Instituto Politécnico Nacional

Escuela de Superior de Computoo

Reporte de Programas Primera Unidad

Uriel Loeza Caudillo

Grupo: 5BM1

Asignatura: Teoria de la computación

Fecha de entrega: 7 de abril del 2024

Índice general

Índice de figuras

Capítulo 1

Programa 1. Universo

1.1. Descripción del problema

El programa que se desarrollará será capaz de trabajar con el universo de cadenas binarias de longitud n (Σ^n), donde el valor de n podrá ser ingresado por el usuario o determinado automáticamente por el programa dentro del intervalo [0, 1000]. Este programa operará en dos modos: automático, donde se realizará el cálculo de todas las cadenas de manera secuencial, y manual, permitiendo al usuario ingresar una cadena específica para su evaluación. Además, después de cada cálculo, el programa preguntará al usuario si desea calcular otra cadena, y podrá salirse del ciclo cuando así lo indique el usuario.

La salida del programa se presentará en notación de conjunto y se guardará en un archivo de texto. Además, se realizará un análisis adicional específico para el caso donde $n = 28$. Para ello, se calculará el número de unos en cada cadena y se graficará esta información, con el eje x representando las cadenas y el eje y representando el número de unos en cada cadena. Posteriormente, se realizará el mismo cálculo de gráfica pero utilizando el logaritmo base 10 de los valores de y. Este proceso se documentará detalladamente en un reporte, proporcionando una explicación completa del análisis, los cálculos realizados y las gráficas generadas.

1.2. Propuesta de solución

Para generar el universo, se parte del caso base, el cual contiene el conjunto inicial {.,#}. Luego, se inicia un ciclo que genera dos archivos de texto modificando este conjunto base. El primero, denominado “ceros”, agrega un "# " al final de cada elemento del conjunto, mientras que el segundo, llamado “unos”, realiza la misma acción pero agregando un “.”. Posteriormente, se combinan de forma ordenada los elementos de los tres archivos y se genera un nuevo caso base que corresponde al siguiente valor de “n” en el ciclo. Este proceso se repite hasta alcanzar el valor deseado de “n”.

1.3. Metodología

1.3.1. Selección de modo

El fragmento de código a continuación representa la selección del modo de operación del programa. Se le da la bienvenida al usuario al "Universo de cadenasz se le solicita ingresar hasta qué número desea observar el universo. Si el usuario ingresa el valor 0, el programa seleccionará aleatoriamente un número. Dependiendo de la respuesta del usuario, la variable `num` se asignará a un número aleatorio o al número ingresado por el usuario.

```
    printf("Bienvenido al Universo de cadenas\n");
    printf("Hasta qué número desea observar el universo?\n");
    printf("Si ingresa un 0, se seleccionará aleatoriamente\n");


    int validacion = 0;
    int num = 0;
    validacion= scanf("%d", &num);
    if (validacion == 1){
        srand(time(NULL));
        int numeroAleatorio = rand() % 15;
        num=numeroAleatorio;
    }
    else{
        num= scanf("d",&num);
    }
    printf("El numero recibido es %d\n", num);
```

1.3.2. Creación del caso base

Se crea un archivo de texto llamado `base.txt`, en el cual se imprime el conjunto correspondiente a $n = 1$.

```
        FILE *base = fopen("base.txt", "w");
    // Validación
    if (base == NULL) {
        perror("Error al crear el archivo base");
        return 1;
    }
    fprintf(base, "#,.,");
    fclose(base);
```

1.3.3. Generación de los textos de apoyo

En esta primera parte del ciclo se crean los archivos de texto `unos` y `ceros` en modo lectura y escritura, y se abre el archivo `base.txt` en modo lectura. A continuación, se lee

el archivo hasta llegar al final del archivo (EOF), y se agregan el carácter "# " o "." al encontrar una coma, respectivamente.

```
1
2       for (int i=1; i<num;i++){
3
4           FILE *base_mod = fopen("base.txt", "r");
5           FILE *ceros = fopen("ceros.txt", "w+");
6
7           // Validación
8           if (base_mod == NULL || ceros == NULL) {
9               perror("Error al abrir los archivos");
10              return 1;
11          }
12
13          // Leer caracteres uno por uno de base.txt
14          int c;
15          while ((c = fgetc(base_mod)) != EOF) {
16              // Si encontramos una coma, escribimos # en ceros.txt
17              if (c == ',') {
18                  fprintf(ceros, "#,");
19              }
20              // Si no, escribir el caracter original en ceros.txt
21              else {
22                  fputc(c, ceros);
23              }
24          }
25
26          //Apuntamos al inicio del archivo
27          fseek(base_mod,0,SEEK_SET);
28
29          FILE *unos = fopen("unos.txt", "w+");
30          // Validación
31          if (base_mod == NULL || unos == NULL) {
32              perror("Error al abrir los archivos");
33              return 1;
34          }
35          // Realizamos el mismo procedimiento pero ahora agregando un .
36          while ((c = fgetc(base_mod)) != EOF) {
37              if (c == ',') {
38                  fprintf(unos, ".,");
39              } else {
40                  fputc(c, unos);
41              }
42          }
```

```
43          fclose(base_mod);
```

1.3.4. Creación del nuevo caso base

En la segunda parte del ciclo, se vuelve a abrir `base.txt` en modo escritura para sobreescribirlo con los caracteres {.,#}, lo cual ayuda a evitar la repetición de instancias. Luego, se copian los conjuntos que se encuentran en los archivos de apoyo para así generar el nuevo caso base.

```
1
2   char caracter;
3
4           // Abrir el archivo de salida para escritura
5           FILE* base_mod_reinicio = fopen("base.txt", "w");
6           fprintf(base_mod_reinicio,"#,.,");
7
8           // Verificar si el archivo de salida se abrió correctamente
9           if (base_mod_reinicio == NULL) {
10              printf("No se pudo crear el archivo de salida.\n");
11              exit(1);
12          }
13
14          fseek(unos,0,SEEK_SET);
15          fseek(ceros,0,SEEK_SET);
16          // Leer y escribir el contenido de los archivos originales
17          while ((caracter = fgetc(ceros)) != EOF) {
18          fputc(caracter, base_mod_reinicio);
19              if (caracter == ',') {
20                  while ((caracter = fgetc(unos)) != EOF){
21                      putc(caracter, base_mod_reinicio);
22                      if (caracter == ','){
23                          break;
24                      }
25                  }
26              }
27          }
28
29          // Cerrar los archivos
30          fclose(base_mod_reinicio);
31          fclose(unos);
32          fclose(ceros);
```

1.3.5. Creación del contador

Este código cuenta con la variable `cont` que se incrementa cada vez que se encuentra un punto en el archivo `base.txt`. Luego, cuando se encuentra una coma, imprime el valor del contador en el archivo, seguido de una coma y un salto de línea.

```
1
2  FILE *universo = fopen("base.txt", "r");
3  FILE *contador = fopen("contador.csv", "w");
4  char c;
5  int cont;
6
7  while ((c = fgetc(universo)) != EOF) {
8      if (c == '.') {
9          cont++;
10     } else if (c == ',') {
11         fprintf(contador, "%d,\n", cont);
12         cont = 0;
13     }
14 }
```

1.3.6. Creación del grafico

Este código en MATLAB lee datos desde un archivo CSV `contador.csv`, que contiene el número de unos en cada cadena en el universo, y los almacena en la variable `eje_y`. Luego, se genera un vector `eje_x` que representa los índices de los datos en el eje x. Utilizando estos datos, se crea un gráfico de dispersión donde cada punto representa el número de unos en una cadena específica. Finalmente, se ajusta la posición de la figura antes de guardarla como una imagen PNG llamada `grafico_normal.png`.

```
1  %Declaracion  de variables
2  eje_y = readmatrix('contador.csv')
3
4  eje_x = 1:length(eje_y)
5  fig = figure;
6
7  plot(eje_x, eje_y, '.');
8  ylim([0 30])
9  title('Repeticiones de unos');
10 xlabel('Index');
11 ylabel('Unos');
12 set(fig, 'Position', [100, 100, 1000, 1000]);
13
14 saveas(fig, 'grafico_normal.png');
```

Para el caso del gráfico con logaritmo, basta con aplicarle un logaritmo a la variable `eje_y`.

```
yLog = log(eje_y); % Aplicar logaritmo
```

1.4. Resultados

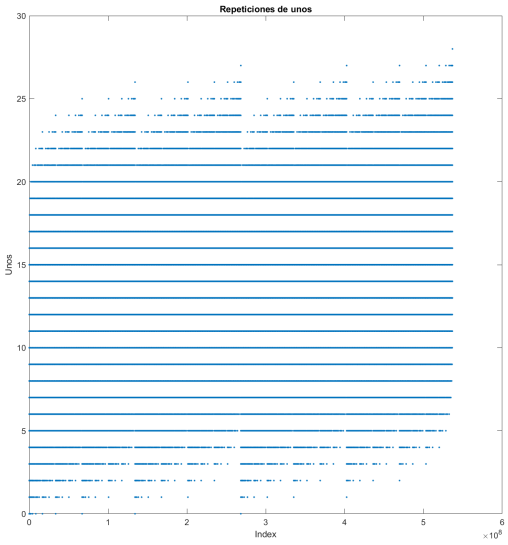

Figura 1.1: Grafico del universo con n=28

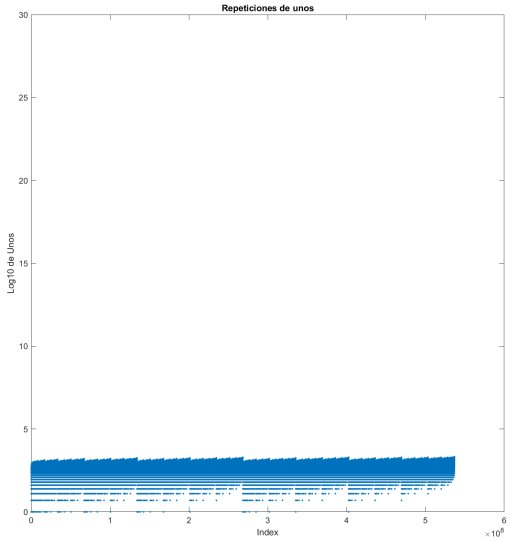

Figura 1.2: Grafico del universo con n=28, y aplicando un logaritmo

1.5. Conclusiones

Durante el desarrollo de este código, se resaltó la importancia de los archivos de texto para la generación de datos abundantes. Sin su uso, el programa no podría generar el universo, ya que la memoria RAM se agotaría si se utilizaran cadenas de caracteres para el almacenamiento. Además, se demostró la capacidad de MATLAB para graficar con una gran cantidad de datos. Se intentó realizar la misma tarea con Python, pero no se logró debido a limitaciones de rendimiento. Este hecho resalta la eficacia y versatilidad de MATLAB en el procesamiento y visualización de grandes conjuntos de datos.

Capítulo 2

Programa 2. Juego

2.1. Descripción del problema

El programa a desarrollar permitirá a dos jugadores realizar movimientos ortogonales y diagonales en un tablero de ajedrez de 4x4. Los movimientos y reglas seguirán las especificaciones del curso de Stanford. El software contará con dos modos de juego: automático y manual. En el modo manual, los usuarios podrán ingresar la cadena de movimientos o generarla aleatoriamente. Cada jugador tendrá un objetivo específico: el primero deberá ir desde la posición 1 hasta la 16, mientras que el segundo deberá ir desde la posición 4 hasta la 13. Se implementará una función para determinar aleatoriamente quién inicia el juego. Además, el programa generará archivos con todos los movimientos posibles y los movimientos ganadores por pieza, que podrán ser reconfigurados para cambiar las rutas, aunque si no se puede avanzar, se requerirá esperar una iteración para continuar. Se ofrecerá una representación gráfica del tablero y de los movimientos realizados, así como de la red (NFA) generada por los movimientos de ambos jugadores. El número máximo de movimientos permitidos se establecerá entre 4 y 100 símbolos.

2.2. Propuesta de solución

Para encontrar los caminos, se propuso generar una red con estructuras y memoria dinámica en C. A diferencia del árbol, esta red genera una cantidad menor de nodos, lo que la hace una aproximación más óptima. Luego, se utilizó una búsqueda por profundidad para encontrar todos los caminos desde la primera capa hasta la última. Finalmente, se filtraron estos caminos para obtener solo las rutas ganadoras en otro archivo. Para la animación, se hizo uso de Python y la librería Pygame.

2.3. Metodología

2.3.1. Definición de estructuras

La estructura `Capa` se utiliza para organizar los nodos de la red y sus conexiones. Contiene un entero que guarda el número de objetos en la capa y un doble apuntador a

otros objetos para almacenar n nodos.

```
typedef struct Capa {
    int num_objetos;
    struct Objeto **objetos;
} Capa;

```

La estructura `Objeto` representa un nodo de la red. Contiene un identificador para guardar el número de casilla que representa, un apuntador a `Capa` para determinar en qué capa se encuentra, un doble apuntador a `Objeto` para almacenar las conexiones con los otros nodos, y un entero que guarda el número de conexiones.

```
typedef struct Objeto {
    int id;
    struct Capa *capa;
    struct Objeto **conexiones;
    int num_conexiones;
} Objeto;

```

2.3.2. crear_objeto()

Esta función asigna la memoria dinámica e inicializa los valores de un objeto.

```
Objeto* crear_objeto(int id, Capa *capa) {
    Objeto *obj = (Objeto*)malloc(sizeof(Objeto));
    obj->id = id;
    obj->capa = capa;
    obj->conexiones = NULL;
    obj->num_conexiones = 0;
    return obj;
}

```

2.3.3. conectar_objeto()

Parámetros:

- `Objeto *obj1`: Objeto actual.
- `Objeto *obj2`: Objeto con el cual se quiere conectar.

Funcionalidad: Esta función comprueba si ya se tienen conexiones. Si no se tienen, se asigna la memoria dinámica y se asigna el apuntador. En otro caso, se hace la reasignación para almacenar al nuevo apuntador. En ambos casos se incrementa el número de conexiones del objeto.

```
void conectar_objetos(Objeto *obj1, Objeto *obj2) {
    // Aumentar el tamaño del array de conexiones de obj1 para incluir una nueva conexión
    obj1->conexiones = (Objeto**)realloc(obj1->conexiones, (obj1->num_conexiones + 1) * s
    if (obj1->conexiones == NULL) {
        printf("Error al realocar memoria para las conexiones\n");
        exit(1); // O manejar el error de manera más apropiada
    }

    // Añadir obj2 al array de conexiones de obj1
    obj1->conexiones[obj1->num_conexiones] = obj2;
    obj1->num_conexiones++; // Incrementar el contador de conexiones
}
```

2.3.4. verificar_o_crear_y_conectar()

Parámetros:

- `Objeto* objeto_actual`: Es el objeto en el que se encuentra el programa.
- `int idObjetivo`: Es el ID del objeto con el que se quiere conectar.
- `Capa** capas`: Es la capa de la red.
- `int capaSiguiente`: El número de la capa donde está o va a estar el objeto a conectar.
- `int* cont_obj`: Sirve para que no se sobrescriban objetos en las conexiones.

Funcionalidad: Esta función sirve para acortar las líneas de código. Lo primero que hace es suponer que el objeto con el que se quiere conectar ya existe en la siguiente capa. Por lo tanto, busca en esta. Si lo encuentra, simplemente llama a la función `conectar_objetos`. Si no lo encuentra, asigna `false` al booleano y crea el nodo desde cero en la siguiente capa. Luego, crea la conexión.

```
void verificar_o_crear_y_conectar(Objeto* objeto_actual, int idObjetivo, Capa** capas, in
    bool existe = false;
    for (int k = 0; k < capas[capaSiguiente]->num_objetos; k++) {
        if (capas[capaSiguiente]->objetos[k]->id == idObjetivo) {
            conectar_objetos(objeto_actual, capas[capaSiguiente]->objetos[k]);
            existe = true;
            break;
        }
    }
    if (!existe) {
        Objeto* nuevo_objeto = crear_objeto(idObjetivo, capas[capaSiguiente]);
        if (capas[capaSiguiente]->num_objetos == 0) {
            capas[capaSiguiente]->objetos = (Objeto**)malloc(sizeof(Objeto*));
```

```
14          } else {
15              capas[capaSiguiente]->objetos = (Objeto**)realloc(capas[capaSiguiente]->objet
16          }
17          capas[capaSiguiente]->objetos[*cont_obj] = nuevo_objeto;
18          capas[capaSiguiente]->num_objetos++;
19          (*cont_obj)++;
20          conectar_objetos(objeto_actual, nuevo_objeto);
21      }
22  }
23
```

2.3.5. dfs()

Parámetros:

- `Objeto *obj`: El objeto desde el cual comienza la búsqueda.
- `char *camino`: Una cadena de caracteres que registra el camino recorrido hasta el momento.
- `int profundidad`: La profundidad actual de la búsqueda en la red.
- `int *contador`: Un contador que se usa para llevar un registro de la cantidad de caminos.
- `FILE* caminos`: Un archivo donde se guardan los caminos encontrados.
- `int num_capas`: El número total de capas en la red.
- `Capa** capas`: Un arreglo de punteros a las capas que componen la red.

Funcionalidad: La función de Búsqueda en Profundidad (DFS) en el programa analizado se encarga de explorar y documentar caminos a través de una red compuesta por objetos organizados en múltiples capas. Comienza su operación desde un objeto de partida, al cual le anexa su identificador a un registro del camino que ha sido recorrido hasta el momento. Cuando se llega a un objeto que pertenece a la última capa de la red, se interpreta que se ha encontrado un camino completo, el cual se procede a guardar en un archivo específico. Para aquellos objetos que se encuentran en capas intermedias, la función se invoca a sí misma de manera recursiva para cada una de sus conexiones, duplicando el registro del camino actual con cada llamada para preservar la independencia de cada ruta explorada.

```
1  void dfs(Objeto *obj, char *camino, int profundidad, int *contador, FILE* caminos, i
2      char idStr[12];
3      sprintf(idStr, "%d ", obj->id);
4
5      if (strlen(camino) + strlen(idStr) < MAX_CAMINO_LEN - 1) {
6          strcat(camino, idStr);
```

```
7       } else {
8           printf("Error: Camino excede longitud máxima.\n");
9           return;
10      }
11
12      // Si el objeto es de la última capa, guardamos el camino
13      if (obj->capa == capas[num_capas ]) {
14          fprintf(caminos,"%s\n",camino);
15      } else {
16          for (int i = 0; i < obj->num_conexiones; i++) {
17              char newCamino[MAX_CAMINO_LEN];
18              strcpy(newCamino, camino); // Copiamos el camino actual para evitar alte
19              dfs(obj->conexiones[i], newCamino, profundidad + 1, contador, caminos, nu
20          }
21      }
22  }
```

2.3.6. encontrar_caminos()

Parámetros:

- `Capa **capas`: Un puntero a un arreglo de punteros a Capa, que representa la estructura de la red compuesta por diferentes capas de objetos.
- `int num_capas`: El número total de capas que componen la red.
- `FILE* caminos`: Un puntero a un archivo donde se desean guardar los caminos encontrados durante la búsqueda.

