Proyecto

Análisis y Diseño de Algoritmos (ADA)

14 de octubre de 2021

1. Introducción

Este es un proyecto grupal para el curso de Análisis y Diseño de Algoritmos, sirve como nota dentro del rubro "proyectos" (20 % de la nota final del curso). El grupo es de máximo tres personas.

2. Problema de codificación de texto

Dado un conjunto C de $2^k - 3$ caracteres, una *tabla de códigos* para C es una biyección entre C y las primeras secuencias de k bits. Por ejemplo, si $C = \{z, i, A, m, j\}$, una posible tabla para C es la función f definida según $f(z) = 000$,$f(k) = 001$, $f(A) = 010$ y $f(m) = 011$.

Dada una cadena s sobre el conjunto de caracteres C, y una tabla f de C, una f-codificación de s es la cadena de bits resultante de aplicar la biyección f a todos los caracteres de s y concatenarlos. Por ejemplo, tomando el mismo C del párrafo anterior, si $s = zzkmA$, la f-codificación de s es la secuencia de bits 000000001011010.

Sean C_1 y C_2 dos conjuntos de caracteres, no necesariamente disjuntos, con tablas de códigos f_1 y f_2, y tamaños $2^{k_1} - 3$ y $2^{k_2} - 3$, respectivamente. Sea s una cadena sobre $C_1 \cup C_2$. Una (f_1, f_2)-codificación de s es una cadena de bits resultante de aplicar alguna de las biyecciones f_1 o f_2 a todos los caracteres de s y concatenarlos. Mas aún, entre caracteres que han sido codificados existen secuencias de bits especiales que indican que la codificación ha cambiado de f_1 a f_2, o viceversa. Por ejemplo, si $C_1 = \{a, b, c, d, e\}$, $C_2 = \{d, e, 0, 1, 2\}$ son dos conjuntos de caracteres con f_1 y f_2 definidos por

$$f_1(x) = \begin{cases} 000 & \text{si } x = a \\ 001 & \text{si } x = b \\ 010 & \text{si } x = c \\ 011 & \text{si } x = d \\ 100 & \text{si } x = e \end{cases}$$

y

$$f_2(x) = \begin{cases} 000 & \text{si } x = d \\ 001 & \text{si } x = e \\ 010 & \text{si } x = 0 \\ 011 & \text{si } x = 1 \\ 100 & \text{si } x = 2 \end{cases}$$

Entonces la cadena $cde1$ puede ser codificada por

$$010111000111100111011,$$

siendo que se comienza codificando con f_1 y la secuencia 111 tanto de f_1 como f_2 sirve para cambiar de función de codificación. Otro ejemplo de codificación válida es

$$010011111001011.$$

Podemos generalizar dicha definición a cuatro conjuntos C_1, C_2, C_3, C_4 de caracteres y tablas f_1, f_2, f_3, f_4 asociadas. Siendo que las tres últimas secuencias de bits de cada conjunto sirven para cambiar de codificación hacia las siguientes secuencias posibles. Por ejemplo, si C_1 tiene tamaño $2^3 - 3$, entonces las secuencias desde 000 hasta 100 sirven para codificar los elementos de C_1; y las secuencias 101,110, y 111 sirven para cambiar de codificación hacia C_2, C_3 y C_4, respectivamente. De manera similar, al estar en C_2, si por ejemplo C_2 tiene tamaño $2^4 - 3$, las secuencias 1101,1110, y 1111 sirven para cambiar de codificación hacia C_3, C_4 y C_1, respectivamente.

A continuación definiremos nuestros conjuntos de trabajo. Nuestro conjunto de caracteres a codificar serán aquellos imprimibles, es decir, desde el codigo ASCII 32 (que representa un espacio en blanco) hasta el código 126 (que representa la negación). Tenemos entonces 95 caracteres, los cuales estarán distribuidos en 4 grupos (no necesariamente disjuntos).

- C_1: guarda las **vocales en minúscula** (a, e, i, o, u). Note que $|C_1| = 5$.
- C_2: guarda las **vocales en minúscula y mayúscula**, los **números del 0 al 9**, y los caracteres **ASCII desde el 32 (espacio) hasta el 40 (paréntesis izquierdo)**. Note que $|C_2| = 29$.
- C_3: Guarda las **letras a–z y A–Z**, y los caracteres **ASCII desde 91 (corchete izquierdo) al 95**, más los caracteres **ASCII desde el 123 (bracket izquierdo) al 126 (negación)**. Note que $|C_3| = 61$
- C_4: Caracteres **ASCII desde el 32 al 126**. Note que $|C_4| = 96$. Este caso es una excepción, pues existen algunas secuencias de bits que no se usarán.

Estos conjuntos son fijos y no cambiaran en las preguntas pedidas. Antes de comenzar a codificar, los dos primeros bits indicaran con cual de las cuatro codificaciones empezaremos. Podemos entonces definir nuestro problema de investigación.

Problema Min-Cod. Dada una cadena s sobre el conjunto C, encontrar una codificación de C óptima.

Por ejemplo, si $s = a0e$, una codificación podría comenzar en C_1 y hacer un único cambio a C_2, lo que resultaría en el código 00000101010101000001. Otra codificación para la misma cadena podría comenzar en C_1, cambiar a C_2, y volver nuevamente a C_1: 00000101010101111001.

Calentamiento: haga todas las posibles codificaciones para s (no tiene nota).

