PONTIFICIA UNIVERSIDAD CATÓLICA DE CHILE
ESCUELA DE INGENIERÍA
DEPARTAMENTO DE CIENCIA DE LA COMPUTACIÓN
IIC2233 — PROGRAMACIÓN AVANZADA

Actividad 12

2^o semestre 2017
26 de octubre de 2017

I/O: Archivos y *bytes*

Introducción

Luego de siglos esperando, Los De La Nazza encontraron una nave alienígena estrellada. Mientras la llevaban al Área 2233 para descubrir sus secretos, la hechicera Chau y sus *minions* los interceptaron. La hechicera, con miedo de que este importante mensaje llegara a las masas, decide obliterar la información contenida en esta nave. Sin embargo, no consideró lo avanzado de la tecnología alienígena, por lo que sólo logró desordenar la información.

Todo el mundo espera con ansias saber qué contiene este histórico mensaje, por lo que la Reina Barrios les pide a ustedes, renombrados científicos del Área 2233, que lo decodifiquen.

Instrucciones

- En la carpeta de la actividad se encuentran dos archivos, `potato.potato` y `herp.derp`. Deben unir estos archivos para poder conseguir el `.mp4` original.
- Para esto, se deben extraer trozos de bytes —desde ahora, denominados *chunks*— que se encuentran de forma intercalada entre ambos archivos. Para obtener el archivo original, deben ordenar/entrelazar los *chunks* de una forma específica.
- El tamaño de cada *chunk* va a estar determinado por los dígitos de π. Cada dígito va a representar una cantidad de *bytes* distinta.
  - $1 \rightarrow 12.000$
  - $2 \rightarrow 11.235$
  - $3 \rightarrow 6.000$
  - $4 \rightarrow 15.000$
  - $5 \rightarrow 12.345$
  - $6 \rightarrow 9.999$
  - $7 \rightarrow 22.233$
  - $8 \rightarrow 13.131$
  - $9 \rightarrow 24.000$

  En la carpeta de la actividad hay una archivo `pi.py` que contiene los dígitos de π a utilizar.

- Sumado a esto, los *chunks* vienen codificados, por lo que es necesario aplicarles un algoritmo antes de agregarlos al archivo original. Si el dígito de π asociado al *chunk* es par, se debe aplicar el algoritmo 1. En el caso contrario, se debe aplicar el algoritmo 2.
- Finalmente, el dígito 0 no está asociado a ninguna cantidad de *bytes*. Sin embargo, cuando este aparece se hace un *switch* de los algoritmos. Esto es, si cierto algoritmo está siendo aplicado a los pares ahora pasa a ser aplicado a los impares y viceversa.

Archivo 1				
Chunk-1	Chunk-3	Chunk-5	...	Chunk-n

Archivo 2				
Chunk-2	Chunk-4	Chunk-6	...	Chunk-n-1

Archivo Original				
Chunk-1	Chunk-2	Chunk-3	...	Chunk-n

Dígitos	Chunks
3	Chunk-1
1	Chunk-2
4	Chunk-3
1	Chunk-4

Algoritmo 1

- Para decodificar el *chunk* corrupto deben tomar los *bytes* de tres en tres y multiplicarlos entre sí. Luego, deben restarle el número obtenido a 255.
- Por ejemplo, si los primeros tres *bytes* del archivo son 3, 2 y 14, el *byte* final sería 171.

Algoritmo 2

- Para aplicar este algoritmo también deben tomar los *bytes* de tres en tres. Se debe invertir (espejar) todo *byte* cuyo inverso no sea mayor a 255 (en el caso contrario, se deja como está). Los *bytes* deben ser considerados como un número de tres dígitos. Es decir, el número 12 debe ser interpretado como 012 y convertido a 210. Finalmente, se debe tomar el primer dígito significativo de cada byte y concatenarlos para formar el *byte* original. Si el *byte* recibido es un cero, este debe ser tomado como tal.

Escribiendo el archivo

- Una vez obtenidos los *bytes* que componen al archivo corregido, es necesario que estos sean escritos mediante *buffering*. Esto quiere decir que la escritura se realice por medio de transportadores de información de tamaño fijo, evitando escribir la información completa de una sola vez. Este método de lectura/escritura es bastante útil cuando se manipula información de gran tamaño. Para escribir el archivo `.mp4` se deberá utilizar esta técnica con *chunks* de **2024** bytes.
- Durante cada iteración, es necesario que su programa imprima una tabla de seguimiento con la estructura mostrada a continuación:

```
|     Total      |   Procesado    |  Sin procesar  |   Deltatime    |
|   1,926,018    |     2,048      |   1,923,970    |    0.001140    |
|   1,926,018    |     4,096      |   1,921,922    |    0.001362    |
```

- **Total:** cantidad total de *bytes*.
- **Procesado:** cantidad de *bytes* procesados hasta el momento.
- **Sin procesar:** cantidad de *bytes* que faltan por procesar.
- **Deltatime:** cantidad de tiempo transcurrido desde el inicio de la escritura.

Notas

- Los archivos `potato.potato`, `herp.derp` y el video `.mp4` **no deben ser subidos a GitHub**. Para lograr esto, se recomienda fuertemente usar un `.gitignore`.
- Tengan mucho cuidado al manipular los *bytes*. Un pequeño error no permitirá recuperar el archivo original.
- No importa el nombre del archivo resultante. Solo importa que su extensión sea `.mp4`.
- Es posible que el último *chunk* pertenezca a cualquiera de los dos archivos corruptos.
- El último *chunk* va a tener un tamaño más pequeño de lo normal (la cantidad de bytes que sobran).
- Apliquen lo aprendido sobre *strings* para crear la tabla.
- Es necesario que el separador de miles sea implementado usando `format`.

Requerimientos

- (1,20 pts.) Se determina correctamente el tamaño de los *chunks*.
- (2,80 pts.) Se aplican correctamente los algoritmos.
  - (0,60 pts.) Se implementa el primer algoritmo.
  - (0,60 pts.) Se implementa el segundo algoritmo.
  - (1,60 pts.) Los algoritmos son aplicados sobre los *chunks* correspondientes.
- (1,20 pts.) Se escribe el archivo correctamente.
  - (0,60 pts.) Se aplica *buffering* correctamente.
  - (0,60 pts.) La tabla sigue la estructura solicitada.
- (0,80 pts.) Se obtiene el archivo final.

Entrega

- **Lugar:** En su repositorio de GitHub en la **carpeta** `Actividades/AC12/`
- **Hora:** 16:55
- Si están trabajando en pareja, basta con que un miembro suba la actividad. Si se suben actividades distintas, se corregirá una de las dos al azar.