

```
int top.TOPTWRoute.getPredeccesor ()
```

Obtiene el nodo predecesor de la ruta.

Devuelve

Nodo predecesor.

6.10.2.3. getSuccesor()

```
int top.TOPTWRoute.getSuccesor ()
```

Obtiene el nodo sucesor de la ruta.

Devuelve

Nodo sucesor.

6.10.2.4. setId()

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

Establece el identificador de la ruta.

Parámetros

id	Identificador de la ruta.

6.10.2.5. setPredeccesor()

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

Establece el nodo predecesor de la ruta.

Parámetros

pre	Nodo predecesor.

6.10.2.6. setSuccesor()

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

Establece el nodo sucesor de la ruta.

Parámetros

suc	Nodo sucesor.

La documentación de esta clase está generada del siguiente archivo:

- src/main/java/top/ **TOPTWRoute.java**

6.11. Referencia de la clase top.TOPTWSolution

Clase que representa una solución para el problema **TOPTW** (pág. 20) (Team Orienteering Problem with Time Windows).

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

6.11.1. Descripción detallada

Clase que representa una solución para el problema **TOPTW** (pág. 20) (Team Orienteering Problem with Time Windows).

6.11.2. Documentación de constructores y destructores

6.11.2.1. TOPTWSolution()

```
top.TOPTWSolution.TOPTWSolution (
              TOPTW problem)
```

Constructor que inicializa una solución para el problema dado. Inicializa los arrays de predecesores, sucesores, tiempos de espera, posición en ruta, y rutas.

Parámetros

problem	La instancia del problema **TOPTW** (pág. 20) a resolver.

6.11.3. Documentación de funciones miembro

6.11.3.1. addRoute()

```
int top.TOPTWSolution.addRoute ()
```

Añade una nueva ruta a la solución, decrementa el contador de vehículos disponibles y ajusta los arrays de predecesores y sucesores.

Devuelve

El índice del nuevo depósito en la ruta añadida.

6.11.3.2. equals()

```
boolean top.TOPTWSolution.equals (
              TOPTWSolution otherSolution)
```

Compara si esta solución es igual a otra solución dada, basada en los predecesores.

Parámetros

otherSolution	La otra solución a comparar.

Devuelve

`true` si las soluciones son equivalentes, `false` en caso contrario.

6.11.3.3. evaluateFitness()

```
double top.TOPTWSolution.evaluateFitness ()
```

Evalúa la función objetivo basada en las puntuaciones de los puntos en cada ruta.

Devuelve

El valor calculado de la función objetivo.

6.11.3.4. getAvailableVehicles()

```
int top.TOPTWSolution.getAvailableVehicles ()
```

Obtiene el número de vehículos disponibles.

Devuelve

Número de vehículos disponibles.

6.11.3.5. getCreatedRoutes()

```
int top.TOPTWSolution.getCreatedRoutes ()
```

Calcula el número de rutas creadas en la solución.

Devuelve

Número de rutas creadas.

6.11.3.6. getDistance()

```
double top.TOPTWSolution.getDistance (
            int x,
            int y)
```

Calcula la distancia entre dos puntos en la solución.

Parámetros

x	Índice del primer punto.
y	Índice del segundo punto.

Devuelve

La distancia entre `x` y `y`.

6.11.3.7. getIndexRoute()

```
int top.TOPTWSolution.getIndexRoute (
            int index)
```

6.11.3.8. getInfoSolution()

```
String top.TOPTWSolution.getInfoSolution ()
```

Obtiene una representación detallada de la solución, incluyendo información de cada ruta y cliente.

Devuelve

Cadena de texto con la información detallada de la solución.

6.11.3.9. getObjectiveFunctionValue()

```
double top.TOPTWSolution.getObjectiveFunctionValue ()
```

Obtiene el valor de la función objetivo de la solución.

Devuelve

Valor de la función objetivo.

6.11.3.10. getPositionInRoute()

```
int top.TOPTWSolution.getPositionInRoute (
            int customer)
```

6.11.3.11. getPredecessor()

```
int top.TOPTWSolution.getPredecessor (
            int customer)
```

6.11.3.12. getPredecessors()

```
int[] top.TOPTWSolution.getPredecessors ()
```

6.11.3.13. getProblem()

```
TOPTW top.TOPTWSolution.getProblem ()
```

6.11.3.14. getSuccessor()

```
int top.TOPTWSolution.getSuccessor (
            int customer)
```

6.11.3.15. getSuccessors()

```
int[] top.TOPTWSolution.getSuccessors ()
```

6.11.3.16. getWaitingTime()

```
double top.TOPTWSolution.getWaitingTime (
            int customer)
```

6.11.3.17. initSolution()

```
void top.TOPTWSolution.initSolution ()
```

Inicializa la solución configurando el depósito y estableciendo los valores predeterminados. Este método reinicia los arrays de predecesores, sucesores y rutas.

6.11.3.18. isDepot()

```
boolean top.TOPTWSolution.isDepot (
            int c)
```

Verifica si el nodo dado es un depósito.

Parámetros

c	El índice del nodo.

Devuelve

`true` si el nodo es un depósito, `false` en caso contrario.

