Descripción de las clases

Algoritmo

Algoritmo es un Enumeration que permite delimitar los tipos de algoritmos que manejan las clases de la capa de dominio. Los tipos de algoritmos son JPEG, LZW, LZ78 y LZSS, además del algoritmo predeterminado, que será a su vez uno de los tres últimos.

CtrlProcesos

La clase Singleton CtrlProcesos es el Controlador de Dominio del programa, y la encargada de crear procesos de compresión y/o descompresión, además de interactuar con las capas de datos y presentación.

Atributos:

- Instance: Privado y estático, es la instancia de CtrlProcesos para garantizar que es una clase Singleton.
- algoritmoDeTextoPredeterminado: Privado y estático, lo comparten todos los atributos de la clase. Puede ser LZW, LZSS o LZ78. Por defecto es LZSS, pero se puede modificar llamando a setAlgoritmoDeTextoPredeterminado(Algoritmo)

Métodos:

- getInstance:Pública y estática, getter de la instancia Singleton del controlador CtrlProcesos y devuelve la instancia Singleton de CtrlProcesos.

- comprimirArchivo: Pública. Como prerrequisito el path que recibe de la capa de presentación debe tener una extensión correcta. Crea y ejecuta un proceso de compresión para un fichero, el cual se comprimirá con el algoritmo indicado.
    - Parámetros:
        - Path: string que indica dónde se encuentra el archivo a comprimir.
        - Algoritmo: algoritmo que se desea utilizar para realizar la compresión.

- descomprimirArchivo: Pública. Como prerrequisito el path que recibe de la capa de presentación debe tener una extensión correcta. Crea y ejecuta un proceso de descompresión para un fichero pasado por parámetro, el cual se descomprimirá con su algoritmo, seleccionado automáticamente en función de su extensión.
    - Parámetros:
        - Path: string que indica dónde se encuentra el archivo a descomprimir.

- comprimirDescomprimirArchivo: Pública. Como prerrequisito el path que recibe de la capa de presentación debe tener una extensión correcta. Crea y ejecuta un proceso de compresión y descompresión para un fichero, el cual se comprimirá con un algoritmo, ambos pasados por parámetro, y el cual se descomprimirá con su algoritmo, seleccionado automáticamente en función de su extensión.
    - Parámetros:
        - Path: string que indica dónde se encuentra el archivo a comprimir.
        - Algoritmo: algoritmo que se desea utilizar para realizar la compresión

- setAlgoritmoDeTextoPredeterminado: Pública, se le pasa por paràmetro un Algoritmo algoritmoDeTextoPredeterminado. Asigna un algoritmo de texto predeterminado al Singleton de CtrlProcesos.

    - Parámetro:
        - Algoritmo: algoritmo que se desea que sea el predeterminado

- getAlgoritmoDeTextoPredeterminado: Pública, devuelve un Algoritmo de texto. Obtiene el algoritmo de texto predeterminado del Singleton de CtrlProcesos.

- setCalidadJPEG: Pública. Llamará al método de setCalidad de la clase JPEG para que asigne la calidad de la compresión al Singleton JPEG.
    - Parámetro:
        - CalidadJPEG: indica la calidad de compresión deseada, entre 1 y 7.

- algoritmosPosibles: Pública, estática, tiene como parámetro de entrada un string path y devolverá un array de Algoritmo. Se utiliza para comprobar el tipo del archivo del path para comprobar qué algoritmos se podrán usar para comprimir o descomprimir.
    - Parámetro:
        - Path: string que indica dónde se encuentra un archivo con una extensión determinada.
    - Return:
        - Algoritmo [ ]: set de algoritmos capaces de tratar el archivo indicado en el path.

- esComprimible: Pública y estática. Comprueba si el archivo del path es capaz de ser comprimido según la extensión del mismo.
    - Parámetros:
        - Path: string que indica dónde se encuentra un archivo que se desea saber si es comprimible.
    - Return:
        - Boolean indicando si es comprimible con alguno de los compresores disponibles.

- esDescomprimible: Pública y estática. Comprueba si el archivo del path es capaz de ser descomprimido según la extensión del mismo.
    - Parámetros:
        - Path: string que indica dónde se encuentra un archivo que se desea saber si es descomprimible.
    - Return:
        - Boolean indicando si es descomprimible con alguno de los descompresores disponibles

CtrlEstadística

La clase CtrlEstadística es un controlador que interactúa con las clases de la capa de dominio.
Atributos:

- Instance: Privado, estático, es la instancia de CtrlEstadística para garantizar que es una clase Singleton.

