Programación 1

Comenzamos en 10 minutos

Ejercicio Tipo Recuperatorio

Enunciado

Un dispositivo que toma imágenes de un fenómeno natural está corriendo sobre una plataforma tecnológica con recursos limitados.
Cada imagen (representada por una matriz de **NxM**)
Está compuesta por píxeles con valores entre 0 y 255.
Se tiene implementado un algoritmo de compresión que comprime aquellas porciones de la imagen distintas del color negro (0 en la escala de valores del pixel).
Dicho algoritmo procede de la siguiente manera: por cada una de las filas de la matriz, toma cada secuencia delimitada por uno o más pixeles de color negro (valor 0) con más de X repeticiones de un valor de píxel (para ser comprimida, todos los elementos de la secuencia deben ser iguales), comprime la secuencia poniendo en la primera posición el valor negado de la cantidad de ocurrencias y a continuación el valor del pixel que se repite.
Cada fila de la matriz empieza y termina con uno o más pixeles negros.
Se pide implementar el algoritmo de descompresión que restablezca la matriz original. Asumir que cada fila posee suficientes lugares como para realizar la descompresión.

Ejemplo de matriz comprimida con X = 3:

Enunciado

0 -8 67 0 14 0 -4 33 0 5 98 0 0 0 0 0 0 0 0 0

0 0 25 25 0 -5 3 0 25 44 44 0 -4 1 0 0 0 0 0 0

0 44 44 44 0 -7 15 0 -4 9 0 12 0 0 0 0 0 0 0 0

La matriz resultante quedaría de la siguiente forma:

0 67 67 67 67 67 67 67 67 0 14 0 33 33 33 33 0 5 98 0

0 0 25 25 0 3 3 3 3 3 0 25 44 44 0 1 1 1 1 0

0 44 44 44 0 15 15 15 15 15 15 15 0 9 9 9 9 0 12 0

Además se debe informar la cantidad total de píxeles descomprimidos y la fila en la cual se encuentra la mayor cantidad de píxeles descomprimidos (la primera, si hubiese más de una).
En este ejemplo, la cantidad total de píxeles descomprimidos fue 32 y la fila con más píxeles descomprimidos fue la fila 0.

```
public class EjercicioTipoRecuperatorio2024 {
  final static int COLUMNAS=20;
  final static int FILAS=3;
  final static int SEPARADOR=0;
  final static int X=3;
  public static void main(String[]args){

          int[][] imagen={
                  {0,-8,67,0,14,0,-4,33,0,5,98,0,0,0,0,0,0,0,0,0},
                  {0,0,25,25,0,-5,3,0,25,44,44,0,-4,1,0,0,0,0,0,0},
                  {0,44,44,44,0,-7,15,0,-4,9,0,12,0,0,0,0,0,0,0,0}
              };
          imprimir_matriz(imagen);
          descomprimir_matriz(imagen);
          imprimir_matriz(imagen);
  }
  public static void descomprimir_matriz(int[][]imagen){
     //Falta Codificar
   }
}
```

Lo principal que pide el enunciado es **descomprimir** la **matriz**

```java
public static void descomprimir_matriz(int[][]imagen){
  int maxDescomprimidosFila=0;
  int maxFila=0;
  int totalDescomprimidos=0;
  for (int fila=0; fila<FILAS; fila++){
      int descomprimidosFila = descomprimir_fila(imagen[fila]);
      if (descomprimidosFila>maxDescomprimidosFila){
                maxDescomprimidosFila=descomprimidosFila;
                maxFila=fila;
      }
      totalDescomprimidos+=descomprimidosFila;
  }

  if (totalDescomprimidos>0){
        System.out.println("La cantidad Máxima de caracteres descomprimidos es de "+totalDescomprimidos
                           +" La fila con mas pixeles descomprimidos es fila "+maxFila);
    }else{
        System.out.println("Nada por descomprimir");
    }

}
```

Tan simple como recorrer cada fila y descomprimirla

Me quedo con la cantidad Máxima y la Fila

SUMO y acumulo la cantidad de descomprimidos

CUIDADO: Donde se ubica cada una de las asignaciones!!!!!

```
public static int descomprimir_fila (int[] arr){
    int totalDescomprimidos =0;
    int ini=0, fin=-1;
    while (ini<COLUMNAS){
        ini = obtener_inicio_secuencia (arr, fin+1);
        if (ini<COLUMNAS){
            fin = obtener_fin_secuencia (arr, ini);

            int comprimidos = -arr[ini];
            int pixel = arr[ini+1];
            if (comprimidos>0){
                descomprimir_secuencia (arr,ini,comprimidos,pixel );
                totalDescomprimidos +=comprimidos;

            }
        }
    }
    return totalDescomprimidos ;
}
```

Búsqueda Clásica de Secuencias

Verifico si es una secuencia comprimida
Recordar que el número es negativo si es así

Descomprimo y sumo el total

Falta Algo???

Que pasó al descomprimir?