Funcionalidad:

La función inicia con un contador seteado en cero, para llevar el seguimiento de los caminos encontrados o para otro tipo de control dentro de la búsqueda. Se posiciona al inicio del archivo `caminos` mediante `fseek`, indicando que se escribirán o leerán caminos desde el principio del archivo. Inicia un bucle que recorre todos los objetos de la primera capa, asumiendo que estos representan los puntos de partida para la búsqueda de caminos. Para cada objeto en la primera capa, inicializa una cadena de caracteres `camino` para almacenar el recorrido que se va generando a medida que se explora la red. Invoca la función `dfs` previamente descrita, pasándole como argumentos el objeto actual, la cadena de camino inicializada, un contador para profundidad (iniciado en 0), un puntero al contador de caminos, el archivo `caminos`, el número total de capas y el puntero a las capas de la red.

```
1   void encontrar_caminos(Capa **capas, int num_capas, FILE*caminos) {
2       int contador = 0;
3       fseek(caminos,0,SEEK_SET);
4       // Iniciar DFS desde la primera capa
```

```
    for (int i = 0; i < capas[0]->num_objetos; i++) {
        char camino[MAX_CAMINO_LEN] = {0}; // Inicializa el camino
        dfs(capas[0]->objetos[i], camino, 0, &contador, caminos,num_capas ,capas);
    }
}
```

2.3.7. caminos_correctos()

Parámetros:

- `int num_capas`: El número de capas generado en la red.
- `FILE* caminos`: Archivo donde se guardaron todos los caminos.
- `FILE* camino_correcto`: Archivo donde se guardarán los caminos correctos.
- `char* final`: Cadena para verificar el último nodo del camino y determinar si es correcto o no.

Funcionalidad:

La función recorre el archivo recibido en `caminos` y verifica si los últimos caracteres son iguales al carácter recibido en `final`, agregándolos a `caminos_correctos` en caso de ser iguales.

```
void caminos_correctos(int num_capas, FILE*caminos, FILE*camino_correcto, char* final
    fseek(caminos, 0, SEEK_SET);
    char linea[MAX_CAMINO_LEN] = {0};
    while(fgets(linea, MAX_CAMINO_LEN, caminos) != NULL){
         fgets(linea, MAX_CAMINO_LEN, caminos);
         int longitud = strlen(linea);
         char seis = linea[longitud - 3];
         char uno = linea[longitud - 4];
         if (uno==final[0] && seis==final[1]){
            fprintf(camino_correcto,linea);
         }
    }
}
```

2.3.8. crearCaminos()

Parámetros:

- `FILE* cadena`: Archivo donde se tiene la cadena para crear la red.
- `FILE* caminos`: Archivo donde se almacenarán todos los caminos.
- `FILE* camino_correcto`: Archivo donde se almacenarán los caminos correctos.

- `int inicio`: Entero que indica en qué casilla empieza el jugador.
- `char* final`: Cadena que sirve para verificar si un camino termina en la casilla deseada, convirtiéndolo en correcto.

Funcionalidad:

Esta es la función más importante del programa, ya que hace uso de las demás funciones para crear la red, encontrar los caminos y filtrarlos. En primer lugar, determina el número de capas que necesitará la red. Luego, inicializa la primera capa con el número enviado en `inicio`. A continuación, procede a crear los objetos y las conexiones de acuerdo a la lógica establecida en el curso. Una vez completada esta etapa, obtiene los caminos utilizando las funciones descritas previamente. Finalmente, se encarga de liberar toda la memoria dinámica generada durante el proceso.

2.3.9. Generacion de cadenas

Para la generación de las cadenas de los jugadores se implementó un programa en Python.

Inicialmente, el programa consulta al usuario si desea ingresar las cadenas manualmente o si prefiere que se generen automáticamente.

Si el usuario elige ingresar las cadenas manualmente, se le solicitará que introduzca las cadenas a través de dos entradas de texto, denominadas `cadena1_usuario` y `cadena2_usuario`. Estas cadenas se almacenarán en los archivos `cadena1.txt` y `cadena2.txt`, respectivamente.

```
print("Desea ingresar las cadenas para los jugadores?\n")
respuesta = input("1.Si\n2.No\n")
if respuesta == '1':
    with open('cadena1.txt','w',encoding='utf-8') as cadena1:
        cadena1_usuario=input("Ingrese la primer cadena, ")
        print("recuerde que el jugador uno debe terminar con b\n")
        cadena1.write(cadena1_usuario)
    with open('cadena2.txt','w',encoding='utf-8') as cadena2:
        cadena2_usuario=input("Ingrese la segunda cadena, ")
        print("recuerde que el jugador dos debe terminar con r\n")
        cadena2.write(cadena1_usuario)
```

En caso contrario, el programa generará automáticamente las cadenas. Para esto, primero se generará un número aleatorio que definirá la longitud de la cadena. Luego, se realizará un ciclo que se ejecutará este número menos uno veces, en el cual se generará otro número aleatorio entre 0 y 1. Si este número es 0, se escribirá una 'r'; si es 1, se escribirá una 'w'. Finalmente, se agregará una 'w' o una 'b' al final de la cadena, dependiendo del jugador.

```
else:
    iteracion=random.randint(4,15)
    with open('cadena1.txt','w',encoding='utf-8') as cadena1:
        for i in range(iteracion-1):
            random1=numero = random.randint(0, 1)
            if random1 == 0:
                cadena1.write("r")
            else:
                cadena1.write("b")
        cadena1.write("b")
    with open('cadena2.txt','w',encoding='utf-8') as cadena2:
        for i in range(iteracion-1):
            random1=numero = random.randint(0, 1)
            if random1 == 0:
                cadena2.write("r")
            else:
                cadena2.write("b")
        cadena2.write("r")
```

2.4. Animación

Antes de comenzar la animación, se creó una lista para representar el tablero. Cada elemento de la lista contiene el número de la casilla y otro entero inicializado en 0, que indica si un jugador se encuentra en esa casilla.

```
tablero = [None for _ in range(16)]

for i in range(16):
        tablero[i] = [i+1, 0]
```

Se creó una lista de 2x1 para representar a los jugadores. El primer elemento representa la casilla en la que se encuentra el jugador, y el segundo elemento indica el número de movimientos realizados.

```
agente=[1,1]
agente2=[4,1]
```

Después, se verifica la longitud de las soluciones para determinar si un jugador ha realizado el número máximo de movimientos posibles.

```

cont_agente_dos=0
with open('caminos.txt', 'r') as archivo:
     primera_linea = archivo.readline()
```

```
    for c in primera_linea:
        if c==' ':
            cont_agente_dos+=1

cont_agente_dos-=1
print(cont_agente_dos)
```

Para continuar, se selecciona una solución aleatoria y se convierte a una lista de enteros para el manejo del tablero.

```
numero_linea_seleccionada = 0
contador_lineas = 0
with open('caminos_correctos.txt', 'r') as archivo:
    # Lee cada línea del archivo
    for linea in archivo:
        # Incrementa el contador de líneas
        contador_lineas += 1

        # Con una probabilidad de 1/contador_lineas, actualiza la línea seleccionada
        if random.random() < 1.0 / contador_lineas:
            linea_seleccionada = linea.strip()
            numero_linea_seleccionada = contador_lineas

#Conversion a arreglo de enteros
linea_seleccionada = linea_seleccionada.split()

solucion_agente = enteros = [int(x) for x in linea_seleccionada]
print(solucion_agente)
```

Finalmente, antes de entrar de lleno con la animación, se crea el estado inicial posicionando los jugadores en el tablero. También se selecciona de forma aleatoria qué jugador iniciará. Esto se logra inicializando el iterador en un número impar o par. Si el iterador es par, es turno del jugador 1; si es impar, es turno del jugador 2.

```
tablero[0]=[1,1]
tablero[3]=[4,2]
print("Estado inicial")
print(tablero)

turno = random.randint(0, 1)

if turno%2==0:
    print("Empieza el jugador 1")
else:
    print("Empieza el jugador 2")
```

Para realizar la animación, se crea un bucle infinito que solo se detiene si uno de los agentes cumple las condiciones de paro. Estas condiciones son que se encuentre en una casilla final y que haya realizado el número máximo de movimientos posibles. Además, se generó una lista con los centros de cada recuadro del tablero para facilitar el dibujo de los jugadores.

```
arreglo_pos = [(100,100),(300,100),(500,100),(700,100),(100,300),(300,300),(500,300)
while True:
   if agente == [16, cont_agente_uno+1] or agente2 == [13, cont_agente_dos+1]:
        break
```

En cada iteración, se pinta el tablero y luego los jugadores, para así generar la animación.

```
pantalla.fill(ROJO)
    pygame.draw.rect(pantalla, NEGRO, (0,0, 200, 200))
    pygame.draw.rect(pantalla, NEGRO, (400,0, 200, 200))
    pygame.draw.rect(pantalla, NEGRO, (200,200, 200, 200))
    pygame.draw.rect(pantalla, NEGRO, (600,200, 200, 200))

    pygame.draw.rect(pantalla, NEGRO, (0,400, 200, 200))
    pygame.draw.rect(pantalla, NEGRO, (400,400, 200, 200))
    pygame.draw.rect(pantalla, NEGRO, (200,600, 200, 200))
    pygame.draw.rect(pantalla, NEGRO, (600,600, 200, 200))

    print("Dibujo")
    print(agente)
    dibujo_agt1 = arreglo_pos[agente[0]-1]
    print(dibujo_agt1)
    pygame.draw.circle(pantalla, BLANCO, dibujo_agt1, 50)

    dibujo_agt2 = arreglo_pos[agente2[0]-1]
    pygame.draw.circle(pantalla, AZUL, dibujo_agt2, 50)

    pygame.display.flip()
```

Para resolver el caso en el que dos jugadores se encuentran, se verifica si en la casilla a la que se va a mover no hay otro jugador. Si es el caso, se realiza lo siguiente: se abre el archivo de caminos_correctos del jugador y se itera sobre las líneas, verificando que en la posición actual haya una casilla diferente. Si se encuentra una solución con una casilla alternativa, se reemplaza el camino elegido. Si no, se aumenta el iterador y ni el tablero ni el jugador sufren modificaciones.

En otro caso, se actualiza el tablero y el jugador.

```
if turno%2==0:
        # dibujo_agt1 = arreglo_pos[agente[0]-1]
```

```
3           # pygame.draw.circle(pantalla, BLANCO, dibujo_agt1, 50)
4           print("Turno agente 1")
5           if tablero[solucion_agente[pos_camino_agt]-1] == [solucion_agente[pos_camino
6               with open("caminos_correctos.txt","r") as archivo:
7                   for linea in archivo:
8                       linea=linea.split()
9                       posible_solucion = enteros = [int(x) for x in linea]
10                      if posible_solucion[pos_camino_agt] != solucion_agente[pos_camin
11                          print("solucion alternativa encontrada!")
12                          print(posible_solucion)
13                          solucion_agente=posible_solucion
14                          agente[0] = solucion_agente[pos_camino_agt]
15                          tablero[solucion_agente[pos_camino_agt]-1]=[solucion_agente
16                          tablero[solucion_agente[pos_camino_agt-1]-1]=[solucion_agen
17                          print(tablero)
18                          pos_camino_agt+=1
19                          agente[1]=pos_camino_agt
20                          print(agente)
21                          turno+=1
22                          break
23                      else:
24                          print("Paso")
25                          print(agente)
26                          turno+=1
27                          break
28
29
30          else:
31              tablero[solucion_agente[pos_camino_agt]-1]=[solucion_agente[pos_camino_ag
32              tablero[solucion_agente[pos_camino_agt-1]-1]=[solucion_agente[pos_camino
33              print(tablero)
34              agente[0] = solucion_agente[pos_camino_agt]
35              pos_camino_agt+=1
36              agente[1]=pos_camino_agt
37              print(agente)
38              turno+=1
```

2.5. Resultados

En este ejemplo se uso la cadena ´´brrbbrb" para el jugador 1 y la cadena ´´bbrbrbr´´ para el jugador 2.

```
213   1 6 7 2 6 9 5 6
214   1 6 7 2 6 9 5 1
215   1 6 7 2 6 9 5 9
216   1 6 7 2 6 9 13 9
217   1 6 7 2 6 9 13 14
218   1 6 7 2 6 9 10 5
219   1 6 7 2 6 9 10 7
220   1 6 7 2 6 9 10 13
221   1 6 7 2 6 9 10 15
222   1 6 7 2 6 11 7 6
223   1 6 7 2 6 11 7 3
224   1 6 7 2 6 11 7 8
225   1 6 7 2 6 11 7 11
226   1 6 7 2 6 11 15 11
227   1 6 7 2 6 11 15 14
228   1 6 7 2 6 11 15 16
229   1 6 7 2 6 11 12 16
230   1 6 7 2 6 11 12 8
231   1 6 7 2 6 11 12 11
```

Figura 2.1: Parte de los caminos creados para el jugador 1

```
 7    1 6 7 2 6 11 15 16
 8    1 6 7 2 3 8 12 16
 9    1 6 7 4 8 11 12 16
10    1 6 7 12 16 11 12 16
11    1 6 7 12 8 11 15 16
12    1 6 7 12 11 14 15 16
13    1 6 7 12 11 16 12 16
14    1 6 7 10 7 8 12 16
15    1 6 7 10 7 11 12 16
16    1 6 7 10 13 14 15 16
17    1 6 7 10 15 11 12 16
18    1 6 7 10 15 16 12 16
19    1 6 5 2 6 11 15 16
20    1 6 5 2 3 8 12 16
21    1 6 5 10 7 11 15 16
22    1 6 5 10 15 11 15 16
23    1 6 5 10 15 14 15 16
24    1 6 5 10 15 16 15 16
25    1 6 10 6 9 14 15 16
```

Figura 2.2: Parte de los caminos correctos encontrados para el jugador 1

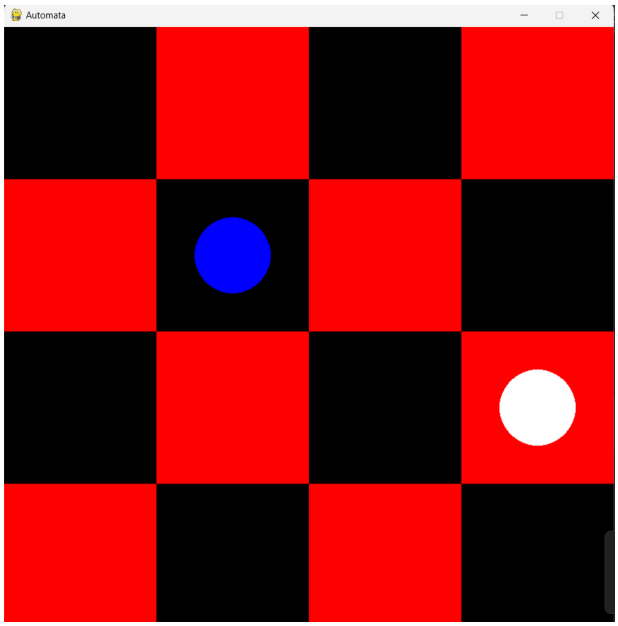

Figura 2.3: Animacion 1

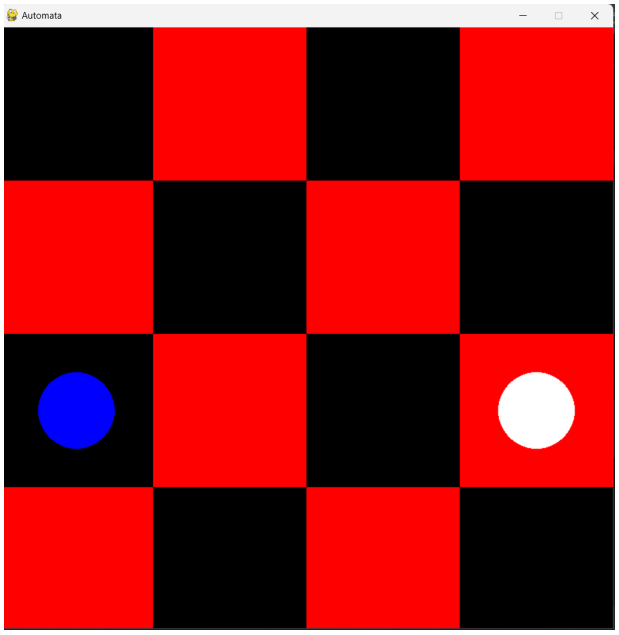

Figura 2.4: Animacion 2

2.6. Conclusiones

Durante el desarrollo de este programa, se realizó un repaso del uso de memoria dinámica en C, ya que la red fue creada de esta manera. Aunque se intentó inicialmente representar la red únicamente con archivos de texto, no se logró concretar una idea para su implementación. Aun así, el resultado fue satisfactorio, ya que incluso con una cadena de gran longitud, la memoria RAM no se llenó.

Un problema encontrado fue el tamaño del archivo de las soluciones, ya que después de una longitud de 30, este se volvió demasiado grande. Durante el desarrollo de la animación, se adquirió experiencia en la integración del funcionamiento del programa a nivel de código con las animaciones.

En resumen, este código sirvió para consolidar varios conocimientos, como el uso de memoria dinámica, el manejo de archivos y la animación.

