CENTRO DE CIENCIAS BÁSICAS

DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN

AUTÓMATAS I

5° "A"

PROYECTO FINAL: NPDA

Profesor: Israel de la Parra González

Alumnos:

Espinoza Sánchez Joel Alejandro

Flores Fernández Óscar Alonso

Gómez Garza Dariana

González Arenas Fernando Francisco

Martínez Gaytán Marco Antonio

Ordaz de Vierna Andrea Juliett

Fecha de Entrega: Aguascalientes, Ags.**, 16** de diciembre de 2020

Índice

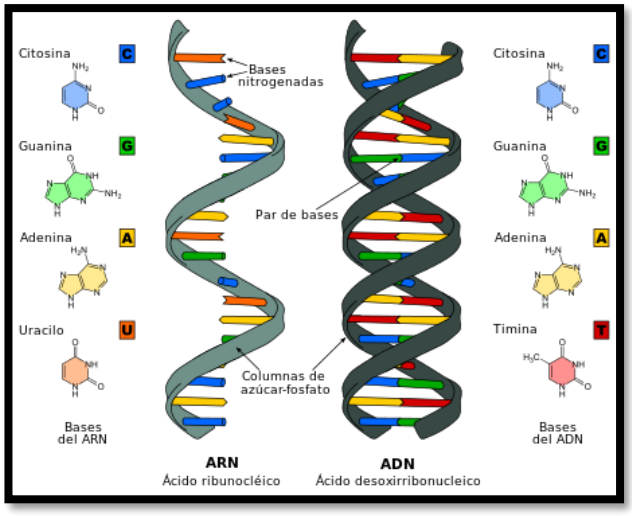
Especificación del proyecto

Los autómatas de pila funcionan de forma similar a los autómatas finitos, también se pueden utilizar para aceptar cadenas de un lenguaje definido sobre un alfabeto A.

Los autómatas de pila pueden aceptar lenguajes que no pueden aceptar los autómatas finitos. Un autómata de pila cuenta con una cinta de entrada y un mecanismo de control que puede encontrarse en uno de entre un número finito de estados. Uno de estos estados se designa como estado inicial, y además algunos estados se llaman de aceptación o finales. A diferencia de los autómatas finitos, los autómatas de pila cuentan con una memoria auxiliar llamada pila. Los símbolos (llamados símbolos de pila) pueden ser insertados o extraídos de la pila, de acuerdo con el manejo last-in-first-out (LIFO).

Las transiciones entre los estados que ejecutan los autómatas de pila dependen de los símbolos de entrada y de los símbolos de la pila. El autómata acepta una cadena x si la secuencia de transiciones, comenzando en estado inicial y con pila vacía, conduce a un estado final, después de leer toda la cadena x.

Se tiene la intención de crear un NPDA relacionado con el ADN del ser humano usando las bases nitrogenadas. El ADN está formado por dos cadenas de nucleótidos complementarias que forman una doble hélice. Los nucleótidos son las moléculas que componen el ADN. Están formados por un grupo fosfato, un azúcar (desoxirribosa) y una base nitrogenada.



En el ADN hay cuatro tipos de nucleótidos que se diferencian por la base nitrogenada que tienen: adenina (A), guanina (G), citosina (C) y timina (T). Estas moléculas se ponen una detrás de otra y forman una cadena muy larga. Para entenderlo mejor, nos podemos imaginar un collar de bolas de cuatro colores. Cada nucleótido es una bola. Como el ADN son dos cadenas, tendríamos dos collares, uno junto al otro y enroscados para formar la doble hélice.

Son complementarias, porque cada nucleótido se une con uno específico de la cadena de delante. La adenina y la timina se pueden unir mediante dos enlaces, y la guanina y la citosina por tres enlaces.

De esta manera para que una cadena de ADN sea válida tendrá que tener sus bases nitrogenadas equilibradas, es decir, que la adenina compense a la timina, la citocina a la guanina y viceversa.

Plan de trabajo

La primera reunión se dedicó para hacer una lluvia de ideas para lograr definir lo que se iba a aplicar para el proyecto, llegando al tema del ADN y el usar como cadenas sus bases nitrogenadas.

Una vez que se estableció el tema del proyecto se empezó la investigación del tema propuesto para el desarrollo del modelo de la solución.

