Tarea 7 - Programación y algoritmos
Giovanni Gamaliel López Padilla

Índice

1. Análisis de la complejidad de Quicksort

El algoritmo de Quicksort puede resumirse en lo siguiente:

Algorithm 1: Quicksort

```
   Input: data
   Output: data
1  if start<last then
2      number_data ← reduce_data(data,start,last)
3      sort(data,start,number_data)
4      sort(data,number_data+1,last)
5  Function reduce_data(F):
6      pivot ← data[last]
7      i ← start
8      for j=start,last-1 do
9          if data[j]>pivot then
10             swap(data[j],data[i])
11             i ← i+1
12     return i
```

Con este algoritmo podemos realizar dos análsis, esto es para el peor caso y mejor caso. En el algoritmo 1 se ve que es recursivo. En la linea 3 y 4 se puede ser que se da como argumento la posición inicial y final de los datos a ordenar. El algortimo acaba cuando las dos funciones tienen el mismo en valor en la posición inicial y final.

1.1. Peor caso

El peor caso posible es que el pivote escogido siempre sea el número más grande o pequeño del arreglo. Esto es porque provocaria que uno de los bloques sea de tamaño n y otro de 1. Esto sería:

$$\begin{aligned} T(n) &= T(n-1) + n \\ &= T(n-2) + (n-1) + n \\ &= T(n-3) + (n-2) + (n-1) + n \\ &\vdots \\ &= 1 + 2 + 3 + \cdots + n \\ &= \frac{n(n+1)}{2} \end{aligned}$$

Entonces, la complejidad para el peor caso es $O(n^2)$

1.2. Mejor caso

El mejor caso posible es que el pivote escogido sea el número sea la mediana del arreglo. Esto provocaria que los dos bloques tengan un tamaño de $\frac{n}{2}$. Entonces:

$$
\begin{aligned}
T(n) &= 2T\left(\frac{n}{2}\right) + n \\
&= 2\left(2T\left(\frac{n}{2}\right) + \frac{n}{2}\right) + n \\
&= 4T\left(\frac{n}{4}\right) + 2n \\
&= 2^k T\left(\frac{n}{2^k}\right) + kn \\
&= nT(1) + log(n)(n) \qquad \text{si } n = 2^k \\
&= n + nlog(n)
\end{aligned}
$$

Entonces, la complejidad para el mejor caso es $O(nlog(n))$.

2. Encriptado y desencriptado de un archivo wav

2.1. Organización de la cabecera

Los datos de un archivo wav se encuentran administrados como header y los dados de audio como se muestra en la tabla 1.

Sección	Tamaño (bytes)	Descripción
RIFF	4	Contiene las letras R, I, F y F
	4	Entero positivo que almacena el tamaño de los bytes restantes
WAVE	4	Contienen las letras: W, A, V y E.
	4	Contiene los caracteres F, M, T
	4	Entero positivo que indica el tamaño en bytes del resto del bloque
	2	Entero positivo de 16 bits que indica el tipo de grabación. Un 1 significa PCM.
	2	Número de canales
	4	Frecuencia de muestreo expresada en Hz
	4	Número promedio de bytes por segundo
	2	Número de bytes usados en el archivo para cada muestra
	2	Bits por muestra
data	4	Contiene las letras d, a, t, a
	4	Entero positivo que indica el espacio de bytes que ocupan los datos
	n	Datos

Tabla 1: Organización de los datos en un archivo de formato wav.

Esta organización hace que los primeros 44 bytes no sean modificados, ya que es información del archivo y no sus datos. Al ser los archivos de muestra de un audio mono, entonces estos tiene un tamaño de 16 bits por muestra, esto se puede ver en las figuras 1 y 2.

```
Información del archivo ftea_10k.wav
ChunkID         RIFF
ChunkSize       39060
Format          WAVE

Subchunk1ID     fmt
Subchunk1Size   16
AudioFormat     1
NumChannels     1
SampleRate      10000
ByteRate        20000
BlockAlign      2
BitsPerSample   16

blockID         data
blockSize       39024
```

Figura 1: Información del archivo ftea_10k.wav

```
Información del archivo mtea_10k.wav
ChunkID         RIFF
ChunkSize       53054
Format          WAVE

Subchunk1ID     fmt
Subchunk1Size   16
AudioFormat     1
NumChannels     1
SampleRate      10000
ByteRate        20000
BlockAlign      2
BitsPerSample   16

blockID         data
blockSize       53018
```

Figura 2: Información del archivo mtea_10k.wav

Si estos fueran un audio stereo, entocnes el número de bits por muestra seria 32, 16 para el audio izquiero y 16 para el derecho. Es por ello que se implemento un arreglo de dimension dinámica para guardar cada muestra del archivo.

2.2. Lectura y organización de los datos

La lectura de los datos de la cabecera serán guardados en tipos definidos. Estos tipos se encuentran en el archivo wave.h. Este proceso no es propio, se encontro un repositorio en GitHub, el cual realiza modificaciones al sonido y se implemento el uso de sus lecturas de los datos. Estos reposiorotios son los siguientes: Lectura de los datos, Estructura de los tipos.

El arreglo de datos es de tipo short para que estos coincidan con los 16 bits que tiene cada muestra de audio. Al tener las muestras de audio organizadas en arreglos es más sencillo realizar el paso por cada uno de estos datos.

2.3. Encriptación y desencriptación

2.3.1. Mapa caótico

En la teoría del caos se describen comportamientos de sistemas dinámicos los cuales se ven afectados por sus condiciones iniciales [1]. En este caso tendremos unas condiciones iniciales llamadas $X_{i,0}$, donde i, es el número de llaves. Al tener 2 bytes por muestra, dividiremos esto en datos de 1 byte. Entonces el número de llaves serán 2.

El mapa caótico que se uso esta definido en la ecuación 1.

$$X_{i,j} = bX_{i,j-1} + \left\lfloor \frac{X_{i,j-1}}{2^m} \right\rfloor + \epsilon \& \bigotimes_{k=1}^{3} X_{k,j-1} \tag{1}$$

donde $\epsilon, b, m \in \mathbb{N}$ y $3 < m < 8$.

2.3.2. Algoritmo

El algoritmo creado para aplicar el mapa caótico de la ecuación 1 es el siguiente:

Algorithm 2: Encriptado y desencriptado

Input: *data*, *keys*
Output: *data*

```
map ← keys
for i = 1,n_samples do
    bytes ← data[i]
    for j=1,2 do
        bytes[j] ← bytes[j]⊗map[j]
        map[j] ← f(map[j], m, ϵ,b)
```

3. Referencias

[1] Edward Ott. *Chaos in dynamical systems*. Cambridge University Press, Cambridge, U.K. New York, 2002.