Propuesta Tema de Título

Optimal Prefix Free Codes Adaptivos

08/10/15

david@fml.cl

Fono: 95534997

Profesor Guía: Jeremy Barbay

Alumno: David Muñoz

_____ _____

Motivación

Un problema bastante conocido en el área de la computación, es el del cálculo de los "optimal prefix free codes" (en adelante OPFC). Este problema consiste en encontrar para cada elemento perteneciente a un conjunto O de símbolos, una codificación optima de tal forma que ningún código sea prefijo de otro.

Esta codificación es óptima en el sentido que se busca optimizar el espacio usado por cada símbolo según algún parámetro conocido: en particular se suele considerar la frecuencia de aparición de cada símbolo dentro de un texto, pero a modo general se asume que el se recibe como parámetro una lista con los pesos de cada elemento.

Este problema ha sido ampliamente estudiado a lo largo de la historia y hay varios resultados interesantes, siendo su implementación más conocida (y enseñada) el algoritmo de Huffman. Este algoritmo es tan conocido que los OPFC son ampliamente llamados "Códigos de Huffman".

Usando el algoritmo de Huffman para determinar los OPFC de un set de n símbolos dada su lista de frecuencias y usando una cola de prioridad, es $\theta(nlogn)$ en el modelo de comparaciones. Es decir, en el peor caso eso es lo que va a tomar calcular un OPFC con una entrada de tamaño n.

Sin embargo, Van Leeuwen demostró que el cálculo de la codificación de Huffman podía llevarse a cabo en tiempo O(n) si la lista de frecuencias era entregada inicialmente de forma ordenada.

A partir de esto surgen dos interrogantes, que son las que se pretenden contestar durante la realización del trabajo:

1. ¿Es posible computar los OPFC en tiempo mejor que $\theta(nlogn)$ al sacar ventaja de ciertos "casos fáciles" específicos?
2. ¿Es posible calcular los OPFC sin errores realizando solamente un ordenamiento parcial de la lista de frecuencias?

Discusión

Formalizando un poco más, el problema de los OPFC puede ser enunciado de la siguiente manera:

"Sea T un texto, conformado por n símbolos pertenecientes a un lenguaje L. Se sabe que cada palabra p perteneciente a L ocupa una cantidad k=log(|L|) de bits, implicando que el texto tiene tamaño |T| = nk bits.

Dada la lista de frecuencias de aparición de cada palabra de L en T, se pide encontrar una nueva codificación para cada palabra de este lenguaje de forma que se genere un nuevo texto T' a partir de T tal que |T'| < nk bits. Se requiere además, que esta codificación sea óptima."

La codificación de Huffman resuelve exactamente este problema en, como ya se dijo, $\theta(nlogn)$ para el peor caso con entradas de tamaño n. Pero las preguntas propuestas apuntan a que se puede lograr algo mejor, si consideramos que existe varios casos que son fáciles de resolver, y que estos son fácilmente detectables.

Por ejemplo, al recibir la lista de pesos, podríamos tratar de determinar lo siguiente:

1. Si la lista de frecuencias viene ordenada: revisar que una lista viene ordenada toma tiempo lineal, y luego solo hay que aplicar Van Leeuwen en vez del algoritmo clásico.
2. Si la frecuencia de todas las palabras es la misma: revisar que todos los elementos de una lista son los mismos toma tiempo lineal, y construir los OPFC dado eso también toma tiempo lineal.
3. Si los elementos de la lista están en el rango [x….2x[: Determinar el menor y el mayor elemento toma tiempo lineal, y resolver este caso también toma tiempo lineal.

También algo a lo que se le puede sacar ventaja es saber que existen algoritmos adaptivos de ordenamiento. Estos algoritmos pueden ordenar en tiempo menor a la cota $\theta(nlogn)$ aprovechándose del hecho que fragmentos de un arreglo pueden venir parcialmente ordenados. Entonces si se ordena de manera adaptiva y después se hace el cálculo de los OPFC se puede ver intuitivamente que se puede hacer algo mejor.

Además, hasta el momento se ha asumido que es necesario ordenar completamente la lista de frecuencias para poder hacer el cálculo de los OPFC. Sin embargo existe la intuición de que esto puede ser hecho directamente sobre un arreglo parcialmente ordenado.

La intuición de por sí es correcta, por ejemplo si se tiene la siguiente lista de frecuencias:

2	1	4	5	8	7	11	10

Sobre esta lista da lo mismo si se ordena o no, ya que Van Leeuwen extrae inicialmente los dos elementos iniciales de la lista (asumiendo que son los mínimos), cosa que en este caso se cumple a pesar de que no están ordenados.

Este caso de por sí es muy simple (y muy conveniente), pero generalizar esto no es trivial, y más aún, no se está mostrando que se puede calcular los OPFC sobre un arreglo no desordenado, solo se está mostrando que aplicar el algoritmo de Van Leeuwen sobre esta entrada en específico no tiene efecto en el resultado final. Pero en realidad en ningún caso se debería aplicar Van Leeuwen sobre un arreglo desordenado.

Esto de hecho, refina la pregunta planteada anteriormente y podemos reformularla como:

1. Sabiendo que es posible calcular los OPFC sobre arreglos parcialmente ordenados. ¿Es posible modificar el algoritmo de Van Leeuwen para que funcione sobre estos arreglos?
2. ¿Es posible definir algún algoritmo adaptivo que permita calcular de forma óptima los OPFC?

Objetivos

General: Identificar una medida de dificultad y definir un algoritmo adaptivo para el cálculo de los Optimal Prefix Free Codes.

Específicos:

1. Hacer un estudio teórico más profundo acerca de los OPFC.
2. Definir un algoritmo adaptivo para el cálculo de OPFC y analizarlo.
3. Implementar el algoritmo definido.
4. Testear el algoritmo de forma de obtener datos empíricos y hacer que el experimento sea repetible.

Referencias

Todavía pendiente, lo agregaré hoy (martes 13).