Al ya contar con la información suficiente proseguimos al diseño del NPDA y a empezar con la programación de la pila. Durante este lapso, empezamos a definir los detalles, así como:

- ➔ Cadena aceptada.
- ➔ Cadenas no aceptadas.
- ➔ Cadenas no aceptadas.
- ➔ Alfabetos.
- ➔ Transiciones.
- ➔ Número de estados.

Para avanzar de una forma práctica optamos porque unas personas se dedicarán al código y otras a la documentación para optimizar el tiempo.

Modelos teóricos de solución

➔ Alfabeto del autómata

Para esto $\sum = \{T, A, G, C, 1\}$

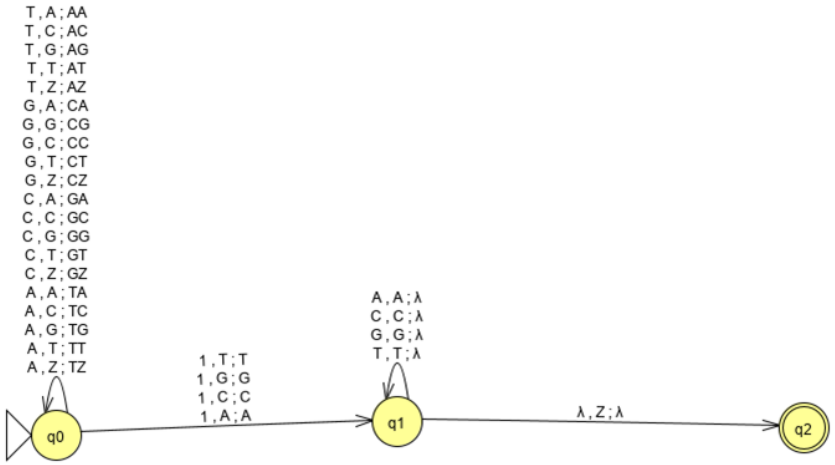
➔ Alfabeto de la pila

Γ = {Z, T, A, G, C},

➔ Estados

{q0, q1, q2}

➔ NPDA:



➔ Cadena aceptada

Sera la cual tenga validez como una cadena correcta de ADN. El que una cadena de ADN sea válida dependerá del número de sus componentes con sus enlaces siguientes.

Como ya mencionamos, la validación de las cadenas dependerá de sus componentes ya que tiene que existir un equilibrio de las bases nitrogenadas dentro de la cadena.

El funcionamiento consistirá en “dividir” la cadena en dos partes que simularan las dos hélices del ADN, la primera parte de la cadena se leerá de inicio a fin mientras que la segunda tendrá que complementar la primera con sus bases nitrogenadas simulando un efecto espejo con el objetivo de completarla y tener un equilibrio.