Capítulo 3

Programa 3. Buscador de palabras

3.1. Descripción del problema

Se programará un autómata que reconozca todas las palabras del conjunto escuela, estudiantes, rencor, rifles, crimen, matanza. El proceso implica diseñar un NFA y transformarlo a DFA, mostrando todos los cálculos en el reporte, así como realizar la conversión a DFA, detallando todo el proceso a través de los subconjuntos y tablas. El programa debe ser capaz de leer un archivo de texto (o de internet) y utilizar el autómata para identificar cada palabra reservada, contabilizarlas e indicar dónde las encontró en el archivo, registrando su posición (x,y). Además, se debe imprimir en un archivo la evaluación del autómata por cada carácter leído, mostrando el cambio de estado y toda la historia del proceso. Por último, se requiere graficar el DFA.

3.2. Propuesta de solución

A partir del NFA generado, se realizó la conversión a DFA utilizando una tabla en Excel. Para la programación del DFA, se implementó un switch case, donde cada caso representa un nodo del autómata. Los registros de los estados se imprimieron en un archivo de texto junto con los resultados obtenidos.

3.3. Metodología

3.3.1. Conversión a NFA

El diseño del DFA se realizó a mano en OneNote. Se propusó que la misma ´´e" sirviera para inciar estudiantes y escuela, también la ´´r" inicializa dos palabras. Asi reduciendo, el número de nodos.

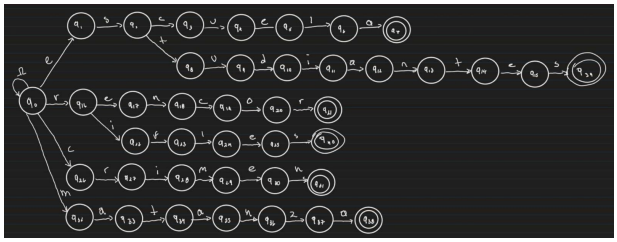

Figura 3.1: Autómata DFA

Depués se realizó la tabla de conversión en una hoja de excel. Y se obtuvieron 40 nodos.

		e	s	c	t	u	l	a	d	i	n	r	o	f	m	z
0	q0	q0,q1	q0	q0,q26	q0	q0	q0	q0	q0	q0	q0	q0,q16	q0	q0	q0,q32	q0
1	q0,q1	q0,q1	q0,q2	q0,q26	q0	q0	q0	q0	q0	q0	q0	q0,q16	q0	q0	q0,q32	q0
2	q0,q2	q0,q1	q0	q0,q26,q3	q0,q8	q0	q0	q0	q0	q0	q0	q0,q16	q0	q0	q0,q32	q0
3	q0,q26,q3	q0,q1	q0	q0,q26	q0	q0,q4	q0	q0	q0	q0	q0	q0,q16,q27	q0	q0	q0,q32	q0
4	q0,q4	q0,q1,q5	q0	q0,q26	q0	q0	q0	q0	q0	q0	q0	q0,q16	q0	q0	q0,q32	q0
5	q0,q1,q5	q0,q1	q0,q2	q0,q26	q0	q0	q0,q6	q0	q0	q0	q0	q0,q16	q0	q0	q0,q32	q0
6	q0,q6	q0,q1	q0	q0,q26	q0	q0	q0	q0,q7	q0	q0	q0	q0,q16	q0	q0	q0,q32	q0
7	q0,q8	q0,q1	q0	q0,q26	q0	q0,q9	q0	q0	q0	q0	q0	q0,q16	q0	q0	q0,q32	q0
8	q0,q9	q0,q1	q0	q0,q26	q0	q0	q0	q0	q0,q10	q0	q0	q0,q16	q0	q0	q0,q32	q0
9	q0,q10	q0,q1	q0	q0,q26	q0	q0	q0	q0	q0	q0,q11	q0	q0,q16	q0	q0	q0,q32	q0
10	q0,q11	q0,q1	q0	q0,q26	q0	q0	q0	q0,q12	q0	q0	q0	q0,q16	q0	q0	q0,q32	q0
11	q0,q12	q0,q1	q0	q0,q26	q0	q0	q0	q0	q0	q0	q0,q13	q0,q16	q0	q0	q0,q32	q0
12	q0,q13	q0,q1	q0	q0,q26	q0,q14	q0	q0	q0	q0	q0	q0	q0,q16	q0	q0	q0,q32	q0
13	q0,q14	q0,q1,q15	q0	q0,q26	q0	q0	q0	q0	q0	q0	q0	q0,q16	q0	q0	q0,q32	q0
14	q0,q16	q0,q1,q17	q0	q0,q26	q0	q0	q0	q0	q0	q0,q22	q0	q0,q16	q0	q0	q0,q32	q0
15	q0,q1,q17	q0,q1	q0,q2	q0,q26	q0	q0	q0	q0	q0	q0	q0,q18	q0,q16	q0	q0	q0,q32	q0
16	q0,q18	q0,q1	q0	q0,q26,q19	q0	q0	q0	q0	q0	q0	q0	q0,q16	q0	q0	q0,q32	q0
17	q0,q26,q19	q0,q1	q0	q0,q26	q0	q0	q0	q0	q0	q0	q0	q0,q16,q27	q0,q20	q0	q0,q32	q0
18	q0,q20	q0,q1	q0	q0,q26	q0	q0	q0	q0	q0	q0	q0	q0,q16,q21	q0	q0	q0,q32	q0
19	q0,q22	q0,q1	q0	q0,q26	q0	q0	q0	q0	q0	q0	q0	q0,q16	q0	q0,q23	q0,q32	q0
20	q0,q23	q0,q1	q0	q0,q26	q0	q0	q0	q0	q0	q0,q24	q0	q0,q16	q0	q0	q0,q32	q0
21	q0,q26	q0,q1	q0	q0,q26	q0	q0	q0	q0	q0	q0	q0	q0,q16,q27	q0	q0	q0,q32	q0
22	q0,q16,q27	q0,q1,q17	q0	q0,q26	q0	q0	q0	q0	q0	q0,q22,q28	q0	q0,q16	q0	q0	q0,q32	q0
23	q0,q22,q28	q0,q1	q0	q0,q26	q0	q0	q0	q0	q0	q0	q0	q0,q16	q0	q0,q23	q0,q32,q29	q0
24	q0,q32,q29	q0,q1,q30	q0	q0,q26	q0	q0	q0	q0,q33	q0	q0	q0	q0,q16	q0	q0	q0,q32	q0
25	q0,q1,q30	q0,q1	q0,q2	q0,q26	q0	q0	q0	q0	q0	q0	q0,q31	q0,q16	q0	q0	q0,q32	q0
26	q0,q32	q0,q1	q0	q0,q26	q0	q0	q0	q0,q33	q0	q0	q0	q0,q16	q0	q0	q0,q32	q0
27	q0,q33	q0,q1	q0	q0,q26	q0,q34	q0	q0	q0	q0	q0	q0	q0,q16	q0	q0	q0,q32	q0
28	q0,q34	q0,q1	q0	q0,q26	q0	q0	q0	q0,q35	q0	q0	q0	q0,q16	q0	q0	q0,q32	q0
29	q0,q35	q0,q1	q0	q0,q26	q0	q0	q0	q0	q0	q0	q0,q36	q0,q16	q0	q0	q0,q32	q0
30	q0,q36	q0,q1	q0	q0,q26	q0	q0	q0	q0	q0	q0	q0	q0,q16	q0	q0	q0,q32	q0,q37
31	q0,q37	q0,q2	q1	q0,q27	q0	q0	q0	q0,q38	q0	q0	q0	q0,q17	q0	q0	q0,q32	q0
32	q0,q1,q15	q0,q1	q0,q2,q39	q0,q26	q0	q0	q0	q0	q0	q0	q0	q0,q16	q0	q0	q0,q32	q0
33	q0,q24	q0,q1,q25	q0	q0,q26	q0	q0	q0	q0	q0	q0	q0	q0,q16	q0	q0	q0,q32	q0
34	q0,q1,q25	q0,q1	q0,q2,q40	q0,q26	q0	q0	q0	q0	q0	q0	q0	q0,q16	q0	q0	q0,q32	q0
35	q0,q7	q0,q1	q0	q0,q26	q0	q0	q0	q0	q0	q0	q0	q0,q16	q0	q0	q0,q32	q0
36	q0,q16,q21	q0,q1,q17	q0	q0,q26	q0	q0	q0	q0	q0	q0,q22	q0	q0,q16	q0	q0	q0,q32	q0
37	q0,q31	q0,q1	q0	q0,q26	q0	q0	q0	q0	q0	q0	q0	q0,q16	q0	q0	q0,q32	q0
38	q0,q2,q39	q0,q1	q0	q0,q26,q3	q0,q8	q0	q0	q0	q0	q0	q0	q0,q16	q0	q0	q0,q32	q0
39	q0,q38	q0,q1	q0	q0,q26	q0	q0	q0	q0	q0	q0	q0	q0,q16	q0	q0	q0,q32	q0
40	q0,q2,q40	q0,q1	q0	q0,q26,q3	q0,q8	q0	q0	q0	q0	q0	q0	q0,q16	q0	q0	q0,q32	q0

Figura 3.2: Tabla de conversión DFA a NFA

3.3.2. Codificación del NFA

- Se declararon tres archivos de texto:
  - `texto`: Contiene el texto a analizar.
  - `registro`: Almacena el historial de nodos y caracteres recibidos.
  - `resultados`: Guarda el número de palabras encontradas y su ubicación en el texto.
- Se tienen varios enteros para contar diferentes estadísticas:
  - `fila`: Sirve para rastrear la fila donde se encuentra el puntero en el texto.

- `columna`: Sirve para rastrear la columna donde se encuentra el puntero en el texto.
- `escuela`: Número de veces que se encontró la palabra "escuela".
- `estudiantes`: Número de veces que se encontró la palabra "estudiantes".
- `rencor`: Número de veces que se encontró la palabra ´´rencor".
- `rifles`: Número de veces que se encontró la palabra ´´rifles".
- `crimen`: Número de veces que se encontró la palabra ´´crimen".
- `matanza`: Número de veces que se encontró la palabra "matanza".

```
1       FILE *texto = fopen("texto_generado.txt","r");
2       FILE *registro = fopen("registro.txt","w");
3       FILE *resultados = fopen("resultados.txt","w");
4       int fila = 0;
5       int columna = 0;
6       int escuela = 0;
7       int estudiantes = 0;
8       int rencor = 0;
9       int rifles = 0;
10      int crimen = 0;
11      int matanza = 0;
12
```

Para manejar los nodos, se declaró un entero llamado `caso`. Este puede tomar un valor del 0 al 40 y se inicializa en 0 al principio. Además, se obtiene el primer carácter del texto y se convierte a minúsculas para generalizar el autómata. También se imprime el estado actual y el carácter recibido.

```
1       int caso = 0;
2       char c = fgetc(texto);
3       c = tolower(c);
4       fprintf(registro,"Edo actual: %d\tCar recibido: %c\n",caso,c);
```

Para leer todo el archivo se crea un ciclo que no termina hasta que el carácter recibido sea `EOF`. Lo primero es aumentar el contador de columna con cada carácter recibido, a menos que sea un salto de línea; en ese caso, se incrementa el contador de fila y se reinicia el contador de columna.

```
1       while (c!=EOF) {
2           if (c != '\n'){
3               columna++;
4           }
5           else if (c == '\n'){
6               columna = 0;
7               fila++;
8           }
```

Para representar las conexiones se utiliza un `switch case` con la variable `caso`. Luego se implementa la lógica del NFA mediante condicionales.

```
        switch (caso)
        {
        case 0:
            if (c == 'e'){
                caso = 1;
            }
            else if (c == 'c')
            {
                caso = 21;
            }
            else if (c == 'r')
            {
                caso = 14;
            }
            else if (c == 'm')
            {
                caso = 26;
            }
            else{
                caso = 0;
            }

            break;
```

Finalmente cada que se mueve a un caso final, se incrementa el contador correspondiente y se imprime la posicion actual en el archivo junto con la palabra.

```
        case 6:
            if (c == 'e'){
                caso = 1;
            }
            else if (c == 'c')
            {
                caso = 21;
            }
            else if (c == 'r')
            {
                caso = 14;
            }
            else if (c == 'm')
            {
                caso = 26;
```

```
16              }
17              else if (c == 'a'){
18                  fprintf(resultados," Palabra 'escuela' en: Fila: %d, columna: %d\n",
19                  escuela++;
20                  caso = 0;
21              }
22              else{
23                  caso = 0;
24              }
25              break;
```

En cada repetición se imprime el caso actual y el nuevo carácter recibido en el archivo de texto `registro`.

```
1       c = fgetc(texto);
2       c = tolower(c);
3       fprintf(registro,"Edo actual: %d\tCar recibido: %c\n",caso,c);
4
```

Para finalizar, se imprimen las frecuencias de las palabras en el texto.

```
1   printf("escuela: %d\n", escuela);
2   printf("estudiantes: %d\n", estudiantes);
3   printf("rencor: %d\n", rencor);
4   printf("rifles: %d\n", rifles);
5   printf("crimen: %d\n", crimen);
6   printf("matanza: %d\n", matanza);
```

3.3.3. Dibujo del NFA

Para el dibujo, se utilizó la biblioteca Pygame en Python. Dado que hay muchas conexiones, se decidió asignar un color a cada letra que representa un movimiento. El formato resultante es el siguiente:

```
1   BLACK = (0, 0, 0) #Omega
2   RED = (255, 0, 0) #z
3   BLUE = (0,0,255) #e
4   Verde = (0, 255, 0) #m
5   Amarillo = (255, 255, 0) #t
6   Naranja = (255, 165, 0) #l
7   Violeta = (128, 0, 128) #d
8   Ros = (255, 192, 203) #u
9   Turquesa = (64, 224, 208) #c
10  Gris = (128, 128, 128) #a
11  Marron = (139, 69, 19) #s
```

```
12 Celeste = (0, 255, 255) #o
13 Magenta = (255, 0, 255) #r
14 Verde_lima = (0, 255, 0) #i
15 Verde_oliva = (128, 128, 0)
16 Cyan = (0, 255, 255) #o
17 Lavanda = (230, 230, 250) #n
18 Coral = (255, 127, 80) #f
```

3.4. Resultados.

```
Edo actual: 0   Car recibido: e
Edo actual: 1   Car recibido: n
Edo actual: 0   Car recibido:
Edo actual: 0   Car recibido: u
Edo actual: 0   Car recibido: n
Edo actual: 0   Car recibido: a
Edo actual: 0   Car recibido:
Edo actual: 0   Car recibido: p
Edo actual: 0   Car recibido: e
Edo actual: 1   Car recibido: q
Edo actual: 0   Car recibido: u
Edo actual: 0   Car recibido: e
Edo actual: 1   Car recibido: Ã
Edo actual: 0   Car recibido: ±
Edo actual: 0   Car recibido: a
Edo actual: 0   Car recibido:
Edo actual: 0   Car recibido: l
Edo actual: 0   Car recibido: o
Edo actual: 0   Car recibido: c
Edo actual: 21  Car recibido: a
Edo actual: 0   Car recibido: l
Edo actual: 0   Car recibido: i
Edo actual: 0   Car recibido: d
Edo actual: 0   Car recibido: a
Edo actual: 0   Car recibido: d
Edo actual: 0   Car recibido: ,
```

Figura 3.3: Ejemplo de registro del autómata

```
escuela: 7
estudiantes: 5
rencor: 3
rifles: 2
crimen: 7
matanza: 4
```

Figura 3.4: Ejemplo de frecuencias encontradas en un texto

```
Palabra 'escuela' en: Fila: 0, columna: 37
Palabra 'estudiantes' en: Fila: 0, columna: 144
Palabra 'crimen' en: Fila: 0, columna: 268
Palabra 'matanza' en: Fila: 0, columna: 294
Palabra 'rifles' en: Fila: 2, columna: 42
Palabra 'rencor' en: Fila: 2, columna: 85
 Palabra 'escuela' en: Fila: 2, columna: 189
Palabra 'estudiantes' en: Fila: 2, columna: 206
Palabra 'crimen' en: Fila: 2, columna: 307
 Palabra 'escuela' en: Fila: 4, columna: 133
Palabra 'crimen' en: Fila: 4, columna: 202
Palabra 'estudiantes' en: Fila: 4, columna: 257
Palabra 'rencor' en: Fila: 4, columna: 381
Palabra 'crimen' en: Fila: 6, columna: 11
Palabra 'matanza' en: Fila: 6, columna: 25
 Palabra 'escuela' en: Fila: 6, columna: 169
Palabra 'rifles' en: Fila: 6, columna: 182
 Palabra 'escuela' en: Fila: 8, columna: 10
Palabra 'crimen' en: Fila: 8, columna: 99
Palabra 'estudiantes' en: Fila: 8, columna: 132
Palabra 'matanza' en: Fila: 8, columna: 254
Palabra 'crimen' en: Fila: 8, columna: 450
Palabra 'crimen' en: Fila: 10, columna: 127
Palabra 'matanza' en: Fila: 10, columna: 149
 Palabra 'escuela' en: Fila: 10, columna: 163
Palabra 'rencor' en: Fila: 10, columna: 257
```

Figura 3.5: Ejemplo de posiciones encontradas en un texto

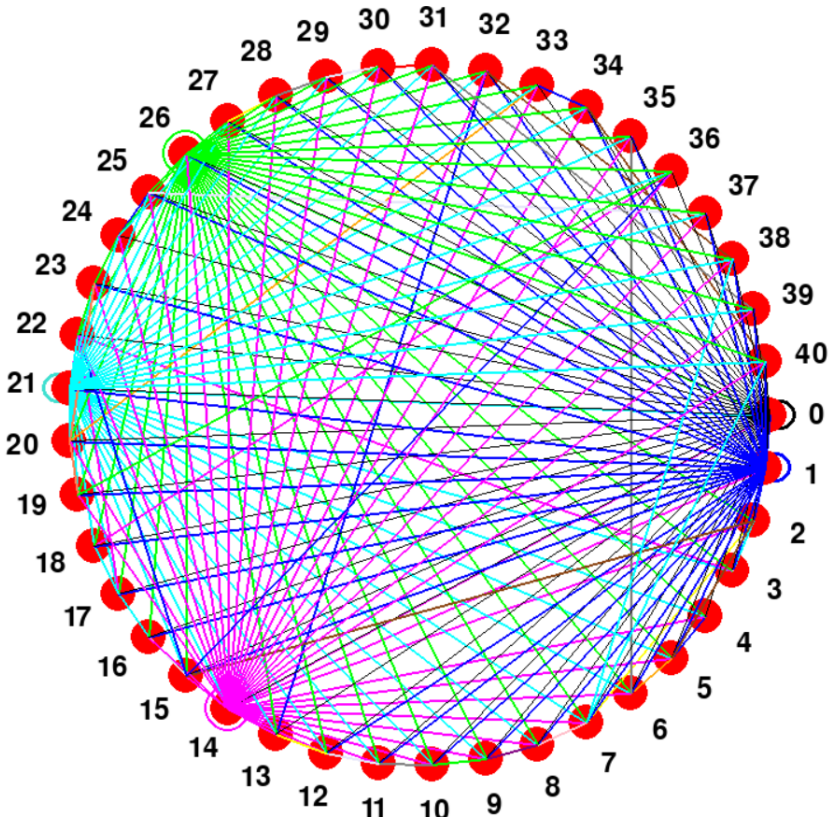

Figura 3.6: Dibujo del NFA

3.5. Conclusiones

En la realización de este programa se reafirmaron los conocimientos sobre cómo realizar la conversión de DFA a NFA. Además, se programó por primera vez un autómata NFA en el lenguaje C, y se vio el uso del switch case como una buena opción para implementar el funcionamiento del autómata. El programa desarrollado resulta útil para encontrar frecuencias de palabras que podrían levantar sospechas en foros y redes sociales, previniendo posibles ataques.

