Compresión de datos sin pérdida
Tarea 1

Año 2021

1. Objetivos

Esta tarea práctica tiene los siguientes objetivos específicos:

- Consolidar a través de la práctica los conocimientos teóricos vistos en las primeras semanas de curso.
- Ensayar técnicas de programación apropiadas para la implementación de algoritmos de compresión.
- Experimentar y evaluar el desempeño de algoritmos de compresión con datos reales.
- Desarrollar la capacidad de análisis y comunicación escrita de resultados experimentales.
- Motivar temas que veremos en profundidad en clases futuras.

Observar que la implementación concreta de un algoritmo de compresión es un medio para llegar a estos objetivos pero no un objetivo en sí mismo. Los objetivos se alcanzan a través del trabajo que implica el desarrollo de la tarea y, por esta razón, absolutamente toda la codificación y el contenido de los informes debe ser desarrollado individualmente por cada estudiante.

2. Descripción de la tarea

2.1. Datos para evaluación experimental

El trabajo experimental se realizará sobre archivos de secuenciación de ADN, los cuales contiene información relativa a un conjunto de fragmentos de una secuencia de ADN obtenidos mediante un dispositivo de secuenciación. Cada fragmento de ADN está formado por una serie de *nucleótidos* encadenados entre sí, que usualmente se representa mediante una secuencia de letras del conjunto $\{A, C, G, T\}$, en referencia a las cuatro bases nitrogenadas diferentes (adenina, citosina, guanina, timina) que dan lugar a cuatro tipos de nucleótidos que conforman un ADN.

Trabajaremos con archivos en formato FASTQ. Este es un formato de archivo de texto en el cual cada lectura de un fragmento de ADN (se denomina *read*) se representa mediante cuatro líneas de texto:

1. La primera línea de un read comienza con el símbolo '@' seguido de un texto que identifica al read.
2. La segunda línea contiene la secuencia de símbolos en el alfabeto $\{A, C, G, T\}$ que representa al fragmento de ADN. Pueden aparecer otros símbolos para representar situaciones particulares, típicamente la letra 'N' se utiliza para representar un nucleótido que no pudo ser identificado correctamente durante el proceso de secuenciación.
3. La tercera línea comienza con el símbolo '+' seguido de un texto descriptivo (eventualmente vacío).
4. La cuarta línea tiene el mismo largo que la línea 2 y está formada con el alfabeto de 94 símbolos cuyos códigos ASCII van de 33 a 126. El i-ésimo símbolo de la línea 4 es un indicador de calidad de la medida que dio lugar al i-ésimo nucleótido en la línea 2 (una función de la probabilidad de error en esa lectura).

Una posible fuente de datos para prueba es el sitio del centro estadounidense para la información en biotecnología (National Center for Biotechnology Information) https://www.ncbi.nlm.nih.gov/. Para acceder a los datos de secuenciación disponibles en esta base de datos se utiliza un programa utilitario que está disponible en https://trace.ncbi.nlm.nih.gov/Traces/sra/sra.cgi?view=software. Para descargar un archivo FASTQ se utiliza el comando

```
fastq-dump -I SRR_ID
```

donde SRR_ID es un identificador del archivo que se quiere descargar. Algunos ejemplos de SRR_ID son ERR161541, ERR236069, SRR254209. La descripción de los datos está disponible en

https://www.ncbi.nlm.nih.gov/sra/?term=SRR_ID

reemplazando SRR_ID por el código correspondiente en cada caso.

2.2. Implementación de un codificador y decodificador LZ77 básico

(I) Implementar el algoritmo de codificación y decodificación LZ77 con las siguientes consideraciones.

- Tanto el codificador como el decodificador reciben como entrada un archivo y un parámetro, N, que define tanto el tamaño de la ventana de diccionario como el valor máximo del largo de una frase (si se encuentra una coincidencia de largo mayor que N la frase correspondiente se trunca a largo N).
- Los símbolos literales se codifican en base a una distribución uniforme. Por defecto se trabaja con un alfabeto de bytes (256 elementos), pero es conveniente contemplar desde el inicio la posibilidad de usar un subconjunto del alfabeto.
- El diseño de un código para el largo y desplazamiento de cada frase queda a criterio de cada estudiante.
- La definición de un esquema de codificación para los primeros N símbolos queda a criterio de cada estudiante.

(II) Experimentar con diferentes archivos, variando el parámetro N. Estudiar cómo varía la tasa de compresión a medida que avanza la codificación de una secuencia x^n (un archivo). Para esto se sugiere generar como salida auxiliar una secuencia de pares (n_i, L_i), $i = 1, 2, 3, \ldots, m$, donde m puede ser diferente para cada archivo, tal que $n_1 \leq n_2 \leq \ldots \leq n_m = n$ y L_i es el largo de código generado hasta el momento de haber procesado el prefijo x^{n_i}.[1] Analizar los resultados.

2.3. Implementación de una variante para FASTQ

Teniendo en cuenta que diferentes tipos de líneas de un archivo en formato FASTQ difieren notablemente entre sí en relación al alfabeto que utilizan y sus propiedades estadísticas, se propone dividir la información contenida en el archivo en segmentos y codificar cada segmento por separado. Estos segmentos se definen de la siguiente manera:

1. Un segmento que contiene la concatenación de todas las líneas de tipo 2. Esto define una secuencia sobre el alfabeto $\{A, C, G, T, N\}$.[2]

2. Un segmento que contiene la concatenación de todas las líneas de tipo 4. Esto define una secuencia sobre el alfabeto de indicadores de calidad que consta de 94 símbolos.

3. Un segmento que contiene la secuencia de largos de los reads del archivo. Esta información permite separar las secuencias de símbolos de los segmentos anteriores en reads individuales.

4. Un segmento con la concatenación de las líneas de tipo 1 y 3 (incluyendo los caracteres de fin de línea).

[1] Por ejemplo, $\{n_i\}$ podría ser un subconjunto de las posiciones donde finalizan frases de LZ77, de forma tal que la separación entre n_i y n_{i+1} sea mayor que un umbral fijado de antemano para todo i, $1 \leq i < m - 1$.

[2] Conviene verificar que los archivos de prueba efectivamente contengan exclusivamente símbolos de este conjunto.

(I) Implementar un codificador y decodificador, con los mismos parámetros que en la sección 2.2, que comprima cada segmento por separado, teniendo en cuenta el alfabeto de cada uno. Para los segmentos 1, 2 y 4 usar una codificación LZ77 basada en la implementación pedida en la sección 2.2. Para el segmento 3 definir un esquema de codificación que considere apropiado; no necesariamente tiene que estar basado en LZ77 y, si resulta útil, puede asumir una cota superior para los largos de los reads. El decodificador debe recomponer el archivo en el formato FASTQ original.

(II) Repetir los experimentos de la sección 2.2 para este nuevo codificador, estudiando tanto el desempeño de compresión global como el de cada segmento por separado. Analizar los resultados.

3. Consideraciones generales sobre las entregas

- Esta tarea es parte de la evaluación del curso y se resuelve individualmente.
- Los lenguajes de programación que se aceptan son C y C++.
- Las decisiones de diseño e implementación que influyen sobre el rendimiento de los programas son importantes y serán evaluadas.
- Las entregas se aceptan en la página del curso hasta el 24 de mayo inclusive. Debe incluirse:
  - Código fuente acompañado de un makefile, sript de compilación o similar.
  - Referencias en línea a los datos utilizados en los experimentos.
  - Instrucciones de cómo compilar y ejecutar los experimentos.
  - Informe donde se describa la solución y se analicen los resultados de los experimentos.