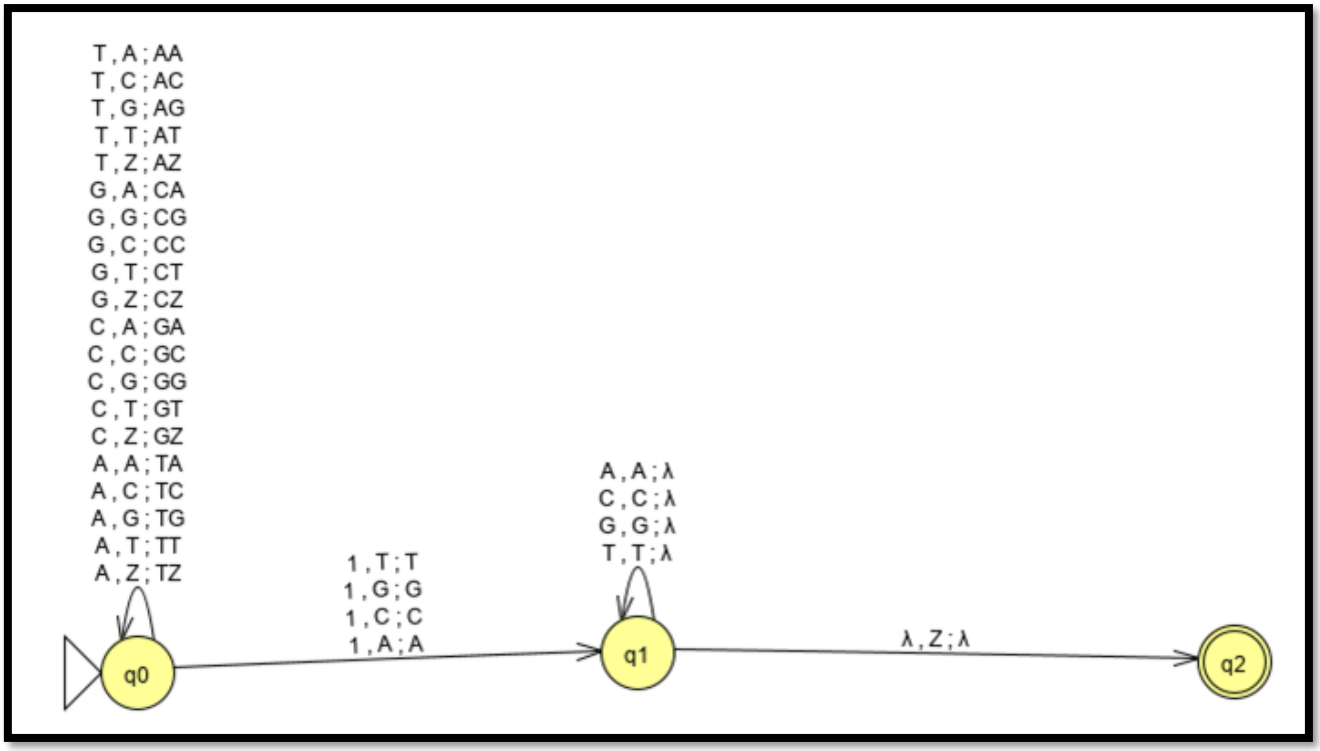
➔ Cadenas no aceptadas:

Las cadenas no aceptadas serán las cuales no satisfagan la condición de equilibrio.

Definiendo M como:

$$M = (\{q0,q1,q2\},\{1, Z, T, A, G, C\},\{T, A, G, C\}, \delta,z,q0,\{q2\})$$

Comprobación del modelo con JFLAP



Nuestro diseño del autómata introducido en JFLAP consiste en 3 estados, se ve sencillo el autómata pero para poder formularlo fue bastante difícil, dado a que estuvimos estudiando las bases nitrogenadas, la forma en la que implementamos el autómata es que como ya se platicó anteriormente el 1 es el que divide nuestras elipses, antes del 1 es la elipse que va hacia arriba y después del 1 es la elipse que va hacia abajo, en este caso la documentación nos dice que si antes del 1 detecta una A (Adenina) entonces después del 1 tendremos una T (Timina), si antes del 1 tenemos una C (Citosina) tendremos después del 1 tendremos una G (Guanina), si antes del 1 tenemos una G (Guanina) después del 1 tendremos una C (Citosina), si antes del 1 tenemos una T (Timina), después del 1 tendremos una A (Adenina).

Entonces dada esta lógica lo que se hace es que como recordaremos una pila al comienzo de la ejecución tiene almacenada en ella una Z, entonces al entrar en nuestro estado inicial (q0) lo que hace es leer la cadena hasta que llegue el 1, es decir leerá la primera elipse, entonces en este primer estado lo que se hará es que se ira llenando la pila, no eliminaremos nada, solo iremos agregando valores a la pila. Como se puede observar habíamos dicho algunas condiciones en este caso si lee una A del otro lado debíamos tener una T, es decir, si tenemos una A, almacenaremos en la pila una T, los otros valores por ejemplo la línea que dice A,Z;TZ lo que se hace es dejar neutra la Z y solo agregar T, ¿Por qué se hace esto? Porque recordaremos que la teoría de la pila es ir leyendo el tope de la misma, por lo que si en una primera instancia leemos una A sabemos que tenemos una Z en el tope, así que eliminamos la Z y agregamos TZ que es lo mismo decir que simplemente solo agregamos T a la pila si eliminar nada, esto se hace para todo nuestro lenguaje (A,C,G,T), en el caso de una C de igual manera, habíamos comentado que si se leía una C entonces del otro lado debía de haber una G, es decir, si se lee una C entonces en la pila

se guarda una G y lo demás de igual manera solo lo hacemos para dejar neutro la pila y que solo vaya agregando a la pila los caracteres, esto en forma de lista seria:

