Informe Práctica 2: Programación Dinámica

Jorge Soria Romeo
872016

Alberto Francisco Solaz García
873373

February 2025

1 Introducción

En este informe analizamos la eficiencia y efectividad del algoritmo de reducción de costuras que hemos implementado en C++ utilizando programación dinámica y memoización. Su propósito es reducir la anchura o altura de una imagen minimizando la pérdida de información.

2 Eficiencia

3 Efectividad

Para probar la corrección y completitud de nuestros algoritmos (codificación estándar, codificación con profundidad limitada y decodificación) se ha usado una serie de ficheros de texto convencionales y ficheros binarios de prueba.

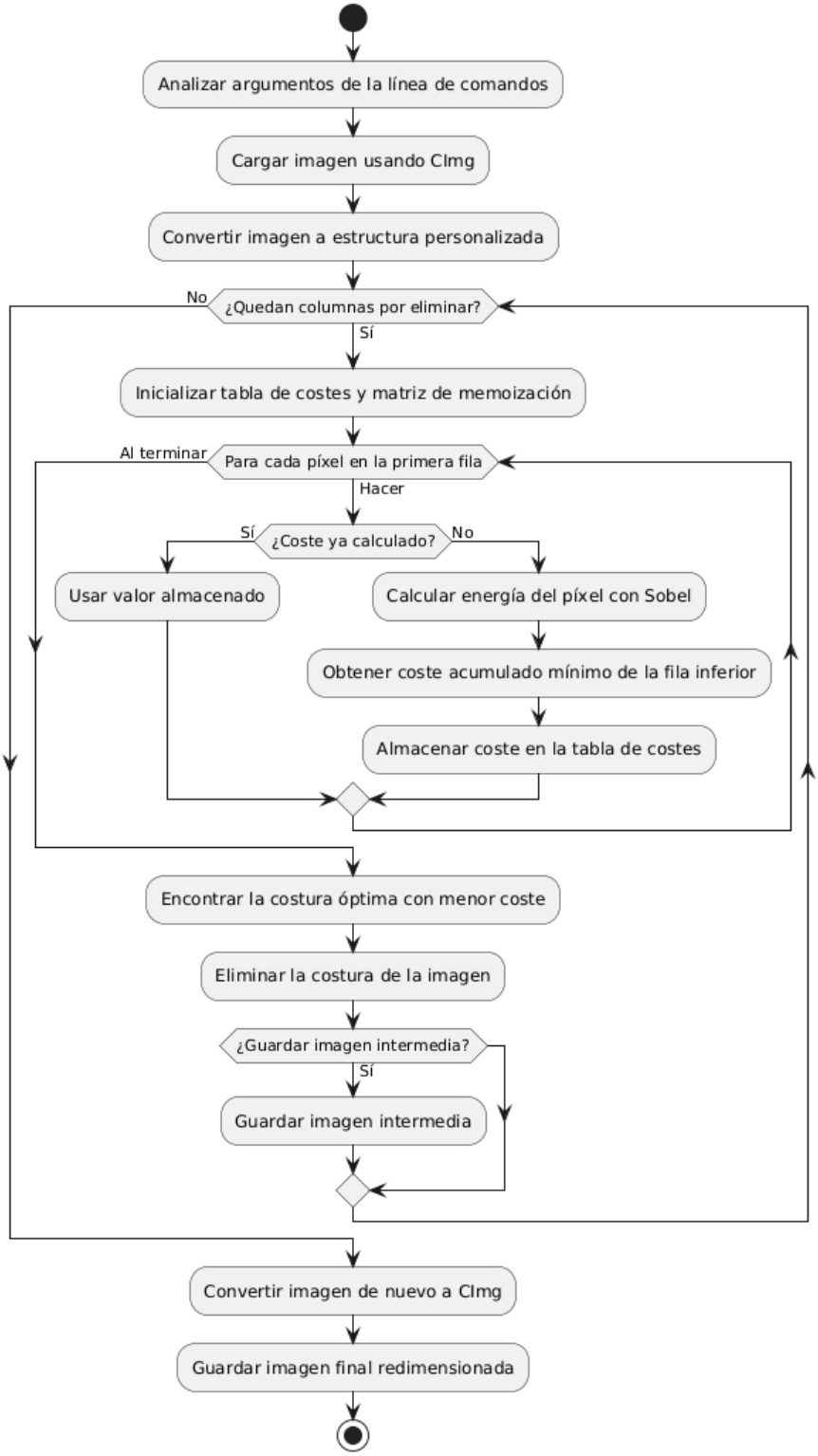


Figure 1: La visión del destino grabada en píxeles.

