**Coloreado de grafos**

***UO285176***

*Eduardo Blanco Bielsa*

* **Tabla de tiempos**

|  |  |  |
| --- | --- | --- |
| Tamaño | ms | Tiempo esperado (ms) |
| 8 | 23 |  |
| 16 | 22 | 46 |
| 32 | 29 | 44 |
| 64 | 52 | 58 |
| 128 | 77 | 104 |
| 256 | 146 | 154 |
| 512 | 192 | 292 |
| 1024 | 343 | 384 |
| 2048 | 565 | 686 |
| 4096 | 1141 | 1130 |
| 8192 | 2166 | 2282 |
| 16384 | 4322 | 4332 |
| 32768 | 9678 | 8644 |
| 65536 | 18673 | 19356 |
| 131072 | 37154 | 37346 |
| 262144 | 72808 | 74308 |
| 524288 | 143637 | 145616 |

1. Mi algoritmo se organiza en tres ficheros: ColoracionMapa.java, FileUtil y Main, así como los ficheros con los colores y las fronteras.

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

* La clase **FileUtil** se encarga de obtener los colores y las fronteras de los ficheros mencionados por medio de los métodos readColors y readCountries respectivamente.
* La clase **Main** crea el mapa y ejecuta el algoritmo llamando al constructor de la clase **ColoracionMapa**. Además, en ella se miden los tiempos de la tabla de tiempos anterior.
* La clase **ColoracionMapa** realiza el algoritmo. En el contructor se invocan tres métodos: sacarColores(), que obtiene los colores y los guarda en un array unidimensional, sacarPaisesYFronteras(), que obtiene los países y los adyacentes guardándolos en un HashMap y colorear() que realiza el algoritmo de coloración del mapa.

Interfaz de usuario gráfica, Texto, Aplicación, Chat o mensaje de texto

Descripción generada automáticamente

* El algoritmo se basa en dos métodos:

Texto

Descripción generada automáticamente

* Este método recorre las claves del HashMap de países, es decir, los países, y para cada país recorre el array de colores para asignarle el óptimo. En caso de que el país no tenga fronteras, se le asignará siempre el primer color y se añadirá al HashMap del mapa solución. De otro modo, se llama al siguiente método privado:

Texto

Descripción generada automáticamente

* Este método busca en el HashMap de los países el país pasado como parámetro del método previo y va recorriendo sus países adyacentes o fronteras. Si en el HashMap solución el adyacente ya ha sido coloreado y tiene el color que pasamos como parámetro devuelve false. En otro caso devolverá true.

1. En mi caso han sido 4 colores. Se adjunta la siguiente imagen de la ejecución del algoritmo como **PruebaGrafos.jpg** para más detalle:

Mapa

Descripción generada automáticamente

1. Sí que podría cambiar el número de colores si se usa un orden distinto.
2. 4 colores como mínimo sería la solución óptima en este mapa. Sin embargo, este número podría variar en función del mapa que intentásemos colorear.

Texto

Descripción generada automáticamente

* La complejidad temporal de mi algoritmo se corresponde con O(n), ya que el bucle que recorre los países tiene un tamaño n, el bucle que recorre el array de colores tiene complejidad 1 porque son siempre 12 colores y el método privado colorUsado() tiene una complejidad 1 porque siempre recorre un número finito de países adyacentes, que es mucho menor al número de países.