Tarea 2 - ELO320 Estructura de Datos y Algoritmos
Prof.: Nicolás Gálvez R.
Ayudantes: Byron Agurto - Lucas Siclari - Gabriel Zapata
Campus San Joaquín, UTFSM. Viernes 13 de Junio de 2025

1. Discordia Game Bot: Guess the Picture

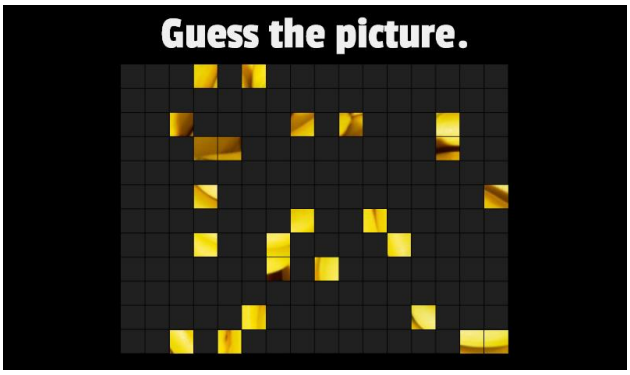

Figura 1: Discordia Game Bot - Guess the Picture

Discordia, la red de mensajería más utilizada por las nuevas generaciones y la plataforma predilecta de los alumnos de ELO320 - Estructura de Datos y Algoritmos - quiere lanzar un nuevo bot de juegos, llamado `Guess the Picture`. Este bot interactúa con un usuario, de tal forma que éste elige una figura entre 8 disponibles, la cual se irá revelando poco a poco. El jugador deberá adivinar la palabra que representa a la imagen seleccionada.

El proceso del juego es relativamente sencillo:

1. El jugador elige una imagen entre 8 figuras disponibles y tendrá 3 intentos para adivinar la palabra.
2. El juego muestra la cantidad de letras que tiene la palabra y muestra un espacio vacío por cada letra que tenga la palabra.
3. Un intento es la acción de adivinar una letra de la palabra o la palabra en sí.
4. Si el jugador adivina una letra, no pierde el intento. De caso contrario, lo pierde.
5. Si el jugador adivina la palabra, gana el juego.
6. En el primer intento se revelará el 25 % de la imagen de forma aleatoria.
7. En el segundo intento se revelará el 50 % de la imagen de forma aleatoria.
8. En el tercer intento se revelará el 75 % de la imagen de forma aleatoria.
9. Al finalizar el juego se revela la imagen de forma completa.

Las imágenes que muestra el juego están en formato *Portable PixMap*, cuya extensión es `.ppm`. En este formato, una imagen representada por un archivo de texto plano parecido al del siguiente ejemplo:

```
P3
3 3
255
27 255 100
10 10 198
222 255 10
2550 255 255
33 66 99
111 111 111
77 144 211
211 7 99
99 99 99
```

Figura 2: Formato `.ppm`. ¿Qué contiene este archivo?

En estas imágenes:

- La primera línea del archivo indica el formato de imagen (P3 $\rightarrow$ RGB 24 bits).
- La segunda línea indica el tamaño de la imagen como matriz de píxeles: cantidad de columnas por cantidad de filas (3×3).
- La tercera fila indica el final de la cabecera del archivo.
- El resto del contenido indica los valores RGB (del inglés *Red, Green and Blue*) de cada píxel de la imagen.
  - Los valores varían entre el 0 y el 255 en formato entero.
  - Se agrupan de a tres valores para indicar el color resultante del píxel: así los primeros tres números son los valores de Rojo, Verde y Azul del primer píxel, respectivamente; los siguientes tres valores corresponden al segundo píxel; y así sucesivamente.
  - Los valores no necesariamente están agrupados de a tres por cada línea del archivo: pueden estar de cualquier forma luego de la linea de fin de la cabecera.