Métodos:

- getInstance: Pública y estática, getter de la instancia Singleton del controlador CtrlEstadística y devuelve la instancia Singleton de CtrlEstadística.
- estadisticas: Pública, construye un mensaje con la información de la compresión y descompresión del algoritmo que se le pasa por paràmetro y devuelve un string con toda esta información. Se encarga de construir un mensaje con toda la información relevante de compresión y descompresión del algoritmo solicitado.
    - Parámetros:
        - algoritmo: El tipo del algoritmo del que se desea obtener las estadísticas
    - Result: Devuelve un String con toda la información preparada para ser presentada, de tal manera que se pueda visualizar fácilmente los datos interesantes de la estadística.

CtrlDatos

La clase CtrlDatos es un controlador que interactúa con las clases de la capa de datos.
Atributos:

- Instance: Privado, estático, es la instancia de CtrlDatos para garantizar que es una clase Singleton.

Métodos:

- getInstance: Pública y estática, getter de la instancia Singleton del controlador CtrlDatos y devuelve la instancia Singleton de CtrlDatos.
- actualizaEstadistica: Pública, actualiza el fichero de estadística de compresión o descompresión del algoritmo correspondiente que se encuentra en el path con los datos del nuevo archivo procesado.
    - Parámetros:
        - dp: Es una instancia de DatosProceso donde se consigue la información estadística del proceso.
        - algoritmo: El Algoritmo del que actualizamos el fichero estadística.
        - esCompresion: Boolean que indica si el fichero estadística es de compresión o descompresión.
- getNumeroElementos: Pública, obtiene el número de elementos que contiene la estadística para compresión o descompresión del algoritmo seleccionado.
    - Parámetros:
        - algoritmo: El Algoritmo que se quiere consultar.
        - esCompresion: El tipo de proceso que se quiere, consultar, si de compresión o descompresión.
    - Result: Devuelve el número de elementos que contiene la estadística indicada.
    - Excepciones:
        - Exception: Ha existido algún tipo de problema accediendo al archivo o no existe.
- getPorcentajeAhorradoMedio: Pública, obtiene el porcentaje medio que representan los archivos procesados respecto al original de la estadística para compresión o descompresión del algoritmo seleccionado.
    - Parámetros:
        - algoritmo: El Algoritmo que se quiere consultar.
        - esCompresion: El tipo de proceso que se quiere, consultar, si de compresión o descompresión.
    - Result: Devuelve el número de elementos que contiene la estadística indicada.

    - Excepciones:
        - Exception: Ha existido algún tipo de problema accediendo al archivo o no existe.
- getTiempoMedio: Pública, obtiene el tiempo medio de la estadística para compresión o descompresión del algoritmo seleccionado.
    - Parámetros:
        - algoritmo: El Algoritmo que se quiere consultar.
        - esCompresion: El tipo de proceso que se quiere, consultar, si de compresión o descompresión.
    - Result: Devuelve el número de elementos que contiene la estadística indicada.
    - Excepciones:
        - Exception: Ha existido algún tipo de problema accediendo al archivo o no existe
- guardaArchivo: Pública, llama a la clase GestorArchivo para que guarde el archivo del path.
    - Pre: El path del archivo debe seguir el formato general de cualquier tipo de path de archivo y puede ser relativo o absoluto.
    - Parámetros:
        - data: Un byte [] de los datos que se guardaran en el archivo.
        - path: Un string que indica el path del archivo donde se quiere guardar.
        - algoritmo: El Algoritmo con el qual se ha comprimido o descomprimido el archivo.
        - esCompresion: Un boolean que indica el tipo de proceso de compresión o descompresión.
        - sobreescribir: Un boolean que indica si se va a sobreescribir.
    - Excepciones:
        - Exception: No se ha podido escribir en el archivo por el path indicado.
- leerArchivo: Pública, llama a la clase GestorArchivo para que lea el archivo del path.
    - Pre: El path del archivo debe seguir el formato general de cualquier tipo de path de archivo y puede ser relativo o absoluto.
    - Parámetros:
        - path: Un string que indica el path del archivo que se quiere leer.
    - Result: Un byte[] del contenido del archivo del path.
    - Excepciones:
        - Exception: No hay ningún archivo en el path que se pasa o no se ha leído bien.