Capítulo 4

Anexo

4.1. Código del universo

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

int main() {
    printf("Bienvenido al Universo de cadenas\n");
    int repetcion= 1;
    while (repetcion!=2){
        printf("Hasta que numero desea observar el universo?\nSi imgresa un 0, se se

        int validacion = 0;
        int num = 0;
        scanf("%d", &validacion);
        if (validacion == 0){
            srand(time(NULL));
            int numeroAleatorio = rand() % 15;
            num=numeroAleatorio;
        }
        else{
            num = validacion;
        }
        printf("El numero recibido es %d\n", num);

        FILE *base = fopen("base.txt", "w");
        // Validación
        if (base == NULL) {
            perror("Error al crear el archivo base");
            return 1;
```

```
29          }
30          fprintf(base, "#,.,");
31          fclose(base);
32
33          for (int i=1; i<num;i++){
34
35              FILE *base_mod = fopen("base.txt", "r");
36              FILE *ceros = fopen("ceros.txt", "w+");
37
38              // Validación
39              if (base_mod == NULL || ceros == NULL) {
40                  perror("Error al abrir los archivos");
41                  return 1;
42              }
43
44              // Leer caracteres uno por uno de base.txt
45              int c;
46              while ((c = fgetc(base_mod)) != EOF) {
47                  // Si encontramos una coma, escribimos # en ceros.txt
48                  if (c == ',') {
49                      fprintf(ceros, "#,");
50                  }
51                  // Si no, escribir el caracter original en ceros.txt
52                  else {
53                      fputc(c, ceros);
54                  }
55              }
56
57              //Apuntamos al inicio del archivo
58              fseek(base_mod,0,SEEK_SET);
59
60              FILE *unos = fopen("unos.txt", "w+");
61              // Validación
62              if (base_mod == NULL || unos == NULL) {
63                  perror("Error al abrir los archivos");
64                  return 1;
65              }
66              // Realizamos el mismo procedimiento pero ahora agregando un .
67              while ((c = fgetc(base_mod)) != EOF) {
68                  if (c == ',') {
69                      fprintf(unos, ".,");
70                  } else {
71                      fputc(c, unos);
72                  }
73              }
```

```
74              fclose(base_mod);
75
76              char caracter;
77
78              // Abrir el archivo de salida para escritura
79              FILE* base_mod_reinicio = fopen("base.txt", "w");
80              fprintf(base_mod_reinicio,"#,.,");
81
82              // Verificar si el archivo de salida se abrió correctamente
83              if (base_mod_reinicio == NULL) {
84                  printf("No se pudo crear el archivo de salida.\n");
85                  exit(1);
86              }
87
88              fseek(unos,0,SEEK_SET);
89              fseek(ceros,0,SEEK_SET);
90              // Leer y escribir el contenido de los archivos originales
91              while ((caracter = fgetc(ceros)) != EOF) {
92              fputc(caracter, base_mod_reinicio);
93                  if (caracter == ',') {
94                      while ((caracter = fgetc(unos)) != EOF){
95                          putc(caracter, base_mod_reinicio);
96                          if (caracter == ','){
97                              break;
98                          }
99                      }
100                 }
101             }
102
103             // Cerrar los archivos
104             fclose(base_mod_reinicio);
105             fclose(unos);
106             fclose(ceros);
107
108         }
109         printf("¿Desea calcular otra n?\n1.Si\n2.No\n");
110         scanf("%d",&repetcion);
111     }
112     return 0;
113 }
114
```

```
1 %Declaracion  de variables
2 eje_y = readmatrix('contador.csv')
```

```
3
4 eje_x = 1:length(eje_y)
5 fig = figure;
6
7 plot(eje_x, eje_y, '.');
8 ylim([0 30])
9 title('Repeticiones de unos');
10 xlabel('Index');
11 ylabel('Unos');
12 set(fig, 'Position', [100, 100, 1000, 1000]);
13
14 saveas(fig, 'grafico_normal.png');
```

4.2. Código del juego

```
1
```

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <stdbool.h>
4  #include <string.h>
5
6  #define MAX_CAMINOS 1000
7  #define MAX_CAMINO_LEN 100
8  #define MAX_ID 10000
9
10
11 // Definición de la estructura de la capa, aqui se almacenan los obj correspondientes
12 typedef struct Capa {
13     int num_objetos;
14     struct Objeto **objetos;
15 } Capa;
16
17 // Definición de la estructura del objeto, cada objeto representa un numero, se puede
18 typedef struct Objeto {
19     int id;
20     struct Capa *capa;
21     struct Objeto **conexiones;
22     int num_conexiones;
23 } Objeto;
24
25 // Función para crear un nuevo objeto, se instancia el numero y la capa
26 Objeto* crear_objeto(int id, Capa *capa) {
```

```
27      Objeto *obj = (Objeto*)malloc(sizeof(Objeto));
28      obj->id = id;
29      obj->capa = capa;
30      obj->conexiones = NULL;
31      obj->num_conexiones = 0;
32      return obj;
33  }
34
35  // Función para conectar dos objetos, asi se forman los caminos
36  void conectar_objetos(Objeto *obj1, Objeto *obj2) {
37      // Aumentar el tamaño del array de conexiones de obj1 para incluir una nueva cone
38      obj1->conexiones = (Objeto**)realloc(obj1->conexiones, (obj1->num_conexiones + 1)
39      if (obj1->conexiones == NULL) {
40          printf("Error al realocar memoria para las conexiones\n");
41          exit(1); // O manejar el error de manera más apropiada
42      }
43
44      // Añadir obj2 al array de conexiones de obj1
45      obj1->conexiones[obj1->num_conexiones] = obj2;
46      obj1->num_conexiones++; // Incrementar el contador de conexiones
47  }
48
49
50  // Función para liberar la memoria utilizada por un objeto
51  void liberar_objeto(Objeto *obj) {
52      free(obj->conexiones);
53      free(obj);
54  }
55
56  void verificar_o_crear_y_conectar(Objeto* objeto_actual, int idObjetivo, Capa** capas
57      bool existe = false;
58      for (int k = 0; k < capas[capaSiguiente]->num_objetos; k++) {
59          if (capas[capaSiguiente]->objetos[k]->id == idObjetivo) {
60              conectar_objetos(objeto_actual, capas[capaSiguiente]->objetos[k]);
61              existe = true;
62              break;
63          }
64      }
65      if (!existe) {
66          Objeto* nuevo_objeto = crear_objeto(idObjetivo, capas[capaSiguiente]);
67          if (capas[capaSiguiente]->num_objetos == 0) {
68              capas[capaSiguiente]->objetos = (Objeto**)malloc(sizeof(Objeto*));
69          } else {
70              capas[capaSiguiente]->objetos = (Objeto**)realloc(capas[capaSiguiente]->o
71          }
```

```
72          capas[capaSiguiente]->objetos[*cont_obj] = nuevo_objeto;
73          capas[capaSiguiente]->num_objetos++;
74          (*cont_obj)++;
75          conectar_objetos(objeto_actual, nuevo_objeto);
76      }
77  }
78
79  void dfs(Objeto *obj, char *camino, int profundidad, int *contador, FILE* caminos, in
80      char idStr[12];
81      sprintf(idStr, "%d ", obj->id);
82
83      if (strlen(camino) + strlen(idStr) < MAX_CAMINO_LEN - 1) {
84          strcat(camino, idStr);
85      } else {
86          printf("Error: Camino excede longitud máxima.\n");
87          return;
88      }
89
90      // Si el objeto es de la última capa, guardamos el camino
91      if (obj->capa == capas[num_capas ]) {
92          fprintf(caminos,"%s\n",camino);
93      } else {
94          for (int i = 0; i < obj->num_conexiones; i++) {
95              char newCamino[MAX_CAMINO_LEN];
96              strcpy(newCamino, camino); // Copiamos el camino actual para evitar alte
97              dfs(obj->conexiones[i], newCamino, profundidad + 1, contador, caminos, nu
98          }
99      }
100 }
101
102 void encontrar_caminos(Capa **capas, int num_capas, FILE*caminos) {
103     int contador = 0;
104     fseek(caminos,0,SEEK_SET);
105     // Iniciar DFS desde la primera capa
106     for (int i = 0; i < capas[0]->num_objetos; i++) {
107         char camino[MAX_CAMINO_LEN] = {0}; // Inicializa el camino
108         dfs(capas[0]->objetos[i], camino, 0, &contador, caminos,num_capas ,capas);
109     }
110 }
111
112 void caminos_correctos(int num_capas, FILE*caminos, FILE*camino_correcto, char* final
113     fseek(caminos, 0, SEEK_SET);
114     char linea[MAX_CAMINO_LEN] = {0};
115     while(fgets(linea, MAX_CAMINO_LEN, caminos) != NULL){
116          fgets(linea, MAX_CAMINO_LEN, caminos);
```

```
117             int longitud = strlen(linea);
118             char seis = linea[longitud - 3];
119             char uno = linea[longitud - 4];
120             if (uno==final[0] && seis==final[1]){
121                fprintf(camino_correcto,linea);
122             }
123         }
124     }
125
126
127     void crearCaminos(FILE* cadena,FILE*caminos,FILE* camino_correcto,int inicio,char* fi
128
129         int numero_capas = 0;
130         char c = fgetc(cadena);
131         while (c != EOF) {
132             numero_capas++;
133             // fputc(c, caminos); // Copiar el carácter a caminos.txt si es necesario
134             c = fgetc(cadena);
135         }
136         printf("Numero de capas: %d\n", numero_capas);
137
138         numero_capas++;
139
140         Capa **capas = (Capa **)malloc(numero_capas * sizeof(Capa *));
141
142         if (capas == NULL) {
143             printf("Error al asignar memoria para capas\n");
144         }
145
146         // Inicialización de cada capa
147         for (int i = 0; i < numero_capas; i++) {
148             capas[i] = (Capa *)malloc(sizeof(Capa));
149             capas[i]->num_objetos = 0; // Inicializa el número de objetos en 0
150             capas[i]->objetos = NULL; // Inicializa la lista de objetos como NULL
151         }
152
153         Capa *primera_capa = capas[0];
154         Objeto *objeto = crear_objeto(inicio, primera_capa);
155         primera_capa->objetos = (Objeto **)malloc(sizeof(Objeto *));
156         primera_capa->objetos[0] = objeto;
157         primera_capa->num_objetos = 1;
158         fseek(cadena, 0, SEEK_SET);
159         numero_capas--;
160         for (int i=0; i<=numero_capas; i++){
161             c = fgetc(cadena);
```

```
162             if(c==EOF){
163                 break;
164             }
165             printf("%c\n",c);
166             printf("Rep num: %d\n",i);
167             Capa* capa_actual=capas[i];
168             int cont_obj=0;
169             for(int j=0;j<capa_actual->num_objetos;j++){
170                 Objeto *objetos_actual =capa_actual->objetos[j];
171                 printf("Objeto actual %d\n",capa_actual->objetos[j]->id);
172 
173                 if (objetos_actual->id == 1 && c == 'b') {
174                     verificar_o_crear_y_conectar(objetos_actual, 6, capas, i + 1, &cont_
175                 }
176 
177                 if (objetos_actual->id == 1 && c == 'r') {
178                     verificar_o_crear_y_conectar(objetos_actual, 2, capas, i + 1, &cont_
179                     verificar_o_crear_y_conectar(objetos_actual, 5, capas, i + 1, &cont_
180                 }
181 
182                 if (objetos_actual->id == 2 && c == 'b') {
183                     verificar_o_crear_y_conectar(objetos_actual, 6, capas, i + 1, &cont_
184                     verificar_o_crear_y_conectar(objetos_actual, 1, capas, i + 1, &cont_
185                     verificar_o_crear_y_conectar(objetos_actual, 3, capas, i + 1, &cont_
186                 }
187                 if (objetos_actual->id == 2 && c == 'r') {
188                     verificar_o_crear_y_conectar(objetos_actual, 5, capas, i + 1, &cont_
189                     verificar_o_crear_y_conectar(objetos_actual, 7, capas, i + 1, &cont_
190                 }
191 
192                 if (objetos_actual->id == 3 && c == 'b') {
193                     verificar_o_crear_y_conectar(objetos_actual, 6, capas, i + 1, &cont_
194                     verificar_o_crear_y_conectar(objetos_actual, 8, capas, i + 1, &cont_
195                 }
196                 if (objetos_actual->id == 3 && c == 'r') {
197                     verificar_o_crear_y_conectar(objetos_actual, 2, capas, i + 1, &cont_
198                     verificar_o_crear_y_conectar(objetos_actual, 7, capas, i + 1, &cont_
199                     verificar_o_crear_y_conectar(objetos_actual, 4, capas, i + 1, &cont_
200                 }
201 
202                 if (objetos_actual->id == 4 && c == 'b') {
203                     verificar_o_crear_y_conectar(objetos_actual, 3, capas, i + 1, &cont_
204                     verificar_o_crear_y_conectar(objetos_actual, 8, capas, i + 1, &cont_
205                 }
206                 if (objetos_actual->id == 4 && c == 'r') {
```

```
207                 verificar_o_crear_y_conectar(objetos_actual, 7, capas, i + 1, &cont_
208             }
209
210             if (objetos_actual->id == 5 && c == 'b') {
211                 verificar_o_crear_y_conectar(objetos_actual, 6, capas, i + 1, &cont_
212                 verificar_o_crear_y_conectar(objetos_actual, 1, capas, i + 1, &cont_
213                 verificar_o_crear_y_conectar(objetos_actual, 9, capas, i + 1, &cont_
214             }
215             if (objetos_actual->id == 5 && c == 'r') {
216                 verificar_o_crear_y_conectar(objetos_actual, 2, capas, i + 1, &cont_
217                 verificar_o_crear_y_conectar(objetos_actual, 10, capas, i + 1, &cont
218             }
219
220             if (objetos_actual->id == 6 && c == 'b') {
221                 verificar_o_crear_y_conectar(objetos_actual, 1, capas, i + 1, &cont_
222                 verificar_o_crear_y_conectar(objetos_actual, 3, capas, i + 1, &cont_
223                 verificar_o_crear_y_conectar(objetos_actual, 9, capas, i + 1, &cont_
224                 verificar_o_crear_y_conectar(objetos_actual, 11, capas, i + 1, &cont
225             }
226             if (objetos_actual->id == 6 && c == 'r') {
227                 verificar_o_crear_y_conectar(objetos_actual, 2, capas, i + 1, &cont_
228                 verificar_o_crear_y_conectar(objetos_actual, 7, capas, i + 1, &cont_
229                 verificar_o_crear_y_conectar(objetos_actual, 5, capas, i + 1, &cont_
230                 verificar_o_crear_y_conectar(objetos_actual, 10, capas, i + 1, &cont
231             }
232
233             if (objetos_actual->id == 7 && c == 'b') {
234                 verificar_o_crear_y_conectar(objetos_actual, 6, capas, i + 1, &cont_
235                 verificar_o_crear_y_conectar(objetos_actual, 3, capas, i + 1, &cont_
236                 verificar_o_crear_y_conectar(objetos_actual, 8, capas, i + 1, &cont_
237                 verificar_o_crear_y_conectar(objetos_actual, 11, capas, i + 1, &cont
238             }
239             if (objetos_actual->id == 7 && c == 'r') {
240                 verificar_o_crear_y_conectar(objetos_actual, 2, capas, i + 1, &cont_
241                 verificar_o_crear_y_conectar(objetos_actual, 4, capas, i + 1, &cont_
242                 verificar_o_crear_y_conectar(objetos_actual, 12, capas, i + 1, &cont
243                 verificar_o_crear_y_conectar(objetos_actual, 10, capas, i + 1, &cont
244             }
245
246             if (objetos_actual->id == 8 && c == 'b') {
247                 verificar_o_crear_y_conectar(objetos_actual, 3, capas, i + 1, &cont_
248                 verificar_o_crear_y_conectar(objetos_actual, 11, capas, i + 1, &cont
249             }
250             if (objetos_actual->id == 8 && c == 'r') {
251                 verificar_o_crear_y_conectar(objetos_actual, 4, capas, i + 1, &cont_
```

```
252                 verificar_o_crear_y_conectar(objetos_actual, 7, capas, i + 1, &cont_
253                 verificar_o_crear_y_conectar(objetos_actual, 12, capas, i + 1, &cont
254             }
255
256
257             if (objetos_actual->id == 9 && c == 'b') {
258                 verificar_o_crear_y_conectar(objetos_actual, 6, capas, i + 1, &cont_
259                 verificar_o_crear_y_conectar(objetos_actual, 14, capas, i + 1, &cont
260             }
261             if (objetos_actual->id == 9 && c == 'r') {
262                 verificar_o_crear_y_conectar(objetos_actual, 5, capas, i + 1, &cont_
263                 verificar_o_crear_y_conectar(objetos_actual, 13, capas, i + 1, &cont
264                 verificar_o_crear_y_conectar(objetos_actual, 10, capas, i + 1, &cont
265             }
266
267
268             if (objetos_actual->id == 10 && c == 'b') {
269                 verificar_o_crear_y_conectar(objetos_actual, 5, capas, i + 1, &cont_
270                 verificar_o_crear_y_conectar(objetos_actual, 7, capas, i + 1, &cont_
271                 verificar_o_crear_y_conectar(objetos_actual, 13, capas, i + 1, &cont
272                 verificar_o_crear_y_conectar(objetos_actual, 15, capas, i + 1, &cont
273             }
274             if (objetos_actual->id == 10 && c == 'r') {
275                 verificar_o_crear_y_conectar(objetos_actual, 6, capas, i + 1, &cont_
276                 verificar_o_crear_y_conectar(objetos_actual, 9, capas, i + 1, &cont_
277                 verificar_o_crear_y_conectar(objetos_actual, 11, capas, i + 1, &cont
278                 verificar_o_crear_y_conectar(objetos_actual, 14, capas, i + 1, &cont
279             }
280
281
282             if (objetos_actual->id == 11 && c == 'b') {
283                 verificar_o_crear_y_conectar(objetos_actual, 6, capas, i + 1, &cont_
284                 verificar_o_crear_y_conectar(objetos_actual, 14, capas, i + 1, &cont
285                 verificar_o_crear_y_conectar(objetos_actual, 8, capas, i + 1, &cont_
286                 verificar_o_crear_y_conectar(objetos_actual, 16, capas, i + 1, &cont
287             }
288             if (objetos_actual->id == 11 && c == 'r') {
289                 verificar_o_crear_y_conectar(objetos_actual, 7, capas, i + 1, &cont_
290                 verificar_o_crear_y_conectar(objetos_actual, 15, capas, i + 1, &cont
291                 verificar_o_crear_y_conectar(objetos_actual, 12, capas, i + 1, &cont
292                 verificar_o_crear_y_conectar(objetos_actual, 10, capas, i + 1, &cont
293             }
294
295
296             if (objetos_actual->id == 12 && c == 'b') {
```

```
297                 verificar_o_crear_y_conectar(objetos_actual, 16, capas, i + 1, &cont
298                 verificar_o_crear_y_conectar(objetos_actual, 8, capas, i + 1, &cont_
299                 verificar_o_crear_y_conectar(objetos_actual, 11, capas, i + 1, &cont
300             }
301             if (objetos_actual->id == 12 && c == 'r') {
302                 verificar_o_crear_y_conectar(objetos_actual, 7, capas, i + 1, &cont_
303                 verificar_o_crear_y_conectar(objetos_actual, 15, capas, i + 1, &cont
304             }
305
306
307             if (objetos_actual->id == 13 && c == 'b') {
308                 verificar_o_crear_y_conectar(objetos_actual, 9, capas, i + 1, &cont_
309                 verificar_o_crear_y_conectar(objetos_actual, 14, capas, i + 1, &cont
310             }
311             if (objetos_actual->id == 13 && c == 'r') {
312                 verificar_o_crear_y_conectar(objetos_actual, 10, capas, i + 1, &cont
313             }
314
315
316             if (objetos_actual->id == 14 && c == 'b') {
317                 verificar_o_crear_y_conectar(objetos_actual, 9, capas, i + 1, &cont_
318                 verificar_o_crear_y_conectar(objetos_actual, 11, capas, i + 1, &cont
319             }
320             if (objetos_actual->id == 14 && c == 'r') {
321                 verificar_o_crear_y_conectar(objetos_actual, 13, capas, i + 1, &cont
322                 verificar_o_crear_y_conectar(objetos_actual, 10, capas, i + 1, &cont
323                 verificar_o_crear_y_conectar(objetos_actual, 15, capas, i + 1, &cont
324             }
325
326
327             if (objetos_actual->id == 15 && c == 'b') {
328                 verificar_o_crear_y_conectar(objetos_actual, 11, capas, i + 1, &cont
329                 verificar_o_crear_y_conectar(objetos_actual, 14, capas, i + 1, &cont
330                 verificar_o_crear_y_conectar(objetos_actual, 16, capas, i + 1, &cont
331             }
332             if (objetos_actual->id == 15 && c == 'r') {
333                 verificar_o_crear_y_conectar(objetos_actual, 10, capas, i + 1, &cont
334                 verificar_o_crear_y_conectar(objetos_actual, 12, capas, i + 1, &cont
335             }
336
337
338             if (objetos_actual->id == 16 && c == 'b') {
339                 verificar_o_crear_y_conectar(objetos_actual, 11, capas, i + 1, &cont
340             }
341             if (objetos_actual->id == 16 && c == 'r') {
```

```
342                 verificar_o_crear_y_conectar(objetos_actual, 12, capas, i + 1, &cont
343                 verificar_o_crear_y_conectar(objetos_actual, 15, capas, i + 1, &cont
344             }
345 
346         }
347 
348     }
349 
350     encontrar_caminos(capas,numero_capas,caminos);
351 
352     caminos_correctos(numero_capas,caminos,camino_correcto,final);
353 
354     for (int i = 0; i < numero_capas; i++) {
355         for (int j = 0; j < capas[i]->num_objetos; j++) {
356             liberar_objeto(capas[i]->objetos[j]);
357         }
358         free(capas[i]->objetos);
359         free(capas[i]);
360     }
361 
362     free(capas);
363 }
364 
365 int main() {
366     // Archivos para el primer agente
367     FILE* camino_agente = fopen("caminos.txt", "w+");
368     FILE* cadena_agente = fopen("cadena1.txt", "r");
369     FILE* camino_correcto_a1 = fopen("caminos_correctos.txt", "w");
370 
371     // Verificar si los archivos se abrieron correctamente
372     if (camino_agente == NULL || cadena_agente == NULL || camino_correcto_a1 == NULL
373         printf("Error al abrir los archivos para el primer agente.\n");
374         return 1; // Terminar el programa con código de error
375     }
376 
377     int inicio_agente = 1;
378     char final_agente[2]= {'1','6'};
379     crearCaminos(cadena_agente, camino_agente, camino_correcto_a1, inicio_agente, fi
380     fclose(camino_agente);
381     fclose(cadena_agente);
382     fclose(camino_correcto_a1);
383 
384     // Archivos para el segundo agente
385     FILE* camino_agente2 = fopen("caminos2.txt", "w+");
386     FILE* cadena_agente2 = fopen("cadena2.txt", "r");
```

```
387     FILE* camino_correcto_a2 = fopen("caminos_correctos2.txt", "w");
388 
389     // Verificar si los archivos se abrieron correctamente
390     if (camino_agente2 == NULL || cadena_agente2 == NULL || camino_correcto_a2 == NU
391         printf("Error al abrir los archivos para el segundo agente.\n");
392         // Cerrar los archivos abiertos previamente
393         fclose(camino_agente);
394         fclose(cadena_agente);
395         fclose(camino_correcto_a1);
396         return 1; // Terminar el programa con código de error
397     }
398 
399     int inicio_agente2 = 4;
400     char final_agente2[2]= {'1','3'};
401     crearCaminos(cadena_agente2, camino_agente2, camino_correcto_a2, inicio_agente2,
402 
403     // Cerrar los archivos abiertos
404 
405     fclose(camino_agente2);
406     fclose(cadena_agente2);
407     fclose(camino_correcto_a2);
408 
409     return 0; // Terminación exitosa
410 }
```