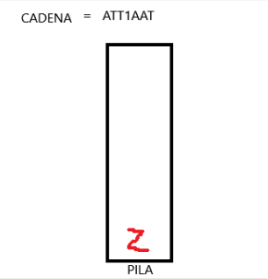
- ✓ Si se lee A == se introduce T a la pila
- ✓ Si se lee C == se introduce G a la pila
- ✓ Si se lee G == se introduce C a la pila
- ✓ Si se lee T == se introduce A a la pila

Dado esto lo repetimos hasta que encuentre un uno, una vez leído el 1 lo que se hace es pasarse al estado q1, el cual en este estado ahora eliminaremos caracteres de la pila esperando a que se quede con una Z sin ningún otro carácter más. En este caso es más sencillo, si después del 1 se lee una A, se elimina es A y no se guarda nada en la pila, si se lee una C , se elimina C y no se guarda nada en la pila, si se lee G, se elimina G y no se guarda nada en la pila y si se lee T, se elimina T y no se guarda en la pila, esto con el objetivo que todo lo que metemos es lo que debe de salir si es que se cumple la condición que pusimos en la lista anterior, una vez que termina de leer la cadena, se procede a pasarse a q1 y se elimina la Z y no se guarda nada en la pila por lo que si llega a q1 y la pila no tiene ningún carácter entonces quiere decir que la cadena es aceptada.

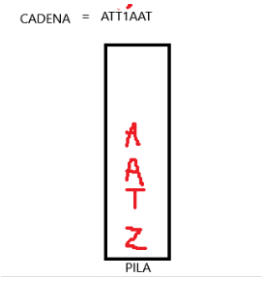
Comprobando esto mismo, se observará el ejemplo de la cadena ATT1AAT la cual es una cadena que debería ser aceptada, pero veremos si nuestro autómata la acepta

Entonces tenemos …

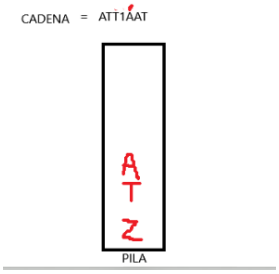
- Al comenzar la pila se encuentra con una Z.



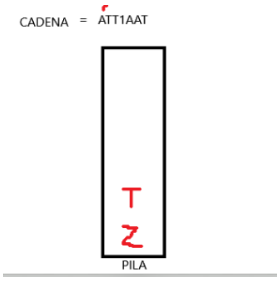
- Se lee el primer carácter el cual es A, así que al ser una A y Observamos que el tope de nuestra pila tiene una Z entonces se elimina una Z y se guarda un TZ .



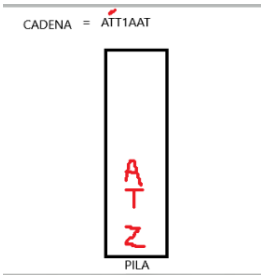
- Se lee el segundo carácter que es una T, así que se ve que el tope es una T por lo que se elimina la T y se guarda un AT



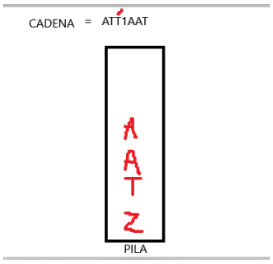
- Se lee el tercer carácter que es una T, así que se ve que el tope es una A por lo que se elimina una A y se guarda una AA



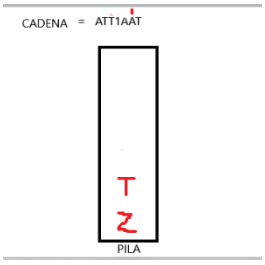
- Se lee el cuarto carácter que es un 1, así que se ve que el tope es una A por lo que se elimina esa A y se guarda una A y se pasa al estado q1



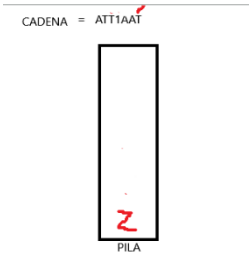
- Se lee el quinto carácter que es una A, así que se ve que el tope es una A por lo que se elimina una A y no se guarda nada en la pila