6.11.3.19. printSolution()

```
double top.TOPTWSolution.printSolution ()
```

Imprime una representación en consola de la solución actual. Incluye los nodos visitados en cada ruta y el valor de la función objetivo.

Devuelve

Valor de la función objetivo después de imprimir la solución.

6.11.3.20. setAvailableVehicles()

```
void top.TOPTWSolution.setAvailableVehicles (
            int availableVehicles)
```

Establece el número de vehículos disponibles.

Parámetros

availableVehicles	Número de vehículos disponibles a establecer.

6.11.3.21. setObjectiveFunctionValue()

```
void top.TOPTWSolution.setObjectiveFunctionValue (
            double objectiveFunctionValue)
```

6.11.3.22. setPositionInRoute()

```
void top.TOPTWSolution.setPositionInRoute (
            int customer,
            int position)
```

6.11.3.23. setPredecessor()

```
void top.TOPTWSolution.setPredecessor (
            int customer,
            int predecessor)
```

6.11.3.24. setSuccessor()

```
void top.TOPTWSolution.setSuccessor (
             int customer,
             int succesor)
```

6.11.3.25. setWaitingTime()

```
void top.TOPTWSolution.setWaitingTime (
             int customer,
             int waitingTime)
```

6.11.4. Documentación de datos miembro

6.11.4.1. NO_INITIALIZED

```
final int top.TOPTWSolution.NO_INITIALIZED = -1  [static]
```

La documentación de esta clase está generada del siguiente archivo:

- src/main/java/top/ **TOPTWSolution.java**

Capítulo 7

Documentación de archivos

7.1. Referencia del archivo src/main/java/es/ull/esit/utilities/BellmanFord.java

```
import java.util.ArrayList;
```
Gráfico de dependencias incluidas en BellmanFord.java:

7.2. Referencia del archivo src/main/java/es/ull/esit/utilities/ExpositoUtilities.java

```
import java.text.DecimalFormat;
```
Gráfico de dependencias incluidas en ExpositoUtilities.java:

Clases

- class **es.ull.esit.utilities.ExpositoUtilities**

Paquetes

- package **es.ull.esit.utilities**

7.3. Referencia del archivo src/main/java/es/ull/esit/utilities/PowerSet.java

```
import java.util.BitSet;
```
Gráfico de dependencias incluidas en PowerSet.java:

Clases

- class **es.ull.esit.utilities.PowerSet**< **E** >

Paquetes

- package **es.ull.esit.utilities**

7.4. Referencia del archivo src/main/java/es/ull/esit/utils/Pair.java

```
import java.util.Objects;
```
Gráfico de dependencias incluidas en Pair.java:

Clases

- class **es.ull.esit.utils.Pair**< **F, S** >

Paquetes

- package **es.ull.esit.utils**

7.5. Referencia del archivo src/main/java/top/mainTOPTW.java

Clases

- class **top.mainTOPTW**

  Clase Main que comienza y dirige la ejecución del programa.

Paquetes

- package **top**

7.6. Referencia del archivo src/main/java/top/TOPTW.java

Clase para gestionar problemas de ruteo con tiempo y puntuación.

```
import java.util.ArrayList;
```
Gráfico de dependencias incluidas en TOPTW.java:

Clases

- class **top.TOPTW**

  Esta clase representa el modelo de un problema de ruteo con restricciones de tiempo.

Paquetes

- package **top**

7.6.1. Descripción detallada

Clase para gestionar problemas de ruteo con tiempo y puntuación.

7.7. Referencia del archivo src/main/java/top/TOPTWEvaluator.java

Clases

- class **top.TOPTWEvaluator**

Paquetes

- package **top**

7.8. Referencia del archivo src/main/java/top/TOPTWGRASP.java

Implementación del algoritmo GRASP para resolver el problema TOPTW.

```
import java.util.ArrayList;
```
Gráfico de dependencias incluidas en TOPTWGRASP.java:

Clases

- class **top.TOPTWGRASP**

  *Clase que implementa el algoritmo GRASP para resolver el problema **TOPTW** (*pág. *20)**.*

Paquetes

- package **top**

7.8.1. Descripción detallada

Implementación del algoritmo GRASP para resolver el problema TOPTW.

7.9. Referencia del archivo src/main/java/top/TOPTWReader.java

```
import java.io.BufferedReader;
```
Gráfico de dependencias incluidas en TOPTWReader.java:

Clases

- class **top.TOPTWReader**

  *Clase para leer archivos de entrada y cargar los datos de problemas de **TOPTW** (*pág. *20)** *(Team Orienteering Problem with Time Windows).*

Paquetes

- package **top**

7.10. Referencia del archivo src/main/java/top/TOPTWRoute.java

Clases

- class **top.TOPTWRoute**

  *Clase que representa una ruta en el problema de **TOPTW** (*pág. *20). Cada ruta tiene un nodo predecesor, un nodo sucesor y un identificador.*

Paquetes

- package **top**

7.11. Referencia del archivo src/main/java/top/TOPTWSolution.java

```
import java.util.Arrays;
```

Gráfico de dependencias incluidas en TOPTWSolution.java:

Clases

- class **top.TOPTWSolution**

  *Clase que representa una solución para el problema **TOPTW** (*pág. *20) (Team Orienteering Problem with Time Windows).*

Paquetes

- package **top**

Índice alfabético