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <stdbool.h>
4  #include <string.h>
5 
6  #define MAX_CAMINOS 1000
7  #define MAX_CAMINO_LEN 100
8  #define MAX_ID 10000
9 
10 
11 // Definición de la estructura de la capa, aqui se almacenan los obj correspondientes
12 typedef struct Capa {
13     int num_objetos;
14     struct Objeto **objetos;
15 } Capa;
16 
17 // Definición de la estructura del objeto, cada objeto representa un numero, se puede
18 typedef struct Objeto {
19     int id;
20     struct Capa *capa;
```

```
21      struct Objeto **conexiones;
22      int num_conexiones;
23  } Objeto;
24
25  // Función para crear un nuevo objeto, se instancia el numero y la capa
26  Objeto* crear_objeto(int id, Capa *capa) {
27      Objeto *obj = (Objeto*)malloc(sizeof(Objeto));
28      obj->id = id;
29      obj->capa = capa;
30      obj->conexiones = NULL;
31      obj->num_conexiones = 0;
32      return obj;
33  }
34
35  // Función para conectar dos objetos, asi se forman los caminos
36  void conectar_objetos(Objeto *obj1, Objeto *obj2) {
37      // Aumentar el tamaño del array de conexiones de obj1 para incluir una nueva cone
38      obj1->conexiones = (Objeto**)realloc(obj1->conexiones, (obj1->num_conexiones + 1)
39      if (obj1->conexiones == NULL) {
40          printf("Error al realocar memoria para las conexiones\n");
41          exit(1); // O manejar el error de manera más apropiada
42      }
43
44      // Añadir obj2 al array de conexiones de obj1
45      obj1->conexiones[obj1->num_conexiones] = obj2;
46      obj1->num_conexiones++; // Incrementar el contador de conexiones
47  }
48
49
50  // Función para liberar la memoria utilizada por un objeto
51  void liberar_objeto(Objeto *obj) {
52      free(obj->conexiones);
53      free(obj);
54  }
55
56  void verificar_o_crear_y_conectar(Objeto* objeto_actual, int idObjetivo, Capa** capas
57      bool existe = false;
58      for (int k = 0; k < capas[capaSiguiente]->num_objetos; k++) {
59          if (capas[capaSiguiente]->objetos[k]->id == idObjetivo) {
60              conectar_objetos(objeto_actual, capas[capaSiguiente]->objetos[k]);
61              existe = true;
62              break;
63          }
64      }
65      if (!existe) {
```

```
66          Objeto* nuevo_objeto = crear_objeto(idObjetivo, capas[capaSiguiente]);
67          if (capas[capaSiguiente]->num_objetos == 0) {
68              capas[capaSiguiente]->objetos = (Objeto**)malloc(sizeof(Objeto*));
69          } else {
70              capas[capaSiguiente]->objetos = (Objeto**)realloc(capas[capaSiguiente]->
71          }
72          capas[capaSiguiente]->objetos[*cont_obj] = nuevo_objeto;
73          capas[capaSiguiente]->num_objetos++;
74          (*cont_obj)++;
75          conectar_objetos(objeto_actual, nuevo_objeto);
76      }
77  }
78
79  void dfs(Objeto *obj, char *camino, int profundidad, int *contador, FILE* caminos, i
80      char idStr[12];
81      sprintf(idStr, "%d ", obj->id);
82
83      if (strlen(camino) + strlen(idStr) < MAX_CAMINO_LEN - 1) {
84          strcat(camino, idStr);
85      } else {
86          printf("Error: Camino excede longitud máxima.\n");
87          return;
88      }
89
90      // Si el objeto es de la última capa, guardamos el camino
91      if (obj->capa == capas[num_capas ]) {
92          fprintf(caminos,"%s\n",camino);
93      } else {
94          for (int i = 0; i < obj->num_conexiones; i++) {
95              char newCamino[MAX_CAMINO_LEN];
96              strcpy(newCamino, camino); // Copiamos el camino actual para evitar alte
97              dfs(obj->conexiones[i], newCamino, profundidad + 1, contador, caminos, nu
98          }
99      }
100 }
101
102 void encontrar_caminos(Capa **capas, int num_capas, FILE*caminos) {
103     int contador = 0;
104     fseek(caminos,0,SEEK_SET);
105     // Iniciar DFS desde la primera capa
106     for (int i = 0; i < capas[0]->num_objetos; i++) {
107         char camino[MAX_CAMINO_LEN] = {0}; // Inicializa el camino
108         dfs(capas[0]->objetos[i], camino, 0, &contador, caminos,num_capas ,capas);
109     }
110 }
```

```
111
112 void caminos_correctos(int num_capas, FILE*caminos, FILE*camino_correcto, char* final
113     fseek(caminos, 0, SEEK_SET);
114     char linea[MAX_CAMINO_LEN] = {0};
115     while(fgets(linea, MAX_CAMINO_LEN, caminos) != NULL){
116          fgets(linea, MAX_CAMINO_LEN, caminos);
117          int longitud = strlen(linea);
118          char seis = linea[longitud - 3];
119          char uno = linea[longitud - 4];
120          if (uno==final[0] && seis==final[1]){
121             fprintf(camino_correcto,linea);
122          }
123     }
124 }
125
126
127 void crearCaminos(FILE* cadena,FILE*caminos,FILE* camino_correcto,int inicio,char* fi
128
129     int numero_capas = 0;
130     char c = fgetc(cadena);
131     while (c != EOF) {
132         numero_capas++;
133         // fputc(c, caminos); // Copiar el carácter a caminos.txt si es necesario
134         c = fgetc(cadena);
135     }
136     printf("Numero de capas: %d\n", numero_capas);
137
138     numero_capas++;
139
140     Capa **capas = (Capa **)malloc(numero_capas * sizeof(Capa *));
141
142     if (capas == NULL) {
143         printf("Error al asignar memoria para capas\n");
144     }
145
146     // Inicialización de cada capa
147     for (int i = 0; i < numero_capas; i++) {
148         capas[i] = (Capa *)malloc(sizeof(Capa));
149         capas[i]->num_objetos = 0; // Inicializa el número de objetos en 0
150         capas[i]->objetos = NULL; // Inicializa la lista de objetos como NULL
151     }
152
153     Capa *primera_capa = capas[0];
154     Objeto *objeto = crear_objeto(inicio, primera_capa);
155     primera_capa->objetos = (Objeto **)malloc(sizeof(Objeto *));
```

```
156     primera_capa->objetos[0] = objeto;
157     primera_capa->num_objetos = 1;
158     fseek(cadena, 0, SEEK_SET);
159     numero_capas--;
160     for (int i=0; i<=numero_capas; i++){
161         c = fgetc(cadena);
162         if(c==EOF){
163             break;
164         }
165         printf("%c\n",c);
166         printf("Rep num: %d\n",i);
167         Capa* capa_actual=capas[i];
168         int cont_obj=0;
169         for(int j=0;j<capa_actual->num_objetos;j++){
170             Objeto *objetos_actual =capa_actual->objetos[j];
171             printf("Objeto actual %d\n",capa_actual->objetos[j]->id);
172 
173             if (objetos_actual->id == 1 && c == 'b') {
174                 verificar_o_crear_y_conectar(objetos_actual, 6, capas, i + 1, &cont_
175             }
176 
177             if (objetos_actual->id == 1 && c == 'r') {
178                 verificar_o_crear_y_conectar(objetos_actual, 2, capas, i + 1, &cont_
179                 verificar_o_crear_y_conectar(objetos_actual, 5, capas, i + 1, &cont_
180             }
181 
182             if (objetos_actual->id == 2 && c == 'b') {
183                 verificar_o_crear_y_conectar(objetos_actual, 6, capas, i + 1, &cont_
184                 verificar_o_crear_y_conectar(objetos_actual, 1, capas, i + 1, &cont_
185                 verificar_o_crear_y_conectar(objetos_actual, 3, capas, i + 1, &cont_
186             }
187             if (objetos_actual->id == 2 && c == 'r') {
188                 verificar_o_crear_y_conectar(objetos_actual, 5, capas, i + 1, &cont_
189                 verificar_o_crear_y_conectar(objetos_actual, 7, capas, i + 1, &cont_
190             }
191 
192             if (objetos_actual->id == 3 && c == 'b') {
193                 verificar_o_crear_y_conectar(objetos_actual, 6, capas, i + 1, &cont_
194                 verificar_o_crear_y_conectar(objetos_actual, 8, capas, i + 1, &cont_
195             }
196             if (objetos_actual->id == 3 && c == 'r') {
197                 verificar_o_crear_y_conectar(objetos_actual, 2, capas, i + 1, &cont_
198                 verificar_o_crear_y_conectar(objetos_actual, 7, capas, i + 1, &cont_
199                 verificar_o_crear_y_conectar(objetos_actual, 4, capas, i + 1, &cont_
200             }
```

```
201
202             if (objetos_actual->id == 4 && c == 'b') {
203                 verificar_o_crear_y_conectar(objetos_actual, 3, capas, i + 1, &cont_c
204                 verificar_o_crear_y_conectar(objetos_actual, 8, capas, i + 1, &cont_c
205             }
206             if (objetos_actual->id == 4 && c == 'r') {
207                 verificar_o_crear_y_conectar(objetos_actual, 7, capas, i + 1, &cont_c
208             }
209
210             if (objetos_actual->id == 5 && c == 'b') {
211                 verificar_o_crear_y_conectar(objetos_actual, 6, capas, i + 1, &cont_c
212                 verificar_o_crear_y_conectar(objetos_actual, 1, capas, i + 1, &cont_c
213                 verificar_o_crear_y_conectar(objetos_actual, 9, capas, i + 1, &cont_c
214             }
215             if (objetos_actual->id == 5 && c == 'r') {
216                 verificar_o_crear_y_conectar(objetos_actual, 2, capas, i + 1, &cont_c
217                 verificar_o_crear_y_conectar(objetos_actual, 10, capas, i + 1, &cont_
218             }
219
220             if (objetos_actual->id == 6 && c == 'b') {
221                 verificar_o_crear_y_conectar(objetos_actual, 1, capas, i + 1, &cont_c
222                 verificar_o_crear_y_conectar(objetos_actual, 3, capas, i + 1, &cont_c
223                 verificar_o_crear_y_conectar(objetos_actual, 9, capas, i + 1, &cont_c
224                 verificar_o_crear_y_conectar(objetos_actual, 11, capas, i + 1, &cont_
225             }
226             if (objetos_actual->id == 6 && c == 'r') {
227                 verificar_o_crear_y_conectar(objetos_actual, 2, capas, i + 1, &cont_c
228                 verificar_o_crear_y_conectar(objetos_actual, 7, capas, i + 1, &cont_c
229                 verificar_o_crear_y_conectar(objetos_actual, 5, capas, i + 1, &cont_c
230                 verificar_o_crear_y_conectar(objetos_actual, 10, capas, i + 1, &cont_
231             }
232
233             if (objetos_actual->id == 7 && c == 'b') {
234                 verificar_o_crear_y_conectar(objetos_actual, 6, capas, i + 1, &cont_c
235                 verificar_o_crear_y_conectar(objetos_actual, 3, capas, i + 1, &cont_c
236                 verificar_o_crear_y_conectar(objetos_actual, 8, capas, i + 1, &cont_c
237                 verificar_o_crear_y_conectar(objetos_actual, 11, capas, i + 1, &cont_
238             }
239             if (objetos_actual->id == 7 && c == 'r') {
240                 verificar_o_crear_y_conectar(objetos_actual, 2, capas, i + 1, &cont_c
241                 verificar_o_crear_y_conectar(objetos_actual, 4, capas, i + 1, &cont_c
242                 verificar_o_crear_y_conectar(objetos_actual, 12, capas, i + 1, &cont_
243                 verificar_o_crear_y_conectar(objetos_actual, 10, capas, i + 1, &cont_
244             }
245
```