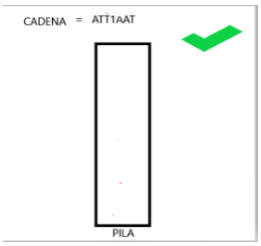
- Se lee el sexto carácter que es una A, así que se ve que el tope es una A por lo que se elimina A y no se guarda nada en la pila



- Se lee el séptimo carácter que es una T, así que se ve que el tope es una T por lo que se elimina una T y no se guarda nada en la pila



- Por último, se observa que ya no hay caracteres que leer por lo que se pasa al estado q2 y se elimina una Z de la pila y no se guarda nada, por lo que se observa que la pila se encuentra vacía por lo que se concluye en que la cadena introducida en efecto es una cadena de aceptación.



Código en el lenguaje de programación

Lo primero que realiza el código es preguntar la cadena que el usuario desea ingresar, esta cadena se guardará en un vector llamado `cadena`; para que éste funcione de manera correcta, se debe especificar las partes de las cadenas que representan las 2 hélices. Esto se hace poniendo un "1" en el intermedio de la cadena, y así el programa entenderá cuál debe de leer primero (al igual que en el autómata).

```
int i,j,k,n;
printf("Escribe la cadena:\n");
scanf("%s",&cadena);
n=0;
do
{
    n++;
}
while(cadena[n]);
printf("\n\n%d\n",n);
```

Al introducir la cadena, verificamos el número de caracteres que contiene.

En el programa podemos encontrar 2 contadores, el contador `i` que es el que irá contando las posiciones que contiene la cadena y el contador `j` que cuenta las posiciones de la pila al igual que su vector, el contador `i` lo inicializamos en 0, ya que comenzará a leer la cadena desde ahí.

En el vector `pila` almacenaremos lo que se irá introduciendo o sacando de la pila.

Comenzamos iniciando el vector `pila` en su posición 0 con una "Z" ya que siempre iniciamos la pila así, al igual que un autómata de JFLAP. E iniciamos su contador en 1, ya que la posición 0 está ocupada.

```
char pila[n+1];
pila[0] = 'Z';
i=0;
j=1;
```

Una vez declarando lo anterior, realizamos un ciclo que se repetirá hasta que la cadena en la posición actual de `i` diferente de "1".

La instrucción que se debe realizar dentro del ciclo es hacer 4 condiciones que preguntan si en el vector `cadena` en su posición actual se encuentra A, C, G o T (un

if para cada carácter) debe agregarse la letra que complementa a esa cadena en el vector `pila` y al contador `j` se le sumará uno. Una vez salga de todos los condicionales, se le incrementará uno al contador `i`.

```
do
{
  //  printf("Estamos leyendo el
    if(cadena[i] == 'A')
    {
        pila[j] = 'T';
        j++;
    }
    if(cadena[i] == 'C')
    {
        pila[j] = 'G';
        j++;
    }
    if(cadena[i] == 'G')
    {
        pila[j] = 'C';
        j++;
    }
    if(cadena[i] == 'T')
    {
        pila[j] = 'A';
        j++;
    }
    i++;
}
while(cadena[i] != '1');
```

Cuando el ciclo termine y `cadena` en su posición `i` sea diferente de “1”, se incrementará 1 el contador `i`, ya que se leyó un carácter más.

Una vez terminando la primera parte vaciaremos la pila; esto lo haremos con otro ciclo. La condición del siguiente ciclo es que se repita el proceso hasta que la pila en su posición anterior sea diferente de “Z” y que nuestro contador `i` sea menor a la cantidad de caracteres de la cadena.