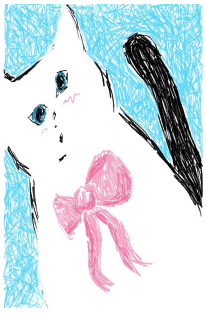
Se ha creado un *script* para *Bash* que compila los ficheros fuente y crea el ejecutable de nuestro programa. Posteriormente, realiza el siguiente proceso para cada fichero de prueba:

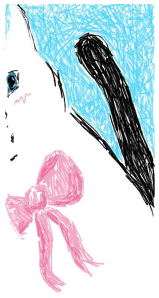
1. Obtiene el *hash* del fichero de prueba (concretamente *hash* SHA256).
2. Codifica (de manera estándar o limitada) el fichero de prueba.
3. Decodifica el fichero codificado y obtiene su *hash*.
4. Compara los *hashes*. Si coinciden el test ha resultado exitoso.

3.1 Resultados

- **test_corto01.txt** y **test_corto02.txt**: ficheros de prueba de extensión muy pequeña. En ambos no se disminuye el número de bytes porque la sobrecarga que implica codificar el árbol óptimo es más significativa que los bits que te ahorras.
- **test_largo01.txt** y **test_largo02.txt**: ficheros de extensión muy larga. El primer fichero es una traducción de la Biblia en inglés y el segundo el Quijote en español (incluye tildes, 'ñ', etc). En ambos se reduce su tamaño en bytes casí a la mitad (se reduce más de 1000000 bytes).
- **test_caracterEsFrecuente.txt**: fichero en el que uno o varios de sus caracteres tiene una frecuencia de aparición superior al resto de caracteres en el fichero.
- **test_caracterNoFrecuente.txt**: fichero en el que uno de sus caracteres tiene muy poca frecuencia de aparición repecto al resto de caracteres en el fichero.
- **test_numAparicionesParecidas.txt**: fichero en el que todos los caracteres tienen la misma frecuencia de aparición.
- **test_binario01.bin**: fichero binario estándar.
- **test_binario02.exe**: fichero binario compilado de programa "Hola, mundo" escrito en C++.


intermediate_1_profile.jpg


intermediate_150_profile.jpg


intermediate_300_profile.jpg


intermediate_450_profile.jpg


intermediate_600_profile.jpg

minus749_profile.jpg

Figure 2: Perfiles de diferentes etapas intermedias y el perfil mínimo. Los archivos muestran la evolución de la codificación con diferentes restricciones de longitud. Todas las imágenes tienen la misma altura para facilitar la comparación visual.

3.2 Coste temporal experimental

Además hemos probado la codificación limitada para ficheros como test_caracterEsFrecuente, test_caracterNoFrecuente o test_numAparicionesParecidas. La codificación limitada no siempre aumenta el tamaño de bytes que se reduce el fichero.

fichero	tamaño original (bytes)	tamaño comprimido (bytes)	decremento total (bytes)	tiempo para comprimir (ms)	tiempo para descomprimir (ms)
corto_01	29	66	-37	66.872	50.872
corto_02	51	64	-13	275.183	61.390
largo_01	2324344	1339313	985031	919.905	427.941
largo_02	2141519	1197821	943698	803.102	370.513
caracterEsFrecuente	100	46	54	36.797	68.630
caracterEsFrecuente (limitado)	100	48	52	56.323	57.646
caracterNoFrecuente	99	66	33	28.318	58.837
caracterNoFrecuente (limitado)	99	84	15	54.366	26.913
numAparicionesParecidas	95	50	45	29.763	53.324
numAparicionesParecidas (limitado)	95	55	40	58.527	56.252
binario01 (.bin)	100	384	-284	35.891	51.271
binario02 (.exe)	16392	5000	11392	41.206	59.713

Figure 3: Resultados