UNIVERSITAT POLITÈCNICA
DE CATALUNYA
BARCELONATECH

Práctica CDI 2023-2024

Compresión de secuencias de ADN

Fecha límite: 31/05/2024

hecho por

Hugo Aranda Sánchez
Javier Villarreal Arias
Albert Ruíz Víves

corregido por

Profesor: Fernando Martínez Sáez

Facultat d'Informática de Barcelona
Universitat Politècnica de Catalunya - BarcelonaTech

Contents

1 Introducción

En esta documentación se explicará y discutirá la solución obtenida para la práctica de final de la asignatura optativa de Compresión de Datos e Imágenes. Tras realizar todos los laboratorios de la asignatura en esta afrontamos un tipo de compresión sin pérdidas (o *lossless*) el cuál ha de mantener la totalidad de los datos a través de las transformaciones y ser recuperable en su totalidad.

El conjunto de datos usado cómo punto de referencia que usamos es un estándar realizado por los investigadores Diogo Pratas y Armando J. Pinho en el paper *A DNA Sequence Corpus for Compression Benchmark*[1]. Este representa fielmente los dominios presentes en esta clase de problema sin ser exageradamente grande de manera que supone un buen *benchmark* para comparar diferentes algoritmos.

2 Solución

Para la solución hemos tenido en cuenta los algoritmos presentados por la asignatura como punto de partida pero hemos realizado una investigación extensiva de las posibles soluciones que el problema tiene para ofrecer. Estas parten principalmente de una disertación de máster de Woramanot Yomjinda[2] la cuál contiene un extensivo estudio de las posibles soluciones a este mismo problema. En este documento se utilizan varios corpus para comparar y el que usamos en el proyecto convenientemente esta mencionado.

2.1 Soluciones barajadas

Cómo posibles soluciones vimos diversas las cuales incluyen algoritmos trabajados durante el curso con alguna permutación y otros más sofisticados que se salían de nuestra capacidad de trabajo.

Las opciones son las siguientes:

- **Huffman**: La codificación Huffman por su simpleza y potencial para alcanzar el ratio de compresión de cuatro con relativa facilidad fue una de las primeras opciones que barajamos. Pues al tener tan pocos símbolos la codificación palabra-código resultante tenía muy pocos bits y permitía una gran compresión con un simple cambio de diccionarios. Este es relativamente rápido y nos brinda buen rendimiento pero en nuestro caso le costaba pasar del 4 en gran medida pues no explora aspectos de localidad espacial y repetición dentro del ADN (Este presenta muchas estructuras reincidentes).

- **LZ77**: Las técnicas de diccionario prometian mucho por explotar en gran medida estos patrones presentes dentro del genoma, al utilizar las versiones implementadas en clase nos dimos cuenta de que estas requerían de algo de sofisticación y otros métodos encontrados mezclaban este con Context Tree Weighting[3] y requería una iteración bastante grande sobre el algoritmo base para hacerlo funcionar a ratios deseables.

- **PPM**: Este algoritmo a pesar de no ser uno de los trabajados durante el curso se presentaba muy prometedor y se realizó una implementación completa del mismo pero en última instancia el algoritmo que brindaba mayor ratio de compresión en menos tiempo fue otro. Este algoritmo resulta interesante por utilizar el modelo Markov, que trabaja con las probabilidades de los símbolos dependiendo del contexto de los que le preceden. Siendo este un modelo adaptativo que mantiene un conjunto actualizado de conteo de símbolos que tiene en cuenta el contexto para computar el siguente símbolo[2].

2.2 Solución encontrada y problemas afrontados

En última instancia el método que mejor tiempos y compresión nos brindó fue el *Integer Arithmetic Encoding*.

Este esta muy discutido en la publicación de Woramanot Yomjinda. En la que se discute que el desacoplamiento de la predicción probabilística y la codificación proporciona una flexibilidad de modelado sustancialmente mayor que los algoritmos que se basan en transformaciones directas de las entradas.[2]. Adicionalmente, es bien sabido que este algoritmo es capaz de obtener un alto rendimiento cuando se trata de textos base con alfabetos pequeños[4]. Esto lo hace una elección fantástica además del hecho de ser este un algoritmo trabajado dentro de los contenidos de la asignatura.

El principal reto de esta implementación fue la velocidad, pues la implementación de *Python* no era suficientemente rápida cómo para respetar las condiciones dadas por el enunciado de tasa de compresión/descompresión mayor a 250kB/s. Por este motivo se optó por implementar en *C++* el código de cara a obtener una sustanciosa mejoría de eficiencia por las propiedades del lenguaje más cercano a la máquina.

