Explicación de los juegos de pruebas
Archivos de texto
Imagenes y algoritmo JPEG

Explicación de los juegos de pruebas

Archivos de texto

Para probar diferentes escenarios hemos creado diversos archivos de texto e imagenes.

- **Blank.txt:** Archivo vacio, para comprobar que al leer un documento con los algoritmes LZ se controla la posibilidad de intentar acceder a información inexistente y que efectivamente se activa la excepción adecuada.
- **DonQuijote.txt:** Archivo que contiene El Quijote de Cervantes, ejemplo de texto medianamente grande, con el cual se comprueba que el diccionario utilizado en el caso de LZ78 y LZW es suficientemente grande y que en caso de overflow, éste este controlado.

```
texts               Textos de prueba
├── Blank.txt   Archivo vacío
|
├── DonQuijote.txt  1,1MB
|     LZ78: compresión 40.35%, 0.41s   | descompresión 0.23s
|     LZSS: compresión 43.66%, 15,30s  | descompresión 0.60s
|     LZW:  compresión 31.66%, 0,29s   | descompresión 0.14
|
├── Large.txt   17,9MB
|     LZ78: compresión 41.68%, 6.32s   | descompresión 5,15s
|     LZSS: compresión 43.96%, 262,35s | descompresión 13,47s
|     LZW:  compresión 41.44%, 6,47s   | descompresión 4,68s
|
├── Prova.txt   14B
|     La compresión en los tres casos es casi inexistente
|
├── Repeticion.txt  6,7MB
|     LZ78: compresión 98.75%, 5,97s   | descompresión 0.06s
|     LZSS: compresión 96.12%, 7,41s   | descompresión 4.50s
|     LZW:  compresión 98.62%, 3,17s   | descompresión 0.06s
|
└── Diferentes.txt  86B
      La compresión en los tres casos es negativa, es decir, el archivo
comprimido ocupa más que es original.
```

- **Large.txt:** Archivo condierablemente grande (17,9MB), el objectivo del cual es comprobar la capacidad de compresión de los algoritmos y su comportamiento al haber de tratar con archivos de estas dimensiones.

- **Prova.txt:** Archivo de prueba muy básico utilitzado al principio para ver el correcto funcionamiento de los algoritmos.
- **Repeticion.txt:** Archivo consistente de una frase con muchos caracteres iguales, repetida una gran cantidad de veces (6,7MB). Con esta prueba queriamos comprobar la compresión de un archivo con un alto nivel de repetición de información.
- **Diferentes.txt:** Archivo con todos los caracteres diferentes. De manera que se comprueba que al intentar comprimir un archivo con repeticiones se obtiene una compresión nula, o incluso se obtiene un archivo de compresión mayor ya que en éste los caracteres ocupan 16 bits en lugar de 8.

Imagenes y algoritmo JPEG

Por otro lado tenemos diferentes escenarios para el algoritmo JPEG:

- **boat.ppm** y **cliff.ppm** son imagenes estandard de ~3MB. Tardan ~10s en comprimir / descomprimir. (compression ~80% a calidad 90)
- **church.ppm** y **lake.ppm** son imagenes muy grandes de ~46MB. Tardan unos 140s en comprimir/descomprimir (compression ~80% a calidad 90)
- **gonza.ppm** es una imagen pequena ~1MB y muy mala calidad. Se consigue una compression del 94% con calidad 90.
- Las imagenes **gradient** son gradientes, que permiten ver facilmente los artefactos genrados en la compression. Hay de distintos tamanos para probar que funciona con imagenes no multiples de 8.

```
images          Imagenes PPM de prueba para JPEG
├── boat_frag32.ppm   Imagen 32x32 (muy pequena y multiplo de 8)
├── boat_frag.ppm             Fragmento 8x8 (un solo bloque)
├── boat.ppm      Imagen tamano estandard
├── church.ppm      Imagen enorme (46 MB) tarda ~150 segundos en comprimir/descomprimir (82% de compressión con calidad 90)
├── cliff.ppm     Imagen tamano estandard (tarda mucho en comprimir)
├── gonza.ppm       Imagen pequena
├── gradient_35.ppm   Imagen 35x35 (patron gradiente muy pequeno y no multiplo de 8)
├── gradient8.ppm   Imagen 8x8 (un solo bloque con patron gradiente)
├── gradients32.ppm   Imagen 35x35 (muy pequena pero no multiplo de 8)
└── lake.ppm      Imagen enorme (46 MB) tarda ~150 segundos en comprimir/descomprimir
```