Teníamos por ejemplo la primer secuencia encontrada en la fila 0:

0 -8 67 0 14 0 -4 33 0 5 98 0 0 0 0 0 0 0 0 0

-8 67 es la primer secuencia a descomprimir, de Tamaño 2, en las posiciones 1 y 2 de la fila

Pasa a ser

67 67 67 67 67 67 67 67 Tamaño 8 y la fila queda de la siguiente manera

0 67 67 67 67 67 67 67 0 14 0 -4 33 0 5 98 0 0 0

Es decir, el final de la secuencia era en un principio de inicio en 1 y fin 2, pasó a ser de inicio 1 y fin 8

El final de la secuencia lo da:

fin original + cantidad de pixeles a descomprimir - 2 lugares que ya ocupaba la secuencia

```
fin=fin + comprimidos  - 2;
```

```java
public static int descomprimir_fila (int[] arr){
    int totalDescomprimidos =0;
    int ini=0, fin=-1;
    while (ini<COLUMNAS){
        ini = obtener_inicio_secuencia (arr, fin+1);
        if (ini<COLUMNAS){
            fin = obtener_fin_secuencia (arr, ini);
            int comprimidos = -arr[ini];
            int pixel = arr[ini+1];
            if (comprimidos>0){
                descomprimir_secuencia(arr,ini,comprimidos,pixel );
                totalDescomprimidos +=comprimidos;
                fin=fin + comprimidos - 2;
            }
      }
    }
    return totalDescomprimidos;
```

Búsqueda Clásica de Secuencias

Verifico si es una secuencia comprimida
Recordar que el número es negativo si es así

Descomprimo y sumo el total

Como el tamaño cambió establezco nuevo fin

Nos falta implementar este método

```
public static void descomprimir_secuencia (int[]arr, int pos, int cantidad, int pixel ){
```

Ej: -8 67 (uno de los elementos ya existe)

```
  arr[pos]=pixel;          //Reemplazo la cantidad de pixeles por el elemento
  cantidad = cantidad-2;   //Dos elementos ya existen

  while (cantidad>0){
      corrimiento_derecha (arr, pos);
      arr[pos]=pixel;
      cantidad--;
  }

}
```

Continúo eliminando los que quedan

Algo más?

Sólo las funciones y métodos que ya sé de las prácticas

```
public static int obtener_fin_secuencia (int[] arrEnteros, int ini) {
        while (ini<COLUMNAS && arrEnteros[ini]!=SEPARADOR)
            ini++;
        return ini-1;
    }
    public static int obtener_inicio_secuencia (int[] arrEnteros, int ini) {
        while (ini<COLUMNAS && arrEnteros[ini]==SEPARADOR)
            ini++;
        return ini;
    }
    public static void corrimiento_derecha (int[]arr, int pos){
        for (int i=COLUMNAS-1; i>pos; i--){
            arr[i] = arr[i-1];
        }
    }
    public static void imprimir_arreglo (int [] arr) {...}
    public static void imprimir_matriz (int [][] arr){...}
```

A que se reduce la resolución?

```
public class EjercicioTipoRecuperatorio2024 {
  final static int COLUMNAS=20;
  final static int FILAS=3;
  final static int SEPARADOR=0;
  final static int X=3;
  public static void main(String[]args){
   int[][] imagen={
       {0,-8,67,0,14,0,-4,33,0,5,98,0,0,0,0,0,0,0,0,0},
       {0,0,25,25,0,-5,3,0,25,44,44,0,-4,1,0,0,0,0,0,0},
       {0,44,44,44,0,-7,15,0,-4,9,0,12,0,0,0,0,0,0,0,0}
   };
   imprimir_matriz(imagen);
   descomprimir_matriz(imagen);
   imprimir_matriz(imagen);
 }
 public static void descomprimir_matriz(int[][]imagen){
   int maxDescomprimidosFila=0;
   int maxFila=0;
   int totalDescomprimidos=0;
   for (int fila=0; fila<FILAS; fila++){
       int descomprimidosFila = descomprimir_fila(imagen[fila]);
       if (descomprimidosFila>maxDescomprimidosFila){
           maxDescomprimidosFila=descomprimidosFila;
           maxFila=fila;
       }
       totalDescomprimidos+=descomprimidosFila;
   } …
 }
```

A que se reduce la resolución?

```
public static int descomprimir_fila(int[] arr){
   int totalDescomprimidos=0;
   int ini=0, fin=-1;
   while (ini<COLUMNAS){
       ini = obtener_inicio_secuencia(arr, fin+1);
       if (ini<COLUMNAS){
           fin = obtener fin secuencia(arr, ini);
           int comprimidos = -arr[ini];
          int pixel = arr[ini+1];
           if (comprimidos>0){
               descomprimir_secuencia(arr,ini,comprimidos,pixel);
               totalDescomprimidos+=comprimidos;
               fin+=comprimidos-2;//Se suman los agregados y se descuentan las dos existentes
           }
       }
   }
   return totalDescomprimidos;
 }
 public static void descomprimir_secuencia(int[]arr, int pos, int cantidad, int pixel){
   arr[pos]=elemento;
   cantidad = cantidad-2; //dos elementos ya existen
   while (cantidad>0){
       corrimiento_derecha(arr, pos);
       arr[pos]=elemento;
       cantidad--;
   }
 }
```