DatosProceso

DatosProceso es una clase que contiene los datos relevantes para la generación de estadísticas de un proceso fichero. Esta clase permite generar estadísticas a partir de los datos que almacena, tanto a nivel de proceso individual como al poder ser usado por el generador de estadísticas globales

Atributos:

- Tiempo: Tiempo de ejecución del proceso de compresión o descompresión de un fichero.
- oldSize: Tamaño del fichero tratado antes del proceso de compresión o descompresión.
- newSize: Tamaño del fichero tratado tras la ejecución del proceso de compresión o descompresión.
- esCompresión: Indicación para saber si es un proceso de compresión o descompresión.
- satisfactorio: Indicación para saber si el proceso ha resultado satisfactorio o no según los datos del proceso.

Métodos:

- DatosProceso: Constructora de la clase, la cual crea una instancia de DatosProceso con un tiempo, oldSize, newSize y esCompresion asignados.
    - Parámetros:
        - time: El tiempo de ejecución del proceso de compresión o descompresión de un fichero.
        - oldSize: Tamaño del fichero tratado antes del proceso de compresión o descompresión.
        - newSize: Tamaño del fichero tratado tras la ejecución del proceso de compresión o descompresión.
        - esCompresion: Indicación para saber si es un proceso de compresión o descompresión.
    - Excepciones:
        - Exception: En el caso que la compresión haya sido perjudicial porque el tamaño resultante ha sido mayor al original.
- getDiffSize: Devuelve la diferencia entre el original size y el new size.
- getDiffSizePercentage: Devuelve el porcentaje de tamaño que representa el nuevo archivo respecto al original.
- getTiempo: Obtiene el nuevo tamaño del fichero sobre el que trabaja el proceso.
- getNewSize: Obtiene el nuevo tamaño del fichero sobre el que trabaja el proceso.
- getOldSize: Obtiene el antiguo tamaño del fichero sobre el que trabaja el proceso.
- isSatisfactorio: Obtiene una indicación de si el proceso ha resultado satisfactorio o no.

ProcesoFichero

Clase abstracta que recoge todos los datos necesarios para poder comprimir y descomprimir con cualquier tipo de algoritmo.
También se encarga de recoger todos los datos posteriores a la ejecución del proceso como medidas para estadística.

Atributos:

- input: Secuencia de bytes a procesar.
- output: Secuencia de bytes procesada. Declarada como null hasta que no se ejecute el proceso.
- tipoAlgoritmo: Algoritmo que utilizará el proceso a comprimir.
- compresorDecompresor: Instancia del algoritmo que utilizará el proceso.
- procesado: Indica si el proceso ya ha sido ejecutado o no.
- datos: Instancia de los datos estadísticos sobre el proceso ejecutado. Declarada como null hasta que no se ejecute el proceso.

Métodos:

- ProcesoFichero: Constructora de proceso de un fichero. Dado que es una clase abstracta no se puede llamar directamente, sólo a través de las clases que extienden de ProcesoFichero.
    - Pre: El algoritmo debe ser adecuado el adecuado para poder comprimir o descomprimir la clase adecuadamente.
    - Excepciones**:**
        - input: Secuencia de bytes a procesar.
        - algoritmo: Algoritmo de compresión a utilizar para procesar el input.

- getDatosProceso: Devuelve la instancia de DatosProceso del proceso ejecutado. Si este aún no se ha ejecutado la instancia es null.
- asignarAlgoritmo: Asigna la instancia del algoritmo adecuado según el valor de tipoAlgoritmo.
- ejecutarProceso: Método abstracto que ejecuta el proceso de compresión o descompresión del input y genera la instancia para estadística. Puede generar una excepción si el algoritmo no se ha ejecutado correctamente y no ha podido terminar.
- esComprimir: Devuelve un booleano indicando si el proceso está creado con intención de comprimir.
- isProcesado: Devuelve un booleano indicando si el proceso ya ha sido ejecutado.
- getInput: Devuelve la secuencia de bytes a procesar.
- getOutput: Devuelve la secuencia de bytes procesada.
- getTipoCompresor: Devuelve el algoritmo que se ha asignado al proceso en el momento de su creación.

ProcesoComprimir

Clase que recoge todos los datos necesarios para poder comprimir con cualquier tipo de algoritmo. También se encarga de recoger todos los datos posteriores a la ejecución del proceso como medidas para estadística.