3 Código

El código del *Integer Arithmetic Encoding* es una traducción a *C++* del código que entregamos en la práctica 3. Este esta basado en el pseudocódigo del punto *4.4.3 Integer Implementation* y *4.4.4 Example* del libro de *Introduction to Data Compression*[4].

Este por tanto parte del código entregada de la práctica y busca mejorar la eficiencia al máximo para obtener los ratios de compresión a una velocidad razonable. La mejoría obtenida es sustancial respecto a *Python*. Haciendo que en pocos segundos se ejecuten partes que en *Python* se pasaban del tiempo marcado por la velocidad de compresión marcada. Para conseguir esto se trabaja con operaciones bit a bit para poder maximizar la eficiencia de todas las operaciones.

Cómo principal reto de programación tras todo el cuatrimestre programando en *Python* fue interesante recuperar un lenguaje más consciente de la arquitectura cómo *C++* que nos permite controlar mejor la reserva de memoria y las referencias para evitar al máximo las acciones innecesarias y encontrar una eficiencia mejorada.

4 Formato archivo

El formato del archivo contiene la codificación de diferentes elementos fijos los cuales sirven para mantener la información necesaria para decodificar el *Código Aritmético* y a continuación el propio código.

Por lo tanto contiene en este orden:

- *Tamaño original del mensaje:* codificado con 32 bits que se corresponden a un *int long*.
- *Tabla de frecuencias:* codificada con $32 \cdot 4 = 256$ bits que se corresponden a las cuatro entradas del diccionario codificadas como cuatro *int long*. Estas son 'A', 'C', 'G' y 'T' respectivamente.
- *Mesaje códificado:* Este contiene el mensaje codificado con *Arithmetic Encoding*

5 Resultados experimentales

Fichero	Tamaño	Compresión			Descompresión
	Original (kbytes)	Ratio Compresión	Tamaño (kbytes)	tiempo	tiempo
AeCa	1.591	4.02595	395	0m0,120s	0m0,137s
AgPh	43,9	4.02826	10,9	0m0,012s	0m0,011s
AnCa	142.189	4.06543	34.975	0m9,364s	0m8,379s
BuEb	18,9	4.02925	4,7	0m0,009s	0m0,058s
DaRe	62.565	4.10293	15.248	0m4,163s	0m3,509s
DrMe	32.181	4.01804	8.009	0m2,182s	0m1,925s
EnIn	26.403	4.10125	6.437	0m1,737s	0m1,505s
EsCo	4.641	4.00028	11.603	0m0,337s	0m0,292s
GaGa	148.532	4.06012	36.583	0m9,750s	0m9,139s
HaHi	3.890	4.09246	950	0m0,302s	0m0,255s
HePy	1.667	4.07324	409	0m0,157s	0m0,115s
HoSa	189.752	4.08217	46.483	0m12,380s	0m11,072s
OrSa	43.262	4.0226	10.754	0m2,908s	0m2,518s
PlFa	8.986	4.26945	2.104	0m0,645s	0m0,575s
ScPo	10.652	4.07419	2.614	0m0,742s	0m0,624s
WaMe	9.144	4.04752	2.259	0m0,721s	0m0,545s
YeMi	73,6	4.25757	17	0m0,041s	0m0,035s

Table 1: Resultados de la experimentación con cada uno de los archivos (el tiempo ha sido tomado con la instrucción de linux time y se obtiene el tiempo real). Los kbytes se refieren a 1000 bytes no a KB, tomamos la medida así

5.1 Tramo de nota obtenido

Según los resultados obtenidos obtenemos una media final de *4.0795*

References

[1] Diogo Pratas and Armando J. Pinho. A dna sequence corpus for compression benchmark. In Florentino Fdez-Riverola, Mohd Saberi Mohamad, Miguel Rocha, Juan F. De Paz, and Pascual González, editors, *Practical Applications of Computational Biology and Bioinformatics, 12th International Conference*, pages 208–215, Cham, 2019. Springer International Publishing.

[2] Woramanot Yomjinda. Lossless dna compression. Master's dissertation, University of Cambridge, Queens' College, August 2020. Supervisor: Christian Steinruecken.

[3] Shengwang Du, Junyi Li, and Naizheng Bian. A compression method for dna. *PLOS ONE*, 15(11):e0238220, Nov 2020.

[4] K. Sayood. *Introduction to Data Compression*. The Morgan Kaufmann Series in Multimedia Information and Systems. Elsevier Science, 2012.