Raimundo Herrera Sufán - 14632152

Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencia de la Computación

IIC2133 — Estructuras de Datos y Algoritmos — 1' 2017

Tarea 2

Análisis de Problema

Modelación

Para todo el problema se trató con las melodías como si fueran strings. Así, el problema de encontrar el alma de la melodía, lo modelé como el desafío de encontrar el *substring* de largo mayor o igual a μ que más se repitiera en un *string* dado, en este caso, la melodía.

Para esto, utilicé la estructura de datos llamada $suffix\ trie$, tras muchos intentos fallidos con el $suffix\ array$. La ventaja del escogido es que es muy fácil de recorrer, y es posible, a medida que se va armando la estructura, ir llenando los nodos del árbol con datos valiosos para la solución. Los nodos representan cada uno a una nota, y su ubicación da luces del $sufijo$ al que pertenece.

Creo que es una buena forma de modelar el problema porque en los audios se demora un tiempo muy razonable en realizarlos, esto hablando empíricamente, pero teóricamente también, porque no es necesario guardar todos los posibles *substrings* del *string* que representa a la melodía, sino que únicamente las formas de ir formándolos, valga la redundancia, con lo que se consigue armar el *trie* en un tiempo menor que el de todas las posibilidades. Por lo mismo la eficiencia en cuanto a espacio que utiliza en memoria, es mejor, al ahorrar elementos.

Modificaciones

Para la representación del problema, incurrí en modificaciones en construcción y en funciones. En primer lugar los nodos de mi *trie* corresponden a elementos con información útil para resolver el problema.
Al construir se iban llenándo estos atributos, de modo que luego para resolver el problema no hubiera que recorrer nuevamente el árbol. Se agregó la cantidad de veces que se visitó el nodo, y la profundidad del mismo, para saber cuantas veces aparecía un *substring* desde el *root* hasta ese nodo, y cuan largo, por profundidad, era ese *substring*, de modo que obtener el más repetido de largo superior a μ fuera inmediato. Además se agregó en cada nodo un puntero a su padre, para que al encontrar el nodo de la solución, este tuviese una forma directa de volver recorriendo su camino, para encontrar la submelodía entera, es decir el alma.

En cuanto a funcionalidades, se hizo una función principal encargada de crear el *trie*, donde se iban agregando todos esos datos adicionales, pero también se implementaron funciones aprovechando la estructura, que recorren desde un nodo a la raíz, o que imprimen la ruta recorrida, etc. Junto con eso, como cada nodo tenía *hijos*, se implementaron funciones para saber si un nodo *estaba en* los hijos de otro nodo, de modo de poder ir marcando los nodos como visitados pertinentemente.

En general las modificaciones de todo tipo realizadas, fueron en pos de reducir la cantidad de recorridos del *trie*, y facilitar la resolución del problema. Es claro que en orden de complejidad, no se aumenta la

misma al aumentar la cantidad de recorridos por la estructura, sin embargo es evidente que las constantes juegan un rol importante, y disminuirlas puede ser, como es el caso, una mejoría sustancial.