Lo que se hará dentro del ciclo es volver a preguntar carácter por carácter si en los vectores `cadena` y `pila` son iguales a cualquier carácter que es válido, entonces al vector `pila` en su posición anterior se agregará un “0”, ya que con esto interpretamos que ese elemento ya se sacó de la pila. Y al contador `j` se le restará 1 y se incrementa el contador `i`.

```
do
{
    //printf("Estamos leyendo el elemento %d: %c con pila %s y borde de pila: %c\n",i,cadena[i],pila,pila[j-1]);
    if((cadena[i] == 'A') && (pila[j-1] == 'A'))
    {
        pila[j-1] = '0';
        j--;
    }
    if((cadena[i] == 'C') && (pila[j-1] == 'C'))
    {
        pila[j-1] = '0';
        j--;
    }
    if((cadena[i] == 'G') && (pila[j-1] == 'G'))
    {
        pila[j-1] = '0';
        j--;
    }
    if((cadena[i] == 'T') && (pila[j-1] == 'T'))
    {
        pila[j-1] = '0';
        j--;
    }
    i++;
```

Y finalmente, se realiza un condicional que pregunta si el último elemento del vector `pila` es igual a “Z”, entonces habremos llegado al estado final y la cadena será aceptada.

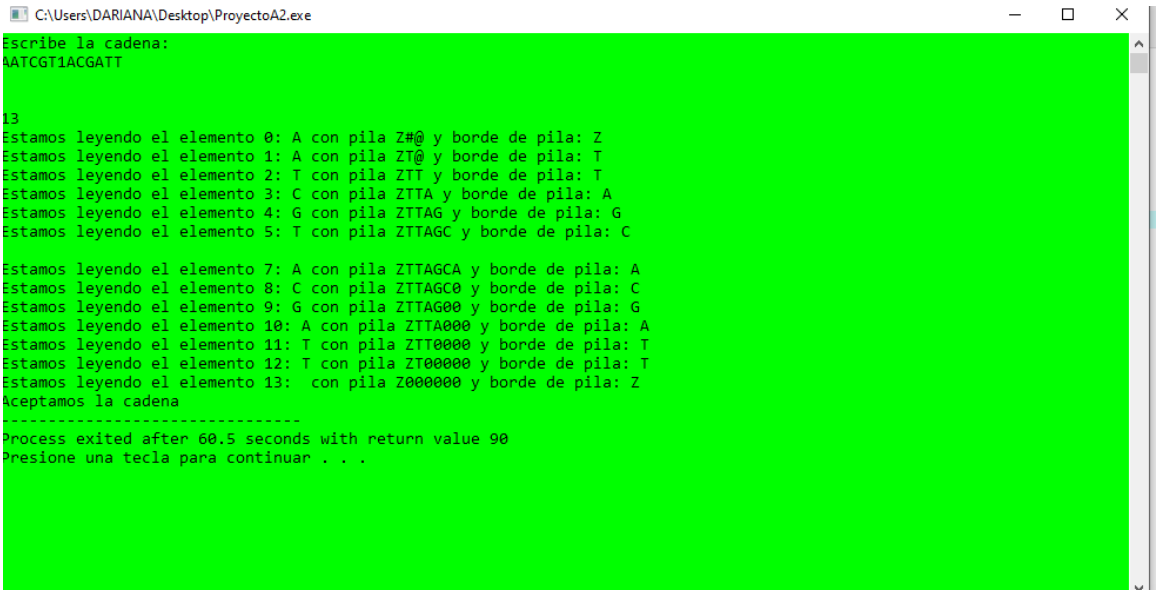
```
if(pila[j-1] == 'Z')
{

    printf("Aceptamos la cadena");
}
```

Si se desea observar el código completo, observar el anexo 1.

A continuación, se muestra una captura de pantalla del funcionamiento del programa al correrlo:

Introducimos la cadena AATCGT1ACGATT, después de introducirla nos muestra los movimientos que se hacen en la pila y si la cadena es aceptada o no.



Conclusiones

Espinoza Sánchez Joel Alejandro:

Este proyecto nos permitió a todos como grupo cambiar la dinámica de programación, ya que había que pensar de forma distinta al desarrollo normal de programación, e incluso tratar especialmente a las cadenas, ya que, en caso de los autómatas, se deberá leer una cadena de texto o como se definió en el curso, una palabra.

Para el proyecto del análisis de ADN por medio de la validación de bases nitrogenadas, la complicación yacía en el planteamiento del problema, ya que era complicado buscar el conjunto de cadenas que requiriera el uso de las pilas, pero para la validación de cadenas de ADN, fue lo más óptimo y su desarrollo no fue tan complicado a nivel de autómata, pero sí a nivel de programación.