Pregunta 1 (Heurística Voraz). Analice, diseñe e implemente una heurística voraz con complejidad polinomial para el problema Min-Cod. Su algoritmo **no** deberá encontrar necesariamente una respuesta óptima. Su heurística deberá tomar una decisión local según algún criterio, aunque en el largo plazo dicha decisión no sea óptima.

Entrada del algoritmo: Una cadena s sobre el conjunto C

Salida del algoritmo: Una cadena de bits que codifica s, y su tamaño.

Tiempo de ejecución del algoritmo: polinomial

El algoritmo anterior, aunque es rápido, no resuelve el problema de manera exacta. El siguiente paso es diseñar un algoritmo recursivo. Para ello encontraremos un subproblema al problema original. Sea $s = a_1 a_2 \cdots a_n$. Sea $OPT(i, j)$ el tamaño de una codificación óptima para $s' = a_1 a_2 \cdots a_i$ que codifica al caracter a_i en el modo C_j.

Pregunta 2 (Recurrencia). Plantee una recurrencia para $OPT(i, j)$.

Basada en la recurrencia hallada en la Pregunta 2, diseñaremos tres algoritmos.

Pregunta 3 (Recursivo). Analice, diseñe e implemente un algoritmo recursivo con complejidad exponencial para el problema Min-Cod. Su algoritmo deberá encontrar una solución óptima.

Entrada del algoritmo: Una cadena s sobre el conjunto C

Salida del algoritmo: Una cadena de bits que codifica s de manera óptima, y su tamaño.

Tiempo de ejecución del algoritmo: exponencial.

Pregunta 4 (Memoizado). Analice, diseñe e implemente un algoritmo memoizado para el problema Min-Cod. Su algoritmo deberá encontrar una solución óptima.

Entrada del algoritmo: Una cadena s sobre el conjunto C

Salida del algoritmo: Una cadena de bits que codifica s de manera óptima, y su tamaño.

Tiempo de ejecución del algoritmo: polinomial.

Pregunta 5 (Programación Dinámica). Analice, diseñe e implemente un algoritmo de programación dinámica para el problema Min-Cod. Su algoritmo deberá encontrar una solución óptima.

Entrada del algoritmo: Una cadena s sobre el conjunto C

Salida del algoritmo: Una cadena de bits que codifica s de manera óptima, y su tamaño.

Tiempo de ejecución del algoritmo: polinomial.

3. Software de codificación y decodificación

Las siguientes preguntas tienen como objetivo organizar las implementaciones anteriores a modo de un minisoftware. También es acá donde implementaremos la decodificación. Estas preguntas son más concretas encuanto las entradas son archivos de texto.

Pregunta 6 (Codificación heurística).
Entrada del algoritmo: Un archivo de texto con caracteres imprimibles.
Salida del algoritmo: Un archivo codificado usando el algoritmo implementado en la Pregunta 1 (heurística voraz).

Pregunta 7 (Codificación óptima).
Entrada del algoritmo: Un archivo de texto con caracteres imprimibles.
Salida del algoritmo: Un archivo codificado usando el algoritmo implementado en la Pregunta 5 (programación dinámica).

Pregunta 8 (Decodificación).
Entrada del algoritmo: Un archivo codificado usando alguna de las preguntas anteriores.
Salida del algoritmo: La cadena original.

Pregunta 9 (Software de codificación y decodificación). En esta pregunta deberá implementar un pequeño software usando las preguntas 6, 7 y 8.

Este software recibirá archivos de texto. El usuario tendrá dos opciones. La primera usará la codificación heurística y la segunda la óptima. En este paso es donde se usan las preguntas 6 y 7, respectivamente. Al elegir la opción de codificación, se generará un archivo binario con el código correspondiente

Al elegir la opción de decodificación, recibirá un archivo binario, el cual deberá decodificar.

Pregunta 10 (Análisis experimental). Realize un análisis experimental entre las dos opciones de codificación. Deberá verse la diferencia en memoria entre una u otra opción. Su análisis de experimental deberá incluir casos de prueba aleatorios e ir incrementando los tamaños de las muestras para notar una diferencia asintótica (en caso exista).

4. Indicaciones finales

Fechas de entrega

Habrán dos fechas de entrega.

- Entrega parcial: Semana 11 (11 y 12 de Noviembre). Preguntas 1–2.
- Entrega final: Semana 16 (16 y 17 de diciembre). Preguntas 3–10.

Entregables

Se deberá entregar una monografía hecha en LaTeX describiendo el análisis diseño e implementación de cada algoritmo pedido. El análisis del algoritmo debe incluir análisis de tiempo de ejecución (si es el caso plantear una recurrencia y resolverla o mostrar por inducción). También deberá incluir todas las demostraciones pertinentes respecto al uso de programación dinámica. El diseño del algoritmo deberá incluir el pseudocódigo del algoritmo. La implementación del algoritmo podrá ser hecha en C,C++,Python o Java. Se deberá compartir el código de las implementaciones en algún repositorio Git.

5. Puntajes

Tendremos dos fechas de entrega

1. Monografía 10 %
2. Primera entrega 15 %
2. Segunda entrega 75 %

No habrá entregas fuera de las fechas, ni reconsideraciones.

Referencias

[1] Cormen, T., Leiserson, C., Rivest, R., and Stein, C. Introduction to Algorithms, Third Edition. *The MIT Press.*, 2009.