Métodos:

- ProcesoComprimir: Constructora de proceso de compresión de un fichero.
    - Pre: El algoritmo debe ser adecuado el adecuado para poder comprimir la clase adecuadamente.
    - Parámetros**:**
        - input: Secuencia de bytes a procesar.
        - algoritmo: Algoritmo de compresión a utilizar para procesar el input.
- ejecutarProceso: Método que ejecuta el proceso de compresión y genera la instancia para estadística. Puede generar una excepción si el algoritmo no se ha ejecutado correctamente y no ha podido terminar.
- esComprimir: Devuelve un booleano indicando que el proceso está creado para comprimir el archivo.

ProcesoDescomprimir

Clase que recoge todos los datos necesarios para poder descomprimir con cualquier tipo de algoritmo. También se encarga de recoger todos los datos posteriores a la ejecución del proceso como medidas para estadística.

Métodos:

- ProcesoDescomprimir: Constructora de proceso de descompresión de un fichero.
    - Pre: El algoritmo debe ser adecuado el adecuado para poder descomprimir la clase adecuadamente.
    - Parámetros**:**
        - input: Secuencia de bytes a procesar.
        - algoritmo: Algoritmo de compresión a utilizar para procesar el input.

- ejecutarProceso: Método que ejecuta el proceso de descompresión, guarda el resultado en output y genera la instancia para estadística. Puede generar una excepción si el algoritmo no se ha ejecutado correctamente y no ha podido terminar.
- esComprimir: Devuelve un booleano indicando que el proceso no está creado para comprimir el archivo.

OutputAlgoritmo

OutputAlgoritmo es una clase que pretende emular un struct de los lenguajes de la familia de C. Esta clase permite que los algoritmos de compresión y descompresión puedan devolver la tupla de datos resultante de haber llevado a cabo la ejecución del proceso.

Atributos:
- tiempo: Tiempo de ejecución de la compresión o descompresión de un fichero.
- output: Byte array continente de la información comprimida o descomprimida tras haber sido sometida a la ejecución del proceso.

Métodos:
- OutputAlgoritmo: Constructora de la clase, la cual crea una instancia de OutputAlgoritmo con un tiempo y output asignados.
    - Parámetros:
        - tiempo: El tiempo de ejecución de la compresión o descompresión de un fichero.
        - output: El byte array contenedor de la información comprimida o descomprimida tras haber sido sometida a la ejecución del proceso.

CompresorDecompresor

Interfaz que define los métodos básicos de un algoritmo de compresión.

Métodos:
- comprimir: Comprime la secuencia de bytes correspondiente al contenido de un archivo.
    - Parámetros:
        - datosInput**:** Secuencia de bytes a comprimir. Debe ser adecuada
    - Return: Devuelve una instancia de OutputAlgoritmo con el tiempo que ha tardado la compresión y con la secuencia de bytes comprimida.
    - Excepciones:
        - FormatoErroneoException: En caso de que el formato del archivo no sea el adecuado para ser comprimido.
- descomprimir: Descomprime la secuencia de bytes correspondiente al contenido de un archivo.
    - Pre: La secuencia debe haber sido comprimida por el método comprimir(byte[]) para que el proceso de descompresión sea satisfactorio.
    - Parámetros:
        - datosInput: Secuencia de bytes a descomprimir.
    - Return: Devuelve una instancia de OutputAlgoritmo con el tiempo que ha tardado la descompresión y con la secuencia de bytes descomprimida.
    - Excepciones:

            - FormatoErroneoException: En caso de que el formato del archivo no sea el adecuado para ser descomprimido.

JPEG

La clase Singleton JPEG es la encargada de procesar archivos de imagen de extensión .ppm o .imgc (.ppm comprimido con atributo extra de calidad de compresión usada) y comprimirlos y descomprimirlos respectivamente.

Atributos:

- Instance (static): Instancia de JPEG para garantizar que es una clase Singleton.
- Calidad: Escalar usado para modular la calidad de compresión y descompresión de las imágenes.
- CalidadHeader: Valor de calidad usado para escribirlo en el header de las imágenes comprimidas con la intención de saber con qué calidad se tienen que descomprimir para que el resultado sea el esperado.
- LuminanceQuantizationTable (static): Tabla de cuantización usada para comprimir y descomprimir los valores de luminosidad (Y) de los cuadrados de píxeles de dimensiones 8x8. Este atributo es estático.
- ChrominanceQuantizationTable (static): Tabla de cuantización usada para comprimir y descomprimir los valores de crominancia (Cb y Cr) de los cuadrados de píxeles de dimensiones 8x8. Este atributo es estático.