```
246             if (objetos_actual->id == 8 && c == 'b') {
247                 verificar_o_crear_y_conectar(objetos_actual, 3, capas, i + 1, &cont_c
248                 verificar_o_crear_y_conectar(objetos_actual, 11, capas, i + 1, &cont_
249             }
250             if (objetos_actual->id == 8 && c == 'r') {
251                 verificar_o_crear_y_conectar(objetos_actual, 4, capas, i + 1, &cont_c
252                 verificar_o_crear_y_conectar(objetos_actual, 7, capas, i + 1, &cont_c
253                 verificar_o_crear_y_conectar(objetos_actual, 12, capas, i + 1, &cont_
254             }
255
256
257             if (objetos_actual->id == 9 && c == 'b') {
258                 verificar_o_crear_y_conectar(objetos_actual, 6, capas, i + 1, &cont_c
259                 verificar_o_crear_y_conectar(objetos_actual, 14, capas, i + 1, &cont_
260             }
261             if (objetos_actual->id == 9 && c == 'r') {
262                 verificar_o_crear_y_conectar(objetos_actual, 5, capas, i + 1, &cont_c
263                 verificar_o_crear_y_conectar(objetos_actual, 13, capas, i + 1, &cont_
264                 verificar_o_crear_y_conectar(objetos_actual, 10, capas, i + 1, &cont_
265             }
266
267
268             if (objetos_actual->id == 10 && c == 'b') {
269                 verificar_o_crear_y_conectar(objetos_actual, 5, capas, i + 1, &cont_c
270                 verificar_o_crear_y_conectar(objetos_actual, 7, capas, i + 1, &cont_c
271                 verificar_o_crear_y_conectar(objetos_actual, 13, capas, i + 1, &cont_
272                 verificar_o_crear_y_conectar(objetos_actual, 15, capas, i + 1, &cont_
273             }
274             if (objetos_actual->id == 10 && c == 'r') {
275                 verificar_o_crear_y_conectar(objetos_actual, 6, capas, i + 1, &cont_c
276                 verificar_o_crear_y_conectar(objetos_actual, 9, capas, i + 1, &cont_c
277                 verificar_o_crear_y_conectar(objetos_actual, 11, capas, i + 1, &cont_
278                 verificar_o_crear_y_conectar(objetos_actual, 14, capas, i + 1, &cont_
279             }
280
281
282             if (objetos_actual->id == 11 && c == 'b') {
283                 verificar_o_crear_y_conectar(objetos_actual, 6, capas, i + 1, &cont_c
284                 verificar_o_crear_y_conectar(objetos_actual, 14, capas, i + 1, &cont_
285                 verificar_o_crear_y_conectar(objetos_actual, 8, capas, i + 1, &cont_c
286                 verificar_o_crear_y_conectar(objetos_actual, 16, capas, i + 1, &cont_
287             }
288             if (objetos_actual->id == 11 && c == 'r') {
289                 verificar_o_crear_y_conectar(objetos_actual, 7, capas, i + 1, &cont_c
290                 verificar_o_crear_y_conectar(objetos_actual, 15, capas, i + 1, &cont_
```

```
291                 verificar_o_crear_y_conectar(objetos_actual, 12, capas, i + 1, &cont
292                 verificar_o_crear_y_conectar(objetos_actual, 10, capas, i + 1, &cont
293             }
294
295
296             if (objetos_actual->id == 12 && c == 'b') {
297                 verificar_o_crear_y_conectar(objetos_actual, 16, capas, i + 1, &cont
298                 verificar_o_crear_y_conectar(objetos_actual, 8, capas, i + 1, &cont_
299                 verificar_o_crear_y_conectar(objetos_actual, 11, capas, i + 1, &cont
300             }
301             if (objetos_actual->id == 12 && c == 'r') {
302                 verificar_o_crear_y_conectar(objetos_actual, 7, capas, i + 1, &cont_
303                 verificar_o_crear_y_conectar(objetos_actual, 15, capas, i + 1, &cont
304             }
305
306
307             if (objetos_actual->id == 13 && c == 'b') {
308                 verificar_o_crear_y_conectar(objetos_actual, 9, capas, i + 1, &cont_
309                 verificar_o_crear_y_conectar(objetos_actual, 14, capas, i + 1, &cont
310             }
311             if (objetos_actual->id == 13 && c == 'r') {
312                 verificar_o_crear_y_conectar(objetos_actual, 10, capas, i + 1, &cont
313             }
314
315
316             if (objetos_actual->id == 14 && c == 'b') {
317                 verificar_o_crear_y_conectar(objetos_actual, 9, capas, i + 1, &cont_
318                 verificar_o_crear_y_conectar(objetos_actual, 11, capas, i + 1, &cont
319             }
320             if (objetos_actual->id == 14 && c == 'r') {
321                 verificar_o_crear_y_conectar(objetos_actual, 13, capas, i + 1, &cont
322                 verificar_o_crear_y_conectar(objetos_actual, 10, capas, i + 1, &cont
323                 verificar_o_crear_y_conectar(objetos_actual, 15, capas, i + 1, &cont
324             }
325
326
327             if (objetos_actual->id == 15 && c == 'b') {
328                 verificar_o_crear_y_conectar(objetos_actual, 11, capas, i + 1, &cont
329                 verificar_o_crear_y_conectar(objetos_actual, 14, capas, i + 1, &cont
330                 verificar_o_crear_y_conectar(objetos_actual, 16, capas, i + 1, &cont
331             }
332             if (objetos_actual->id == 15 && c == 'r') {
333                 verificar_o_crear_y_conectar(objetos_actual, 10, capas, i + 1, &cont
334                 verificar_o_crear_y_conectar(objetos_actual, 12, capas, i + 1, &cont
335             }
```

```
336
337
338             if (objetos_actual->id == 16 && c == 'b') {
339                 verificar_o_crear_y_conectar(objetos_actual, 11, capas, i + 1, &cont
340             }
341             if (objetos_actual->id == 16 && c == 'r') {
342                 verificar_o_crear_y_conectar(objetos_actual, 12, capas, i + 1, &cont
343                 verificar_o_crear_y_conectar(objetos_actual, 15, capas, i + 1, &cont
344             }
345
346         }
347
348     }
349
350     encontrar_caminos(capas,numero_capas,caminos);
351
352     caminos_correctos(numero_capas,caminos,camino_correcto,final);
353
354     for (int i = 0; i < numero_capas; i++) {
355         for (int j = 0; j < capas[i]->num_objetos; j++) {
356             liberar_objeto(capas[i]->objetos[j]);
357         }
358         free(capas[i]->objetos);
359         free(capas[i]);
360     }
361
362     free(capas);
363 }
364
365 int main() {
366     // Archivos para el primer agente
367     FILE* camino_agente = fopen("caminos.txt", "w+");
368     FILE* cadena_agente = fopen("cadena1.txt", "r");
369     FILE* camino_correcto_a1 = fopen("caminos_correctos.txt", "w");
370
371     // Verificar si los archivos se abrieron correctamente
372     if (camino_agente == NULL || cadena_agente == NULL || camino_correcto_a1 == NULL
373         printf("Error al abrir los archivos para el primer agente.\n");
374         return 1; // Terminar el programa con código de error
375     }
376
377     int inicio_agente = 1;
378     char final_agente[2]= {'1','6'};
379     crearCaminos(cadena_agente, camino_agente, camino_correcto_a1, inicio_agente, fi
380     fclose(camino_agente);
```

```
381     fclose(cadena_agente);
382     fclose(camino_correcto_a1);
383 
384     // Archivos para el segundo agente
385     FILE* camino_agente2 = fopen("caminos2.txt", "w+");
386     FILE* cadena_agente2 = fopen("cadena2.txt", "r");
387     FILE* camino_correcto_a2 = fopen("caminos_correctos2.txt", "w");
388 
389     // Verificar si los archivos se abrieron correctamente
390     if (camino_agente2 == NULL || cadena_agente2 == NULL || camino_correcto_a2 == NU
391         printf("Error al abrir los archivos para el segundo agente.\n");
392         // Cerrar los archivos abiertos previamente
393         fclose(camino_agente);
394         fclose(cadena_agente);
395         fclose(camino_correcto_a1);
396         return 1; // Terminar el programa con código de error
397     }
398 
399     int inicio_agente2 = 4;
400     char final_agente2[2]= {'1','3'};
401     crearCaminos(cadena_agente2, camino_agente2, camino_correcto_a2, inicio_agente2,
402 
403     // Cerrar los archivos abiertos
404 
405     fclose(camino_agente2);
406     fclose(cadena_agente2);
407     fclose(camino_correcto_a2);
408 
409     return 0; // Terminación exitosa
410 }
```

4.3. Código del NFA

```
1  #include <stdio.h>
2  #include <ctype.h>
3 
4  int main(){
5      FILE *texto = fopen("texto_generado.txt","r");
6      FILE *registro = fopen("registro.txt","w");
7      FILE *resultados = fopen("resultados.txt","w");
8      char c = fgetc(texto);
9      c = tolower(c);
10     int fila = 0;
```

```
11      int columna = 0;
12      int escuela = 0;
13      int estudiantes = 0;
14      int rencor = 0;
15      int rifles = 0;
16      int crimen = 0;
17      int matanza = 0;
18
19      int caso = 0;
20      fprintf(registro,"Edo actual: %d\tCar recibido: %c\n",caso,c);
21      while (c!=EOF) {
22          if (c != '\n'){
23              columna++;
24          }
25          else if (c == '\n'){
26              columna = 0;
27              fila++;
28          }
29          switch (caso)
30          {
31          case 0:
32              if (c == 'e'){
33                  caso = 1;
34              }
35              else if (c == 'c')
36              {
37                  caso = 21;
38              }
39              else if (c == 'r')
40              {
41                  caso = 14;
42              }
43              else if (c == 'm')
44              {
45                  caso = 26;
46              }
47              else{
48                  caso = 0;
49              }
50
51              break;
52          case 1:
53              if (c == 'e'){
54                  caso = 1;
55              }
```

```
56              else if(c == 's'){
57                  caso = 2;
58              }
59              else if (c == 'c')
60              {
61                  caso = 21;
62              }
63              else if (c == 'r')
64              {
65                  caso = 14;
66              }
67              else if (c == 'm')
68              {
69                  caso = 26;
70              }
71              else{
72                  caso = 0;
73              }
74
75              break;
76          case 2:
77               if (c == 'e'){
78                  caso = 1;
79              }
80              else if (c == 'c')
81              {
82                  caso = 3;
83              }
84              else if (c == 'r')
85              {
86                  caso = 14;
87              }
88              else if (c == 't'){
89                  caso = 7;
90              }
91              else if (c == 'm')
92              {
93                  caso = 26;
94              }
95              else{
96                  caso = 0;
97              }
98              break;
99          case 3:
100             if (c == 'e'){
```

```
101                 caso = 1;
102             }
103             else if (c == 'c')
104             {
105                 caso = 21;
106             }
107             else if (c == 'u'){
108                 caso = 4;
109             }
110             else if (c == 'r')
111             {
112                 caso = 22;
113             }
114             else if (c == 'm')
115             {
116                 caso = 26;
117             }
118             else{
119                 caso = 0;
120             }
121             break;
122         case 4:
123             if (c == 'e'){
124                 caso = 5;
125             }
126             else if (c == 'c')
127             {
128                 caso = 21;
129             }
130             else if (c == 'r')
131             {
132                 caso = 14;
133             }
134             else if (c == 'm')
135             {
136                 caso = 26;
137             }
138             else{
139                 caso = 0;
140             }
141             break;
142         case 5:
143             if (c == 'e'){
144                 caso = 1;
145             }
```

```
146             else if (c == 's'){
147                 caso = 2;
148             }
149             else if (c == 'c')
150             {
151                 caso = 21;
152             }
153             else if (c == 'l'){
154                 caso = 6;
155             }
156             else if (c == 'r')
157             {
158                 caso = 14;
159             }
160             else if (c == 'm')
161             {
162                 caso = 26;
163             }
164             else{
165                 caso = 0;
166             }
167             break;
168         case 6:
169             if (c == 'e'){
170                 caso = 1;
171             }
172             else if (c == 'c')
173             {
174                 caso = 21;
175             }
176             else if (c == 'r')
177             {
178                 caso = 14;
179             }
180             else if (c == 'm')
181             {
182                 caso = 26;
183             }
184             else if (c == 'a'){
185                 fprintf(resultados," Palabra 'escuela' en: Fila: %d, columna: %d\n",
186                 escuela++;
187                 caso = 0;
188             }
189             else{
190                 caso = 0;
```

```
191             }
192             break;
193         case 7:
194             if (c == 'e'){
195                 caso = 1;
196             }
197             else if (c == 'c')
198             {
199                 caso = 21;
200             }
201             else if (c == 'u'){
202                 caso = 8;
203             }
204             else if (c == 'r')
205             {
206                 caso = 14;
207             }
208             else if (c == 'm')
209             {
210                 caso = 26;
211             }
212             else{
213                 caso = 0;
214             }
215             break;
216         case 8:
217             if (c == 'e'){
218                 caso = 1;
219             }
220             else if (c == 'c')
221             {
222                 caso = 21;
223             }
224             else if (c == 'd'){
225                 caso = 9;
226             }
227             else if (c == 'r')
228             {
229                 caso = 14;
230             }
231             else if (c == 'm')
232             {
233                 caso = 26;
234             }
235             else{
```

```
236                 caso = 0;
237             }
238             break;
239         case 9:
240             if (c == 'e'){
241                 caso = 1;
242             }
243             else if (c == 'c')
244             {
245                 caso = 21;
246             }
247             else if (c == 'i'){
248                 caso = 10;
249             }
250             else if (c == 'r')
251             {
252                 caso = 14;
253             }
254             else if (c == 'm')
255             {
256                 caso = 26;
257             }
258             else{
259                 caso = 0;
260             }
261             break;
262         case 10:
263             if (c == 'e'){
264                 caso = 1;
265             }
266             else if (c == 'c')
267             {
268                 caso = 21;
269             }
270             else if (c == 'a'){
271                 caso = 11;
272             }
273             else if (c == 'r')
274             {
275                 caso = 14;
276             }
277             else if (c == 'm')
278             {
279                 caso = 26;
280             }
```

```
281                 else{
282                     caso = 0;
283                 }
284                 break;
285             case 11:
286                 if (c == 'e'){
287                     caso = 1;
288                 }
289                 else if (c == 'c')
290                 {
291                     caso = 21;
292                 }
293                 else if (c == 'n'){
294                     caso = 12;
295                 }
296                 else if (c == 'r')
297                 {
298                     caso = 14;
299                 }
300                 else if (c == 'm')
301                 {
302                     caso = 26;
303                 }
304                 else{
305                     caso = 0;
306                 }
307                 break;
308             case 12:
309                 if (c == 'e'){
310                     caso = 1;
311                 }
312                 else if (c == 'c')
313                 {
314                     caso = 21;
315                 }
316                 else if (c == 't'){
317                     caso = 13;
318                 }
319                 else if (c == 'r')
320                 {
321                     caso = 14;
322                 }
323                 else if (c == 'm')
324                 {
325                     caso = 26;
```

```
326                 }
327                 else{
328                     caso = 0;
329                 }
330                 break;
331             case 13:
332                 if (c == 'e'){
333                     caso = 32;
334                 }
335                 else if (c == 'c')
336                 {
337                     caso = 21;
338                 }
339 
340                 else if (c == 'r')
341                 {
342                     caso = 14;
343                 }
344                 else if (c == 'm')
345                 {
346                     caso = 26;
347                 }
348                 else{
349                     caso = 0;
350                 }
351                 break;
352             case 32:
353                 if (c == 'e'){
354                     caso = 1;
355                 }
356                 else if (c == 's'){
357                     fprintf(resultados,"Palabra 'estudiantes' en: Fila: %d, columna: %d\
358                     estudiantes++;
359                     caso = 2;
360                 }
361                 else if (c == 'c')
362                 {
363                     caso = 21;
364                 }
365                 else if (c == 'r')
366                 {
367                     caso = 14;
368                 }
369                 else if (c == 'm')
370                 {
```

```
371                 caso = 26;
372             }
373             else{
374                 caso = 0;
375             }
376             break;
377         default:
378             break;
379         case 14:
380             if (c == 'e'){
381                 caso = 15;
382             }
383             else if (c == 'c')
384             {
385                 caso = 21;
386             }
387             else if (c == 'i'){
388                 caso = 19;
389             }
390 
391             else if (c == 'r')
392             {
393                 caso = 14;
394             }
395             else if (c == 'm')
396             {
397                 caso = 26;
398             }
399             else{
400                 caso = 0;
401             }
402             break;
403         case 15:
404             if (c == 'e'){
405                 caso = 1;
406             }
407             else if (c == 'c')
408             {
409                 caso = 21;
410             }
411             else if (c == 's'){
412                 caso = 2;
413             }
414             else if (c == 'n'){
415                 caso = 16;
```

```
416                 }
417                 else if (c == 'r')
418                 {
419                     caso = 14;
420                 }
421                 else if (c == 'm')
422                 {
423                     caso = 26;
424                 }
425                 else{
426                     caso = 0;
427                 }
428                 break;
429             case 16:
430                 if (c == 'e'){
431                     caso = 1;
432                 }
433                 else if (c == 'c')
434                 {
435                     caso = 17;
436                 }
437                 else if (c == 'r')
438                 {
439                     caso = 14;
440                 }
441                 else if (c == 'm')
442                 {
443                     caso = 26;
444                 }
445                 else{
446                     caso = 0;
447                 }
448                 break;
449             case 17:
450                 if (c == 'e'){
451                     caso = 1;
452                 }
453                 else if (c == 'c')
454                 {
455                     caso = 21;
456                 }
457                 else if (c == 'r')
458                 {
459                     caso = 22;
460                 }
```