Flores Fernández Óscar Alonso:

Al ir realizando la segunda entrega de este proyecto veíamos que hay muchas cosas a las que podríamos aplicar un NPDA pero de igual manera unas más interesante e incluso más sencillas de elaborar que otras. Al darnos cuenta de esto y de que las posibilidades de su aplicación podían inclinarse al campo genético creímos que la idea era bastante buena. Por la situación actual gracias al Covid creímos que el basarnos en un pedacito de lo que el mundo enfrenta hoy en día sería interesante. A pesar de seguir siendo un proyecto un tanto ambicioso, nos dimos cuenta de que el resultado obtenido requería mucho menos “espacio” que nuestra primera parte a pesar de la dificultad que representaba el poder darle estructura, ya que los caminos y posibilidades, así como las diferencias entre ambos métodos, se generalizaban en una repuesta más simple. Esto lo podemos ver tanto en el tamaño de nuestro autómata por la diferencia de transiciones y estados que requeríamos y en nuestro código, cuya elaboración, al estar basada en el mismo autómata, fue más corta

Gómez Garza Dariana:

En esta segunda parte del proyecto teníamos algunas ideas de cómo aplicar el NPDA, pero optamos por esta decisión ya que se nos hacía muy interesante la propuesta de nuestro compañero Joel. Para la ejecución de esta segunda parte tardamos un poco menos en poder darle una solución a comparación de la primera parte, ya que era una situación muy diferente y elegimos tener un resultado más general al anterior. Esto no significa que fue fácil, pero sí que su nivel de dificultad disminuyó a comparación del anterior.

Nos tomó menos tiempo realizar el autómata en JFLAP por la diferencia de estados y de transiciones que tenía nuestro autómata, y por lo tanto nuestro segundo código fue mucho más corto que el anterior.

Me gustó el desarrollo de la idea y el proceso de todo este proyecto, fue muy interesante el desarrollo del proyecto completo y ver como complementamos las ideas de cada uno del equipo.

González Arenas Fernando Francisco:

Con el desarrollo de este proyecto se aplicaron muchos conceptos del diseño e implementación de autómatas para la resolución de problemas reales, los cuales fueron autómatas para la predicción de precipitaciones de lluvia (con ayuda de un NFA), y también otro autómata para la validación de cadenas de ADN por medio de bases nitrogenadas (con un NPDA).

Con la realización de este proyecto se reforzó todo lo aprendido en el curso de autómatas 1, lo cual nos puede ser de mucha utilidad para materias futuras en la carrera, para posteriores proyectos en nuestra vida académica o incluso para nuestra vida laboral, ya que los autómatas son un paso muy importante para la creación de compiladores entre muchas otras cosas.

Martínez Gaytán Marco Antonio:

Concluyo en que este autómata fue genial como se analizó y como se planifico ya que son temas biológicos que son difíciles de comprender pero geniales de aprender, fue difícil como obtener un autómata de este problema, parece que fue fácil por el número de estados que se usó pero realmente fue difícil el poder construirlo y que sirviera al 100%, el trabajo en equipo fue perfecto, hubo mucha comunicación y así se logró poder montar un proyecto de muy buena calidad, por lo tanto fue genial dedicar muchas horas a esto por el objetivo al que se llegó y por la forma de trabajo que realizamos en el equipo.

Ordaz de Vierna Andrea Juliett:

Para esta segunda parte del proyecto donde se puso en práctica el tema de este último parcial estuvo entretenido el discutir qué tema queríamos usar para esta aplicación ya que si de por si, en lo personal, este fue el tema más “divertido” del semestre ahora ponerlo en práctica para alguna aplicación… Fue entretenido, pero una vez que se fue definiendo la idea fue más sencillo desarrollar el proyecto gracias al maravilloso trabajo en equipo que pudimos lograr. Tengo que agregar que el proyecto también ayudo mucho a la comprensión del tema, ya que al buscar cómo usarlo debes de entenderlo bien como primer paso. Para concluir, el proyecto cumplió el objetivo de hacernos comprender mejor el tema, así como nos enseñó a cómo aplicarlo a la programación.