Métodos:

- getInstance: Getter de la instancia Singleton de JPEG.
- JPEG: Constructora de JPEG, la cual inicializa todos los atributos de clase excepto el propio instance, del cual se encarga getInstance.
- setCalidad: Asigna un valor de calidad de compresión al Singleton de JPEG.
    - Pre:
        - Su valor se sitúa entre 1 y 7. En caso contrario se corrige al valor extremo más cercano.
    - Parámetros:
        - calidad: La calidad de compresión de JPEG. A mayor valor, más alta será la calidad de la compresión.
- comprimir: Algoritmo de compresión JPEG.
    - Pre:
        - La imagen de entrada tiene que ser .ppm.
    - Parámetros:
        - datosInput: Un byte array contenedor de una imagen a comprimir.
    - Result: Un byte array contenedor de una imagen comprimida.
    - Excepciones:
        - FormatoErroneoException: El formato en el que está codificada la imagen .ppm no es correcto.
- descomprimir: Algoritmo de descompresión JPEG.
    - Pre: La imagen a descomprimir tiene que haber sido comprimida por el método comprimir.
    - Parámetros:
        - datosInput: Un byte array contenedor de una imagen a descomprimir.

    - Return: Un byte array contenedor de una imagen descomprimida.
    - Excepciones:
        - FormatoErroneoException: El formato en el que está codificada la imagen comprimida .imgc no es correcto.

LZSS

Clase que implementa los métodos necesarios para comprimir y descomprimir secuencias de bytes con el algoritmo de compresión LZSS.

Atributos:

- instance: Instancia de LZSS para garantizar que es una clase Singleton.

Métodos:

- LZSS: Constructor de la clase LZSS. Su visibilidad es privada para impedir que se tengan múltiples instancias de la clase y satisfacer las propiedades de Singleton.
- getInstance: Método que devuelve la instancia la instancia de la clase LZSS.
- comprimir: Comprime la secuencia de bytes aportada según el patrón de compresión LZSS.
    - Parámetros:
        - datosInput: Secuencia de bytes a comprimir.
    - Return: Devuelve una instancia de OutputAlgoritmo compuesta del tiempo que ha tardado el método en ejecutarse y la secuencia de bytes comprimida.
- descomprimir: Descomprime la secuencia de bytes aportada según el patrón de compresión. LZSS.
    - Pre: La secuencia debe haber sido comprimida por el método comprimir(byte[]) para que el proceso de descompresión sea satisfactorio.
    - Parámetros:
        - datosInput: Secuencia de bytes a descomprimir.
    - Return: Devuelve una instancia de OutputAlgoritmo compuesta del tiempo que ha tardado el método en ejecutarse y la secuencia de bytes descomprimida.
    - Excepciones:
        - FormatoErroneoException: El formato en el que está codificado el texto no es correcto.

LZW

La clase Singleton LZW es la encargada de procesar archivos de texto de extensión .txt o .lzw y comprimirlos y descomprimirlos respectivamente.

Atributos:

- Instance: Instancia de LZW privada y estática para garantizar que es una clase Singleton.

Métodos:

- getInstance: Es una función pública y estática, que hace getter de la instancia Singleton de LZW, la utilizara otra clase para utilizar las funciones del controlador. Devuelve la instancia Singleton de LZW
- LZW: Privado, estático, constructora de LZW. Su visibilidad es privada para impedir que se tengan múltiples instancias de la clase y satisfacer las propiedades de Singleton.
- comprimir: Comprime la secuencia de bytes aportada según el patrón de compresión LZW.
    - Parámetros:

        - datosInput: Secuencia de bytes a comprimir.
    - Return: Devuelve una instancia de OutputAlgoritmo compuesta del tiempo que ha tardado el método en ejecutarse y la secuencia de bytes comprimida.
- descomprimir: Descomprime la secuencia de bytes aportada según el patrón de compresión LZW.
    - Pre: La secuencia debe haber sido comprimida por el método comprimir(byte[]) para que el proceso de descompresión sea satisfactorio.
    - Parámetros:
        - datosInput: Secuencia de bytes a descomprimir.
    - Return: Devuelve una instancia de OutputAlgoritmo compuesta del tiempo que ha tardado el método en ejecutarse y la secuencia de bytes descomprimida.
    - Excepciones:
        - FormatoErroneoException: El formato en el que está codificado el texto no es correcto.