```
461             else if (c == 'o'){
462                 caso = 18;
463             }
464             else if (c == 'm')
465             {
466                 caso = 26;
467             }
468             else{
469                 caso = 0;
470             }
471             break;
472         case 18:
473             if (c == 'e'){
474                 caso = 1;
475             }
476             else if (c == 'c')
477             {
478                 caso = 21;
479             }
480             else if (c == 'r')
481             {
482                 fprintf(resultados,"Palabra 'rencor' en: Fila: %d, columna: %d\n",fi
483                 rencor++;
484                 caso = 14;
485             }
486             else if (c == 'm')
487             {
488                 caso = 26;
489             }
490             else{
491                 caso = 0;
492             }
493             break;
494         case 19:
495             if (c == 'e'){
496                 caso = 1;
497             }
498             else if (c == 'c')
499             {
500                 caso = 21;
501             }
502             else if (c == 'f'){
503                 caso = 20;
504             }
505             else if (c == 'r')
```

```
506                 {
507                     caso = 14;
508                 }
509                 else if (c == 'm')
510                 {
511                     caso = 26;
512                 }
513                 else{
514                     caso = 0;
515                 }
516                 break;
517             case 20:
518                 if (c == 'e'){
519                     caso = 1;
520                 }
521                 else if (c == 'c')
522                 {
523                     caso = 21;
524                 }
525                 else if (c == 'r')
526                 {
527                     caso = 14;
528                 }
529                 else if (c == 'l'){
530                     caso = 33;
531                 }
532                 else if (c == 'm')
533                 {
534                     caso = 26;
535                 }
536                 else{
537                     caso = 0;
538                 }
539                 break;
540             case 33:
541                 if (c == 'e'){
542                     caso = 34;
543                 }
544                 else if (c == 'c')
545                 {
546                     caso = 21;
547                 }
548                 else if (c == 'r')
549                 {
550                     caso = 14;
```

```
551                 }
552                 else if (c == 'm')
553                 {
554                     caso = 26;
555                 }
556                 else{
557                     caso = 0;
558                 }
559                 break;
560             case 34:
561                 if (c == 'e'){
562                     caso = 1;
563                 }
564                 else if (c == 's'){
565                     fprintf(resultados,"Palabra 'rifles' en: Fila: %d, columna: %d\n",fi
566                     rifles++;
567                     caso = 2;
568                 }
569                 else if (c == 'c')
570                 {
571                     caso = 21;
572                 }
573                 else if (c == 'r')
574                 {
575                     caso = 14;
576                 }
577                 else if (c == 'm')
578                 {
579                     caso = 26;
580                 }
581                 else{
582                     caso = 0;
583                 }
584                 break;
585             case 21:
586                 if (c == 'e'){
587                     caso = 1;
588                 }
589                 else if (c == 'c')
590                 {
591                     caso = 21;
592                 }
593                 else if (c == 'r')
594                 {
595                     caso = 22;
```

```
596                 }
597                 else if (c == 'm')
598                 {
599                     caso = 26;
600                 }
601                 else{
602                     caso = 0;
603                 }
604                 break;
605             case 22:
606                 if (c == 'e'){
607                     caso = 15;
608                 }
609                 else if (c == 'i'){
610                     caso = 23;
611                 }
612                 else if (c == 'c')
613                 {
614                     caso = 21;
615                 }
616                 else if (c == 'r')
617                 {
618                     caso = 14;
619                 }
620                 else if (c == 'm')
621                 {
622                     caso = 26;
623                 }
624                 else{
625                     caso = 0;
626                 }
627                 break;
628             case 23:
629                 if (c == 'e'){
630                     caso = 1;
631                 }
632                 else if (c == 'c')
633                 {
634                     caso = 21;
635                 }
636                 else if (c == 'r')
637                 {
638                     caso = 14;
639                 }
640                 else if (c == 'f'){
```

```
641                 caso = 20;
642             }
643             else if (c == 'm')
644             {
645                 caso = 24;
646             }
647             else{
648                 caso = 0;
649             }
650             break;
651         case 24:
652             if (c == 'e'){
653                 caso = 25;
654             }
655             else if (c == 'c')
656             {
657                 caso = 21;
658             }
659             else if (c == 'r')
660             {
661                 caso = 14;
662             }
663             else if (c == 'm')
664             {
665                 caso = 26;
666             }
667             else if (c == 'a'){
668                 caso = 27;
669             }
670             else{
671                 caso = 0;
672             }
673             break;
674         case 25:
675             if (c == 'e'){
676                 caso = 1;
677             }
678             else if (c == 'c')
679             {
680                 caso = 21;
681             }
682             else if (c == 's'){
683                 caso = 2;
684             }
685             else if (c == 'n'){
```

```
686                 fprintf(resultados,"Palabra 'crimen' en: Fila: %d, columna: %d\n",fi
687                 crimen++;
688                 caso = 0;
689             }
690             else if (c == 'r')
691             {
692                 caso = 14;
693             }
694             else if (c == 'm')
695             {
696                 caso = 26;
697             }
698             else{
699                 caso = 0;
700             }
701             break;
702         case 26:
703             if (c == 'e'){
704                 caso = 1;
705             }
706             else if (c == 'c')
707             {
708                 caso = 21;
709             }
710             else if (c == 'a'){
711                 caso = 27;
712             }
713             else if (c == 'r')
714             {
715                 caso = 14;
716             }
717             else if (c == 'm')
718             {
719                 caso = 26;
720             }
721             else{
722                 caso = 0;
723             }
724             break;
725         case 27:
726             if (c == 'e'){
727                 caso = 1;
728             }
729             else if (c == 'c')
730             {
```

```
731                 caso = 21;
732             }
733             else if (c == 'r')
734             {
735                 caso = 14;
736             }
737             else if (c == 't'){
738                 caso = 28;
739             }
740             else if (c == 'm')
741             {
742                 caso = 26;
743             }
744             else{
745                 caso = 0;
746             }
747             break;
748         case 28:
749             if (c == 'e'){
750                 caso = 1;
751             }
752             else if (c == 'c')
753             {
754                 caso = 21;
755             }
756             else if (c == 'r')
757             {
758                 caso = 14;
759             }
760             else if (c == 'm')
761             {
762                 caso = 26;
763             }
764             else if (c == 'a'){
765                 caso = 29;
766             }
767             else{
768                 caso = 0;
769             }
770             break;
771         case 29:
772             if (c == 'e'){
773                 caso = 1;
774             }
775             else if (c == 'c')
```

```
776                 {
777                     caso = 21;
778                 }
779                 else if (c == 'r')
780                 {
781                     caso = 14;
782                 }
783                 else if (c == 'n'){
784                     caso = 30;
785                 }
786                 else if (c == 'm')
787                 {
788                     caso = 26;
789                 }
790                 else{
791                     caso = 0;
792                 }
793                 break;
794             case 30:
795                 if (c == 'e'){
796                     caso = 1;
797                 }
798                 else if (c == 'c')
799                 {
800                     caso = 21;
801                 }
802                 else if (c == 'z'){
803                     caso = 31;
804                 }
805                 else if (c == 'r')
806                 {
807                     caso = 14;
808                 }
809                 else if (c == 'm')
810                 {
811                     caso = 26;
812                 }
813                 else{
814                     caso = 0;
815                 }
816                 break;
817             case 31:
818                 if (c == 'e'){
819                     caso = 1;
820                 }
```

```
821             else if (c == 'a'){
822                 fprintf(resultados,"Palabra 'matanza' en: Fila: %d, columna: %d\n",f
823                 matanza++;
824                 caso = 0;
825             }
826             else if (c == 'c')
827             {
828                 caso = 21;
829             }
830             else if (c == 'r')
831             {
832                 caso = 14;
833             }
834             else if (c == 'm')
835             {
836                 caso = 26;
837             }
838             else{
839                 caso = 0;
840             }
841             break;
842 
843     }
844     c = fgetc(texto);
845     c = tolower(c);
846     fprintf(registro,"Edo actual: %d\tCar recibido: %c\n",caso,c);
847 }
848 fprintf(resultados,"escuela: %d\n", escuela);
849 fprintf(resultados,"estudiantes: %d\n", estudiantes);
850 fprintf(resultados,"rencor: %d\n", rencor);
851 fprintf(resultados,"rifles: %d\n", rifles);
852 fprintf(resultados,"crimen: %d\n", crimen);
853 fprintf(resultados,"matanza: %d\n", matanza);
854 fclose(texto);
855 fclose(registro);
856 fclose(resultados);
857 
858 return 0;
859 }
```

```
1 import pygame
2 import sys
3 import math
4 
5 
```

```
6
7  pygame.init()
8
9  # Dimensiones de la ventana
10 WIDTH = 900
11 HEIGHT = 700
12
13 # Colores
14 WHITE = (255, 255, 255)
15 BLACK = (0, 0, 0) #Universo
16 RED = (255, 0, 0) #z
17 BLUE = (0,0,255) #e
18
19 Verde = (0, 255, 0) #m
20 Amarillo = (255, 255, 0) #t
21 Naranja = (255, 165, 0) #l
22 Violeta = (128, 0, 128) #d
23 Ros = (255, 192, 203) #u
24 Turquesa = (64, 224, 208) #c
25 Gris = (128, 128, 128) #a
26 Marron = (139, 69, 19) #s
27 Celeste = (0, 255, 255) #o
28 Magenta = (255, 0, 255) #r
29 Verde_lima = (0, 255, 0) #i
30 Verde_oliva = (128, 128, 0)
31 Cyan = (0, 255, 255) #o
32 Lavanda = (230, 230, 250) #n
33 Coral = (255, 127, 80) #f
34
35
36 screen = pygame.display.set_mode((WIDTH, HEIGHT))
37 pygame.display.set_caption("Dibujar Circunferencias")
38
39 # Definir propiedades de la circunferencia
40 center = (WIDTH // 2, HEIGHT // 2)
41 main_radius = 300
42 main_thickness = 2
43
44 # Definir las propiedades de la circunferencia de los numeros
45 center_num = (WIDTH // 2, HEIGHT // 2)
46 main_radius_num = 340
47 main_thickness_num = 2
48
49 # Definir propiedades de los círculos pequeños
50 num_small_circles = 41
```

```
small_radius = 15

#Lista para almacenaar los centros
lista_centros = []

running = True
while running:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            running = False

    # Dibujar la circunferencia principal en la ventana
    screen.fill(WHITE)
    pygame.draw.circle(screen, WHITE, center, main_radius, main_thickness)

    pygame.draw.circle(screen, WHITE, center, main_radius_num, main_thickness_num)
    # Calcular el ángulo de separación entre los círculos pequeños
    angle_increment = 2 * math.pi / num_small_circles

    pygame.draw.circle(screen, BLACK, (760, 350), 13)
    pygame.draw.circle(screen, WHITE, (760, 350), 10)

    pygame.draw.circle(screen, BLUE, (756, 395), 13)
    pygame.draw.circle(screen, WHITE, (756, 395), 10)

    pygame.draw.circle(screen, Magenta, (286, 601), 20)
    pygame.draw.circle(screen, WHITE, (286, 601), 18)

    pygame.draw.circle(screen, Turquesa, (140, 327), 13)
    pygame.draw.circle(screen, WHITE, (140, 327), 10)

    pygame.draw.circle(screen, Verde, (250, 126), 20)
    pygame.draw.circle(screen, WHITE, (250, 126), 18)
    # Dibujar los círculos pequeños sobre la circunferencia principal
    for i in range(num_small_circles):
        # Calcular la posición del centro del círculo pequeño
        angle = i * angle_increment
        small_center_x = int(center[0] + main_radius * math.cos(angle))
        small_center_y = int(center[1] + main_radius * math.sin(angle))
        pygame.draw.circle(screen, RED, (small_center_x, small_center_y), small_radiu
        lista_centros.append((small_center_x, small_center_y))

    for i in range(num_small_circles):
        # Calcular la posición del centro del círculo pequeño
        angle = i * angle_increment
```

```
96          small_center_x = int(center[0] + main_radius_num * math.cos(angle))
97          small_center_y = int(center[1] + main_radius_num * math.sin(angle))
98          pygame.draw.circle(screen, WHITE, (small_center_x, small_center_y), small_rad
99
100         circle_center = (small_center_x,small_center_y)
101         # Configurar el texto
102         font = pygame.font.Font(None, 36)
103         text = font.render(str(i), True, BLACK)
104         text_rect = text.get_rect(center=circle_center)
105         # Dibujar el texto en la ventana
106         screen.blit(text, text_rect)
107
108
109
110     # Nodo 0
111     pygame.draw.line(screen, BLUE, lista_centros[0], lista_centros[1], 2)
112     pygame.draw.line(screen, Celeste, lista_centros[0], lista_centros[21], 2)
113     pygame.draw.line(screen, Magenta, lista_centros[0], lista_centros[14], 2)
114     pygame.draw.line(screen, Verde, lista_centros[0], lista_centros[26], 2)
115     # Nodo 1
116     pygame.draw.line(screen, BLUE, lista_centros[1], lista_centros[1], 2)
117     pygame.draw.line(screen, Celeste, lista_centros[1], lista_centros[21], 2)
118     pygame.draw.line(screen, Magenta, lista_centros[1], lista_centros[14], 2)
119     pygame.draw.line(screen, Verde, lista_centros[1], lista_centros[26], 2)
120     pygame.draw.line(screen, BLACK, lista_centros[1], lista_centros[0], 1)
121     pygame.draw.line(screen, Marron, lista_centros[1], lista_centros[2], 2)
122     #Nodo 2
123     pygame.draw.line(screen, BLUE, lista_centros[2], lista_centros[1], 2)
124     pygame.draw.line(screen, Celeste, lista_centros[2], lista_centros[3], 2)
125     pygame.draw.line(screen, Magenta, lista_centros[2], lista_centros[14], 2)
126     pygame.draw.line(screen, Verde, lista_centros[2], lista_centros[26], 2)
127     pygame.draw.line(screen, BLACK, lista_centros[2], lista_centros[0], 1)
128     pygame.draw.line(screen, Amarillo, lista_centros[2], lista_centros[7], 2)
129     #Nodo 3
130     pygame.draw.line(screen, BLUE, lista_centros[3], lista_centros[1], 2)
131     pygame.draw.line(screen, Celeste, lista_centros[3], lista_centros[21], 2)
132     pygame.draw.line(screen, Magenta, lista_centros[3], lista_centros[22], 2)
133     pygame.draw.line(screen, Verde, lista_centros[3], lista_centros[26], 2)
134     pygame.draw.line(screen, BLACK, lista_centros[3], lista_centros[0], 1)
135     pygame.draw.line(screen, Ros, lista_centros[3], lista_centros[3], 2)
136     #Nodo 4
137     pygame.draw.line(screen, BLUE, lista_centros[4], lista_centros[5], 2)
138     pygame.draw.line(screen, Celeste, lista_centros[4], lista_centros[21], 2)
139     pygame.draw.line(screen, Magenta, lista_centros[4], lista_centros[14], 2)
140     pygame.draw.line(screen, Verde, lista_centros[4], lista_centros[26], 2)
```

```
141         pygame.draw.line(screen, BLACK, lista_centros[4], lista_centros[0], 1)
142         #Nodo 5
143         pygame.draw.line(screen, BLUE, lista_centros[5], lista_centros[1], 2)
144         pygame.draw.line(screen, Celeste, lista_centros[5], lista_centros[21], 2)
145         pygame.draw.line(screen, Magenta, lista_centros[5], lista_centros[14], 2)
146         pygame.draw.line(screen, Verde, lista_centros[5], lista_centros[26], 2)
147         pygame.draw.line(screen, BLACK, lista_centros[5], lista_centros[0], 1)
148         pygame.draw.line(screen, Marron, lista_centros[5], lista_centros[2], 2)
149         pygame.draw.line(screen, Naranja, lista_centros[5], lista_centros[6], 2)
150         #Nodo 6
151         pygame.draw.line(screen, BLUE, lista_centros[6], lista_centros[1], 2)
152         pygame.draw.line(screen, Celeste, lista_centros[6], lista_centros[21], 2)
153         pygame.draw.line(screen, Magenta, lista_centros[6], lista_centros[14], 2)
154         pygame.draw.line(screen, Verde, lista_centros[6], lista_centros[26], 2)
155         pygame.draw.line(screen, BLACK, lista_centros[6], lista_centros[0], 1)
156         pygame.draw.line(screen, Gris, lista_centros[6], lista_centros[35], 2)
157         #Nodo 7
158         pygame.draw.line(screen, BLUE, lista_centros[7], lista_centros[1], 2)
159         pygame.draw.line(screen, Celeste, lista_centros[7], lista_centros[21], 2)
160         pygame.draw.line(screen, Magenta, lista_centros[7], lista_centros[14], 2)
161         pygame.draw.line(screen, Verde, lista_centros[7], lista_centros[26], 2)
162         pygame.draw.line(screen, BLACK, lista_centros[7], lista_centros[0], 1)
163         pygame.draw.line(screen, Ros, lista_centros[7], lista_centros[8], 2)
164         #Nodo 8
165         pygame.draw.line(screen, BLUE, lista_centros[8], lista_centros[1], 2)
166         pygame.draw.line(screen, Celeste, lista_centros[8], lista_centros[21], 2)
167         pygame.draw.line(screen, Magenta, lista_centros[8], lista_centros[14], 2)
168         pygame.draw.line(screen, Verde, lista_centros[8], lista_centros[26], 2)
169         pygame.draw.line(screen, BLACK, lista_centros[8], lista_centros[0], 1)
170         pygame.draw.line(screen, Violeta, lista_centros[8], lista_centros[9], 2)
171         #Nodo 9
172         pygame.draw.line(screen, BLUE, lista_centros[9], lista_centros[1], 2)
173         pygame.draw.line(screen, Celeste, lista_centros[9], lista_centros[21], 2)
174         pygame.draw.line(screen, Magenta, lista_centros[9], lista_centros[14], 2)
175         pygame.draw.line(screen, Verde, lista_centros[9], lista_centros[26], 2)
176         pygame.draw.line(screen, BLACK, lista_centros[9], lista_centros[0], 1)
177         pygame.draw.line(screen, Verde_lima, lista_centros[9], lista_centros[10], 2)
178         #Nodo 10
179         pygame.draw.line(screen, BLUE, lista_centros[10], lista_centros[1], 2)
180         pygame.draw.line(screen, Celeste, lista_centros[10], lista_centros[21], 2)
181         pygame.draw.line(screen, Magenta, lista_centros[10], lista_centros[14], 2)
182         pygame.draw.line(screen, Verde, lista_centros[10], lista_centros[26], 2)
183         pygame.draw.line(screen, BLACK, lista_centros[10], lista_centros[0], 1)
184         pygame.draw.line(screen, Gris, lista_centros[10], lista_centros[11], 2)
185         #Nodo 11
```