Autoevaluación

Espinoza Sánchez Joel Alejandro:

- Espinoza Sánchez Joel Alejandro: 10.
- Flores Fernández Óscar Alonso: 10.
- Gómez Garza Dariana: 10.
- González Arenas Fernando Francisco: 10.
- Martínez Gaytán Marco Antonio: 10.
- Ordaz de Vierna Andrea Juliett: 10.

Flores Fernández Óscar Alonso:

- Espinoza Sánchez Joel Alejandro: 10
- Flores Fernández Óscar Alonso: 8
- Gómez Garza Dariana: 10
- González Arenas Fernando Francisco: 10
- Martínez Gaytán Marco Antonio: 10
- Ordaz de Vierna Andrea Juliett: 10

Gómez Garza Dariana:

- Espinoza Sánchez Joel Alejandro: 10
- Flores Fernández Óscar Alonso: 9
- Gómez Garza Dariana: 8
- González Arenas Fernando Francisco: 10
- Martínez Gaytán Marco Antonio: 10
- Ordaz de Vierna Andrea Juliett: 9

González Arenas Fernando Francisco:

- Espinoza Sánchez Joel Alejandro: 10
- Flores Fernández Óscar Alonso: 9
- Gómez Garza Dariana: 9
- González Arenas Fernando Francisco: 8
- Martínez Gaytán Marco Antonio: 10
- Ordaz de Vierna Andrea Juliett: 9

Martínez Gaytán Marco Antonio:

- Espinoza Sánchez Joel Alejandro: 10
- Flores Fernández Óscar Alonso: 10
- Gómez Garza Dariana: 10
- González Arenas Fernando Francisco: 10
- Martínez Gaytán Marco Antonio: 10
- Ordaz de Vierna Andrea Juliett: 10

Ordaz de Vierna Andrea Juliett:

- Espinoza Sánchez Joel Alejandro: 10
- Flores Fernández Óscar Alonso: 10
- Gómez Garza Dariana: 10
- González Arenas Fernando Francisco: 10
- Martínez Gaytán Marco Antonio: 10
- Ordaz de Vierna Andrea Juliett: 10

Bibliografía

- Anónimo. (2015). *El ADN.* Marzo, 2015, de lacienciaentumundo Sitio web: http://www.lacienciaentumundo.com/el-adn/#:~:text=El%20ADN%20est%C3%A1%20formado%20por,que%20forman%20una%20doble%20h%C3%A9lice.&text=En%20el%20ADN%20hay%20cuatro,forman%20una%20cadena%20muy%20larga.
- Anónimo. *Autómatas de pila,* Sitio web: https://users.exa.unicen.edu.ar/catedras/ccomp1/Apunte4.pdf

Anexos

1. Código completo

```c
#include <stdio.h>

char cadena[];

main()
{
    int i,j,k,n;
    printf("Escribe la cadena:\n");
    scanf("%s",&cadena);
    n=0;
    do
    {
        n++;
    }
    while(cadena[n]);
    printf("\n\n%d\n",n);

    char pila[n+1];
    pila[0] = 'Z';
    i=0;
    j=1;
```

```c
char pila[n+1];
pila[0] = 'Z';
i=0;
j=1;
do
{

    if(cadena[i] == 'A')
    {
        pila[j] = 'T';
        j++;
    }
    if(cadena[i] == 'C')
    {
        pila[j] = 'G';
        j++;
    }
    if(cadena[i] == 'G')
    {
        pila[j] = 'C';
        j++;
    }
    if(cadena[i] == 'T')
    {
        pila[j] = 'A';
        j++;
    }
    i++;
}
while(cadena[i] != '1');

printf("\n");
i++;
```

```
do
{
    if((cadena[i] == 'A') && (pila[j-1] == 'A'))
    {
        pila[j-1] = '0';
        j--;
    }
    if((cadena[i] == 'C') && (pila[j-1] == 'C'))
    {
        pila[j-1] = '0';
        j--;
    }
    if((cadena[i] == 'G') && (pila[j-1] == 'G'))
    {
        pila[j-1] = '0';
        j--;
    }
    if((cadena[i] == 'T') && (pila[j-1] == 'T'))
    {
        pila[j-1] = '0';
        j--;
    }
    i++;
    if(pila[j-1] == 'Z')
    {

        printf("Aceptamos la cadena");
    }
}
while((pila[j-1] != 'Z') && (i < n));
```