LZ78

Clase que implementa los métodos necesarios para comprimir y descomprimir secuencias de bytes con el algoritmo de compresión LZ78.
Atributos:

- instance: Instancia de LZ78 para garantizar que es una clase Singleton.

Métodos:

- LZ78: Constructor de la clase LZ78. Su visibilidad es privada para impedir que se tengan múltiples instancias de la clase y satisfacer las propiedades de Singleton.
- getInstance: Método que devuelve la instancia la instancia de la clase LZ78. Return: Devuelve la instancia de la clase LZ78.
- comprimir: Comprime la secuencia de bytes aportada según el patrón de compresión LZ78.
    - Parámetros:
        - datosInput: Secuencia de bytes a comprimir.
    - Return: Devuelve una instancia de OutputAlgoritmo compuesta del tiempo que ha tardado el método en ejecutarse y la secuencia de bytes comprimida.
- descomprimir: Descomprime la secuencia de bytes aportada según el patrón de compresión LZ78.
    - Pre: La secuencia debe haber sido comprimida por el método comprimir(byte[]) para que el proceso de descompresión sea satisfactorio.
    - Parámetros:
        - datosInput: Secuencia de bytes a descomprimir.
    - Return: Devuelve una instancia de OutputAlgoritmo compuesta del tiempo que ha tardado el método en ejecutarse y la secuencia de bytes descomprimida.
    - Excepciones:
        - FormatoErroneoException: El formato en el que está codificado el texto no es correcto.

Pair

Clase Pair usada en descomprimir(byte[]) para procesar todos los índices y bytes de la secuencia de bytes comprimidos.

Atributos:

- index: Índice del byte predecesor.
- b: Byte que representa el componente final de una secuencia de bytes comprimida.

Métodos:

- Pair: Constructora.
    - Parámetros:
        - index: Índice del byte predecesor.
        - b**:** Byte que representa el componente final de una secuencia de bytes comprimida.

Trie

Estructura de Datos Trie que consiste en un árbol con pesos en las ramas de cada nodo. Esta estructura es utilizada por LZ78.

Atributos:

- root: Nodo raíz desde el que empiezan todas las ramas.
- indexCount: Contador del tamaño del diccionario. Su rango es entre 0 y 4194303.

Métodos:

- trie: Constructora.
- Insert: Añade al árbol la secuencia de bytes aportada por parámetro.
    - Pre: Todos los elementos de la secuencia menos el último deben existir previamente en el árbol.
    - Parámetros:
        - bytes**:** Conjunto de bytes a añadir al árbol.
    - Return:
        - Devuelve el índice que precede al nuevo nodo añadido.
- search: Busca si la secuencia de bytes se encuentra en el árbol o no.
    - Parámetros:
        - bytes: Secuencia de bytes a localizar en el árbol.
    - Return:
        - Devuelve un booleano indicando si la secuencia de bytes se encuentra dentro del árbol o no.
- searchNode: Retorna el último nodo de la secuencia de bytes solicitada por parámetro. Si la secuencia no existe el valor retornado es null.
    - Parámetros:
        - bytes: Secuencia de bytes a localizar en el árbol.
    - Return:
        - Devuelve el último nodo de la secuencia de bytes solicitada. Si no existe el valor es null.
- isFull: Indica si el árbol ha alcanzado su límite de tamaño.
    - Return: Devuelve un booleano indicando si el árbol se ha llenado.

TrieNode

Clase que representa cada nodo dentro del árbol Trie.

Atributos:

- c: Byte que forma parte de un conjunto de secuencias de bytes. Siempre es el último byte de una y solo una secuencia.
- index: Indica el número de la secuencia que acaba en este nodo.
- edges: Ramas que surgen del nodo con pesos que indican el byte del nodo correspondiente.
- isLeaf: Indica si el nodo no tiene hijos.

Métodos:

- TrieNode: Constructora sin parámetros.
- TrieNode: Constructora con los parámetros básicos del nodo.
    - Parámetros:
        - c: Byte perteneciente a una secuencia de bytes seguida.
        - index: Indica el índice del byte que precede al actual.