```
186         pygame.draw.line(screen, BLUE, lista_centros[11], lista_centros[1], 2)
187         pygame.draw.line(screen, Celeste, lista_centros[11], lista_centros[21], 2)
188         pygame.draw.line(screen, Magenta, lista_centros[11], lista_centros[14], 2)
189         pygame.draw.line(screen, Verde, lista_centros[11], lista_centros[26], 2)
190         pygame.draw.line(screen, BLACK, lista_centros[11], lista_centros[0], 1)
191         pygame.draw.line(screen, Lavanda, lista_centros[11], lista_centros[12], 2)
192         #Nodo 12
193         pygame.draw.line(screen, BLUE, lista_centros[12], lista_centros[1], 2)
194         pygame.draw.line(screen, Celeste, lista_centros[12], lista_centros[21], 2)
195         pygame.draw.line(screen, Magenta, lista_centros[12], lista_centros[14], 2)
196         pygame.draw.line(screen, Verde, lista_centros[12], lista_centros[26], 2)
197         pygame.draw.line(screen, BLACK, lista_centros[12], lista_centros[0], 1)
198         pygame.draw.line(screen, Amarillo, lista_centros[12], lista_centros[13], 2)
199         #Nodo 13
200         pygame.draw.line(screen, BLUE, lista_centros[13], lista_centros[32], 2)
201         pygame.draw.line(screen, Celeste, lista_centros[13], lista_centros[21], 2)
202         pygame.draw.line(screen, Magenta, lista_centros[13], lista_centros[14], 2)
203         pygame.draw.line(screen, Verde, lista_centros[13], lista_centros[26], 2)
204         pygame.draw.line(screen, BLACK, lista_centros[13], lista_centros[0], 1)
205         #Nodo 32
206         pygame.draw.line(screen, BLUE, lista_centros[32], lista_centros[1], 2)
207         pygame.draw.line(screen, Celeste, lista_centros[32], lista_centros[21], 2)
208         pygame.draw.line(screen, Magenta, lista_centros[32], lista_centros[14], 2)
209         pygame.draw.line(screen, Verde, lista_centros[32], lista_centros[26], 2)
210         pygame.draw.line(screen, BLACK, lista_centros[32], lista_centros[0], 1)
211         pygame.draw.line(screen, Marron, lista_centros[32], lista_centros[38], 2)
212         #Nodo 14
213         pygame.draw.line(screen, BLUE, lista_centros[14], lista_centros[15], 2)
214         pygame.draw.line(screen, Celeste, lista_centros[14], lista_centros[21], 2)
215         pygame.draw.line(screen, Magenta, lista_centros[14], lista_centros[14], 2)
216         pygame.draw.line(screen, Verde, lista_centros[14], lista_centros[26], 2)
217         pygame.draw.line(screen, BLACK, lista_centros[14], lista_centros[0], 1)
218         pygame.draw.line(screen, Verde_lima, lista_centros[14], lista_centros[19], 2)
219         #Nodo 15
220         pygame.draw.line(screen, BLUE, lista_centros[15], lista_centros[1], 2)
221         pygame.draw.line(screen, Celeste, lista_centros[15], lista_centros[21], 2)
222         pygame.draw.line(screen, Magenta, lista_centros[15], lista_centros[14], 2)
223         pygame.draw.line(screen, Verde, lista_centros[15], lista_centros[26], 2)
224         pygame.draw.line(screen, BLACK, lista_centros[15], lista_centros[0], 1)
225         pygame.draw.line(screen, Marron, lista_centros[15], lista_centros[2], 2)
226         pygame.draw.line(screen, Lavanda, lista_centros[15], lista_centros[16], 2)
227         #Nodo 16
228         pygame.draw.line(screen, BLUE, lista_centros[16], lista_centros[1], 2)
229         pygame.draw.line(screen, Celeste, lista_centros[16], lista_centros[17], 2)
230         pygame.draw.line(screen, Magenta, lista_centros[16], lista_centros[14], 2)
```

```
231         pygame.draw.line(screen, Verde, lista_centros[16], lista_centros[26], 2)
232         pygame.draw.line(screen, BLACK, lista_centros[16], lista_centros[0], 1)
233         #Nodo 17
234         pygame.draw.line(screen, BLUE, lista_centros[17], lista_centros[1], 2)
235         pygame.draw.line(screen, Celeste, lista_centros[17], lista_centros[21], 2)
236         pygame.draw.line(screen, Magenta, lista_centros[17], lista_centros[22], 2)
237         pygame.draw.line(screen, Verde, lista_centros[17], lista_centros[26], 2)
238         pygame.draw.line(screen, BLACK, lista_centros[17], lista_centros[0], 1)
239         pygame.draw.line(screen, Cyan, lista_centros[17], lista_centros[18], 2)
240         #Nodo 18
241         pygame.draw.line(screen, BLUE, lista_centros[18], lista_centros[1], 2)
242         pygame.draw.line(screen, Celeste, lista_centros[18], lista_centros[21], 2)
243         pygame.draw.line(screen, Magenta, lista_centros[18], lista_centros[36], 2)
244         pygame.draw.line(screen, Verde, lista_centros[18], lista_centros[26], 2)
245         pygame.draw.line(screen, BLACK, lista_centros[18], lista_centros[0], 1)
246         #Nodo 19
247         pygame.draw.line(screen, BLUE, lista_centros[19], lista_centros[1], 2)
248         pygame.draw.line(screen, Celeste, lista_centros[19], lista_centros[21], 2)
249         pygame.draw.line(screen, Magenta, lista_centros[19], lista_centros[14], 2)
250         pygame.draw.line(screen, Verde, lista_centros[19], lista_centros[26], 2)
251         pygame.draw.line(screen, BLACK, lista_centros[19], lista_centros[0], 1)
252         pygame.draw.line(screen, Coral, lista_centros[19], lista_centros[20], 2)
253         #Nodo 20
254         pygame.draw.line(screen, BLUE, lista_centros[20], lista_centros[1], 2)
255         pygame.draw.line(screen, Celeste, lista_centros[20], lista_centros[21], 2)
256         pygame.draw.line(screen, Magenta, lista_centros[20], lista_centros[14], 2)
257         pygame.draw.line(screen, Verde, lista_centros[20], lista_centros[26], 2)
258         pygame.draw.line(screen, BLACK, lista_centros[20], lista_centros[0], 1)
259         pygame.draw.line(screen, Naranja, lista_centros[20], lista_centros[33], 2)
260         # Nodo 33
261         pygame.draw.line(screen, BLUE, lista_centros[33], lista_centros[34], 2)
262         pygame.draw.line(screen, Celeste, lista_centros[33], lista_centros[21], 2)
263         pygame.draw.line(screen, Magenta, lista_centros[33], lista_centros[14], 2)
264         pygame.draw.line(screen, Verde, lista_centros[33], lista_centros[26], 2)
265         pygame.draw.line(screen, BLACK, lista_centros[33], lista_centros[0], 1)
266         # Nodo 34
267         pygame.draw.line(screen, BLUE, lista_centros[34], lista_centros[1], 2)
268         pygame.draw.line(screen, Celeste, lista_centros[34], lista_centros[21], 2)
269         pygame.draw.line(screen, Magenta, lista_centros[34], lista_centros[14], 2)
270         pygame.draw.line(screen, Verde, lista_centros[34], lista_centros[26], 2)
271         pygame.draw.line(screen, BLACK, lista_centros[34], lista_centros[0], 1)
272         pygame.draw.line(screen, Marron, lista_centros[34], lista_centros[40], 2)
273         # Nodo 21
274         pygame.draw.line(screen, BLUE, lista_centros[21], lista_centros[1], 2)
275         pygame.draw.line(screen, Celeste, lista_centros[21], lista_centros[21], 2)
```

```
276         pygame.draw.line(screen, Magenta, lista_centros[21], lista_centros[22], 2)
277         pygame.draw.line(screen, Verde, lista_centros[21], lista_centros[26], 2)
278         pygame.draw.line(screen, BLACK, lista_centros[21], lista_centros[0], 1)
279         # Nodo 22
280         pygame.draw.line(screen, BLUE, lista_centros[22], lista_centros[15], 2)
281         pygame.draw.line(screen, Celeste, lista_centros[22], lista_centros[21], 2)
282         pygame.draw.line(screen, Magenta, lista_centros[22], lista_centros[14], 2)
283         pygame.draw.line(screen, Verde, lista_centros[22], lista_centros[26], 2)
284         pygame.draw.line(screen, BLACK, lista_centros[22], lista_centros[0], 1)
285         pygame.draw.line(screen, Verde_lima, lista_centros[22], lista_centros[23], 2)
286         # Nodo 23
287         pygame.draw.line(screen, BLUE, lista_centros[23], lista_centros[1], 2)
288         pygame.draw.line(screen, Celeste, lista_centros[23], lista_centros[21], 2)
289         pygame.draw.line(screen, Magenta, lista_centros[23], lista_centros[14], 2)
290         pygame.draw.line(screen, Verde, lista_centros[23], lista_centros[24], 2)
291         pygame.draw.line(screen, BLACK, lista_centros[23], lista_centros[0], 1)
292         pygame.draw.line(screen, Coral, lista_centros[23], lista_centros[20], 2)
293         # Nodo 24
294         pygame.draw.line(screen, BLUE, lista_centros[24], lista_centros[25], 2)
295         pygame.draw.line(screen, Celeste, lista_centros[24], lista_centros[21], 2)
296         pygame.draw.line(screen, Magenta, lista_centros[24], lista_centros[14], 2)
297         pygame.draw.line(screen, Verde, lista_centros[24], lista_centros[26], 2)
298         pygame.draw.line(screen, BLACK, lista_centros[24], lista_centros[0], 1)
299         pygame.draw.line(screen, Celeste, lista_centros[24], lista_centros[27], 2)
300         # Nodo 25
301         pygame.draw.line(screen, BLUE, lista_centros[25], lista_centros[1], 2)
302         pygame.draw.line(screen, Celeste, lista_centros[25], lista_centros[21], 2)
303         pygame.draw.line(screen, Magenta, lista_centros[25], lista_centros[14], 2)
304         pygame.draw.line(screen, Verde, lista_centros[25], lista_centros[26], 2)
305         pygame.draw.line(screen, BLACK, lista_centros[25], lista_centros[0], 1)
306         pygame.draw.line(screen, Lavanda, lista_centros[25], lista_centros[37], 2)
307         # Nodo 26
308         pygame.draw.line(screen, BLUE, lista_centros[26], lista_centros[1], 2)
309         pygame.draw.line(screen, Celeste, lista_centros[26], lista_centros[21], 2)
310         pygame.draw.line(screen, Magenta, lista_centros[26], lista_centros[14], 2)
311         pygame.draw.line(screen, Verde, lista_centros[26], lista_centros[26], 2)
312         pygame.draw.line(screen, BLACK, lista_centros[26], lista_centros[0], 1)
313         pygame.draw.line(screen, Gris, lista_centros[26], lista_centros[27], 2)
314         # Nodo 27
315         pygame.draw.line(screen, BLUE, lista_centros[27], lista_centros[1], 2)
316         pygame.draw.line(screen, Celeste, lista_centros[27], lista_centros[21], 2)
317         pygame.draw.line(screen, Magenta, lista_centros[27], lista_centros[14], 2)
318         pygame.draw.line(screen, Verde, lista_centros[27], lista_centros[26], 2)
319         pygame.draw.line(screen, BLACK, lista_centros[27], lista_centros[0], 1)
320         pygame.draw.line(screen, Amarillo, lista_centros[27], lista_centros[28], 2)
```

```
321         # Nodo 28
322         pygame.draw.line(screen, BLUE, lista_centros[28], lista_centros[1], 2)
323         pygame.draw.line(screen, Celeste, lista_centros[28], lista_centros[21], 2)
324         pygame.draw.line(screen, Magenta, lista_centros[28], lista_centros[14], 2)
325         pygame.draw.line(screen, Verde, lista_centros[28], lista_centros[26], 2)
326         pygame.draw.line(screen, BLACK, lista_centros[28], lista_centros[0], 1)
327         pygame.draw.line(screen, Gris, lista_centros[28], lista_centros[29], 2)
328         # Nodo 29
329         pygame.draw.line(screen, BLUE, lista_centros[29], lista_centros[1], 2)
330         pygame.draw.line(screen, Celeste, lista_centros[29], lista_centros[21], 2)
331         pygame.draw.line(screen, Magenta, lista_centros[29], lista_centros[14], 2)
332         pygame.draw.line(screen, Verde, lista_centros[29], lista_centros[26], 2)
333         pygame.draw.line(screen, BLACK, lista_centros[29], lista_centros[0], 1)
334         pygame.draw.line(screen, Lavanda, lista_centros[29], lista_centros[30], 2)
335         # Nodo 30
336         pygame.draw.line(screen, BLUE, lista_centros[30], lista_centros[1], 2)
337         pygame.draw.line(screen, Celeste, lista_centros[30], lista_centros[21], 2)
338         pygame.draw.line(screen, Magenta, lista_centros[30], lista_centros[14], 2)
339         pygame.draw.line(screen, Verde, lista_centros[30], lista_centros[26], 2)
340         pygame.draw.line(screen, BLACK, lista_centros[30], lista_centros[0], 1)
341         pygame.draw.line(screen, RED, lista_centros[30], lista_centros[31], 2)
342         # Nodo 31
343         pygame.draw.line(screen, BLUE, lista_centros[31], lista_centros[1], 2)
344         pygame.draw.line(screen, Celeste, lista_centros[31], lista_centros[21], 2)
345         pygame.draw.line(screen, Magenta, lista_centros[31], lista_centros[14], 2)
346         pygame.draw.line(screen, Verde, lista_centros[31], lista_centros[26], 2)
347         pygame.draw.line(screen, BLACK, lista_centros[31], lista_centros[0], 1)
348         pygame.draw.line(screen, Gris, lista_centros[31], lista_centros[39], 2)
349         # Nodo 35
350         pygame.draw.line(screen, BLUE, lista_centros[35], lista_centros[1], 2)
351         pygame.draw.line(screen, Celeste, lista_centros[35], lista_centros[21], 2)
352         pygame.draw.line(screen, Magenta, lista_centros[35], lista_centros[14], 2)
353         pygame.draw.line(screen, Verde, lista_centros[35], lista_centros[26], 2)
354         pygame.draw.line(screen, BLACK, lista_centros[35], lista_centros[0], 1)
355         # Nodo 36
356         pygame.draw.line(screen, BLUE, lista_centros[36], lista_centros[15], 2)
357         pygame.draw.line(screen, Celeste, lista_centros[36], lista_centros[21], 2)
358         pygame.draw.line(screen, Magenta, lista_centros[36], lista_centros[14], 2)
359         pygame.draw.line(screen, Verde, lista_centros[36], lista_centros[26], 2)
360         pygame.draw.line(screen, BLACK, lista_centros[36], lista_centros[0], 1)
361         pygame.draw.line(screen, Verde_lima, lista_centros[36], lista_centros[19], 2)
362         # Nodo 37
363         pygame.draw.line(screen, BLUE, lista_centros[37], lista_centros[1], 2)
364         pygame.draw.line(screen, Celeste, lista_centros[37], lista_centros[21], 2)
365         pygame.draw.line(screen, Magenta, lista_centros[37], lista_centros[14], 2)
```

```
366         pygame.draw.line(screen, Verde, lista_centros[37], lista_centros[26], 2)
367         pygame.draw.line(screen, BLACK, lista_centros[37], lista_centros[0], 1)
368         # Nodo 38
369         pygame.draw.line(screen, BLUE, lista_centros[38], lista_centros[1], 2)
370         pygame.draw.line(screen, Celeste, lista_centros[38], lista_centros[21], 2)
371         pygame.draw.line(screen, Magenta, lista_centros[38], lista_centros[14], 2)
372         pygame.draw.line(screen, Verde, lista_centros[38], lista_centros[26], 2)
373         pygame.draw.line(screen, BLACK, lista_centros[38], lista_centros[0], 1)
374         pygame.draw.line(screen, Celeste, lista_centros[38], lista_centros[7], 2)
375         # Nodo 39
376         pygame.draw.line(screen, BLUE, lista_centros[39], lista_centros[1], 2)
377         pygame.draw.line(screen, Celeste, lista_centros[39], lista_centros[21], 2)
378         pygame.draw.line(screen, Magenta, lista_centros[39], lista_centros[14], 2)
379         pygame.draw.line(screen, Verde, lista_centros[39], lista_centros[26], 2)
380         pygame.draw.line(screen, BLACK, lista_centros[39], lista_centros[0], 1)
381         # Nodo 40
382         pygame.draw.line(screen, BLUE, lista_centros[40], lista_centros[1], 2)
383         pygame.draw.line(screen, Celeste, lista_centros[40], lista_centros[21], 2)
384         pygame.draw.line(screen, Magenta, lista_centros[40], lista_centros[14], 2)
385         pygame.draw.line(screen, Verde, lista_centros[40], lista_centros[26], 2)
386         pygame.draw.line(screen, BLACK, lista_centros[40], lista_centros[0], 1)
387         pygame.draw.line(screen, Celeste, lista_centros[40], lista_centros[7], 2)
388         # Actualizar la ventana
389         pygame.display.flip()
390
391
392 print(lista_centros[0])
393 print(lista_centros[1])
394 print(lista_centros[14])
395 print(lista_centros[21])
396 print(lista_centros[26])
397 # Salir del programa
398 pygame.quit()
399 sys.exit()
```