Sin embargo, y para evitar trampas de los usuarios, el bot almacena las imágenes encriptadas en un archivo de **Discordia Images**, cuya extensión es `.dis`. La encriptación de un archivo `.ppm` a `.dis` ocurre de la siguiente forma:

1. Se mantienen las líneas de la cabecera del archivo `.ppm`.
2. Se agrupan la información de cada píxel, vale decir, se forman conjuntos de tres (3) valores que representan el color RGB del pixel, concatenando sus valores. Note que:
   - Si un valor es de un dígito, se le anteponen dos ceros (00) antes de la concatenación.
   - Si un valor es de dos dígitos, se le antepone un cero (0) antes de la concatenación.
3. Se le asigna un id a cada conjunto que es mayor que el id del conjunto anterior por un valor aleatorio entre uno (1) y nueve (9). Note que el primer conjunto se compara respecto al número cero (0).
4. Luego se colocan en una línea del archivo el id y el valor concatenado del conjunto. La línea donde es colocada es asignada seleccionada aleatoriamente, vale decir, el archivo está desordenado.
5. Habrá una línea por cada píxel de la imagen.

```
P3
3 3
255
42 111111111
21 222255010
18 010010198
63 099099099
57 211007099
34 033066099
25 255255255
9 027255100
48 077144211
```

Figura 3: Formato `.dis`: Encriptando un archivo `.ppm`.

6. El resultado es un archivo `.dis`, como el del siguiente ejemplo:

Ud. es contratado por Discordia para programar el Bot Game de `Guess the Picture` en C, en el cual debe ir desencriptando la imagen, vale decir, leer el archivo `.dis` y ordenar los datos para obtener la información RGB de cada píxel y así regenerar la imagen `.ppm`. Su jefe también le entrega la siguiente cabecera para realizar esta labor:

`discordia.h`

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

typdef struct pixel{
    long int id; // id del conjunto de datos del pixel.
    int red; // valor del color rojo del pixel.
    int green; // valor del color verde del pixel.
    int blue; // valor del color azul del pixel.
} pix_t;
```

Además, su jefe **le obliga a usar esta estructura y le comenta que no le puede eliminar nada, solo agregar cosas nuevas**. Y también le entrega la información que todo archivo `.ppm` puede ser abierto en cualquier visor de imágenes. A pesar de esta libertad para su desarrollo, le recomienda el uso de la librería **imlib2** para carga nativa de las imágenes desde cualquier programa C.

Para satisfacer las necesidades de Discordia, ud debe crear el juego Guess The Picture, considerando lo siguiente:

(a) (20 points) Leer una imagen encriptada `.dis`, cargando su información en una Estructura de Datos basada en arreglos de su elección. **El nombre del archivo debe ser entregado como parámetro del ejecutable**.

(b) (30 points) Elegir tres algoritmos de ordenamiento para desencriptar parcialmente la información obtenida del archivo *.dis*. Estos deben poseer las siguientes complejidades, sin repetirse:

- Revelación del 25 % aleatorio $\rightarrow$ algoritmo $\mathcal{O}(n^2)$,
- Revelación del 50 % aleatorio $\rightarrow$ algoritmo $\mathcal{O}(n \log n)$, y
- Revelación del 75 % aleatorio y 100 % $\rightarrow$ algoritmo $\mathcal{O}(d \times (n + k))$;

(c) (20 points) Transformar la información desencriptada a una imagen formato `.ppm` del tamaño indicado en el archivo `.dis`. Para la revelaciones parciales, ocupe el valor $(0, 0, 0)$, para los pixeles no seleccionados (pixel negro). Verifique su correctitud si ocurre una situación similar a la descrita en la Figura 4.

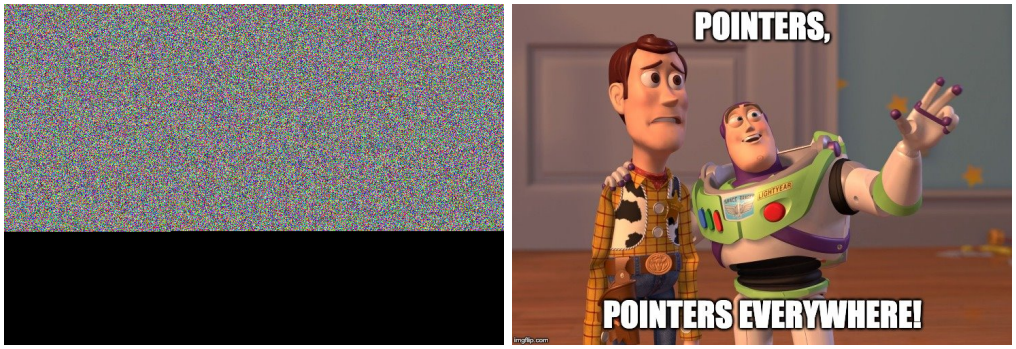

Figura 4: Desencriptar: de `.dis` a `.ppm`

(d) (30 points) Generar un reporte, en el archivo de texto plano `analisis.txt`, que mida empíricamente la complejidad de los algoritmos seleccionados. Además en el README debe incluir una pequeña explicación del por qué de sus resultados. Esta labor la puede realizar de diversas formas:

- Midiendo el tiempo que toman cada uno de los algoritmos, a través de funciones de la biblioteca `<time.h>`.
- Contando la cantidad instrucciones realizadas en ejecución por su algoritmo. Considere que puede escoger los siguientes escenarios: optimista, real y/o pesimista.
- Contando la cantidad de comparaciones que hacen los algoritmos.
- Mostrando la cantidad de memoria usada en ejecución.
- Cualquier otro criterio visto en clases.

2. Reglas de entrega y Consideraciones Generales

- Este trabajo debe realizarse **individualmente**, vale decir, **grupos de un estudiante**. No se harán excepciones.
- Dos archivos encriptados de imágenes se encontrarán disponibles en AULA USM. Todos las imágenes fueron obtenidas de **Google Images**.
- El programa debe ser desarrollado en lenguaje C, y compilado con la versión de gcc disponible en el servidor Aragorn[1]: `gcc 4.8.5`.
- La tarea debe ser entregada en la plata forma **AULA USM**, el día Viernes 27 de Junio de 2025 hasta las 23:59:59 Hora de Chile Continental (UTC -4).
- La tarea debe incluirse en un archivo comprimido **.tar.gz**. El nombre del archivo debe seguir la siguiente estructura: `tarea2-eda-nombre-apellido-paralelo.tar.gz`; e.g., `tarea2-eda-oliver-atom-p200.tar.gz`.
- El archivo comprimido debe incluir lo siguiente:
  - Cabecera `.h`: Archivo cabecera en el cual se deben incluir todas las bibliotecas a usar en el programa, además de las definiciones de macros, variables globales, tipos de datos personalizados, struct y prototipos de todas las funciones.
  - Código `.c`: Código del programa solicitado. Considere que éste es un desafío de al menos tamaño mediano, por lo tanto es recomendable dividir el código en diferentes archivos agrupados por su finalidad.
  - `README`: Archivo de texto plano en el cual se debe incluir: (1) una pequeña reseña del programa, (2) las condiciones de compilación y ejecución, (3) las instrucciones de compilación y ejecución, y (4) la información del creador del sistema.
  - `Makefile`: Archivo de compilación automática del sistema. Una cápsula de vídeo para su confección se encontrará disponible en AULA USM.
- La revisión de los sistemas se hará utilizando diferentes archivos de los entregados en AULA USM. Por lo tanto, debe programar de forma genérica para cualquier cantidad de información de entrada.
- Cada fuga de memoria será penalizada. Utilice `valgrind` para verificar la correcta asignación, uso y liberación de memoria en la ejecución de su programa.
- Si su programa no compila, su nota será automáticamente **un cero (0)**.
- Cualquier atisbo de copia, será penalizada con máxima severidad.

[1] `ssh aragorn.elo.utfsm.cl -l cuentausm`