

INSTITUTO POLITECNICO NACIONAL

ESCUELA SUPERIOR DE CÓMPUTO

PRACTICA 2

Autor:
Gomez Hernandez Alan Javier

12 octubre, 2021—

PROGRAMA COMBINATORIO

RESUMEN

El programa debe funcionar automáticamente. Generar 10 a la 6, cadenas binarias aleatoriamente de longitud 64. Hacer que el programa se espere 1 segundo. Posteriormente validar cada una de las cadenas con el AFD de paridad. Generar tres archivos de texto para las salidas. El primer archivo tendrá todas las cadenas generadas, el segundo archivo tendrá las cadenas aceptadas y el tercer archivo las cadenas rechazadas. Si el programa entra más de una vez, los archivos deben de almacenar los datos de todas las corridas. Tener la opción de graficar el AFD completo (protocolo y paridad en el mismo grafo).

INTRODUCCION

Para realizar este programa se requieren de conocimientos previos como es el manejo de archivos, math, recursividad, funciones, uso de graficas con LabView, y uso de cadenas para almacenar datos en C. Pero antes de esto el conocimiento previo de el universo de las cadenas binarias (Sigma). Dada una "n". Para empezar un archivo es un conjunto de datos estructurados en una colección de entidades elementales o básicas denominadas registros que son de igual tipo y constan a su vez de diferentes entidades de nivel más bajos denominadas campos. Hay dos tipos de archivos, archivos de texto y archivos binarios. Un archivo de texto es una secuencia de caracteres organizadas en líneas terminadas por un carácter de nueva línea. En estos archivos se pueden almacenar canciones, fuentes de programas, base de datos simples, etc. Los archivos de texto se caracterizan por ser planos, es decir, todas las letras tienen el mismo formato y no hay palabras subrayadas, en negrita, o letras de distinto tamaño o ancho. Y para este trabajo será el que ocuparemos.

Nombre	Función
fopen()	Abre un archivo.
fclose()	Cierra un archivo.
fgets()	Lee una cadena de un archivo.
fputs()	Escribe una cadena en un archivo.
fseek()	Busca un byte específico de un archivo.
fprintf()	Escribe una salida con formato en el archivo.
fscanf()	Lee una entrada con formato desde el archivo.

Figure 1: librería STDIO.H para el uso de esta función

Finalmente lo que es un arreglo Los arrays son variables estructuradas, donde cada elemento se almacena de forma consecutiva en memoria. Ya que obten-

dremos valores y en general el proposito es generar una grafica se utilizara LabView que es realizado para ese proposito. Funciones basicas para realizar el programa serian ciclos anidados, for, if para el protocolo y la paridad.

DESARROLLO

0.1 PROPUESTA DE SOLUCION

Para este programa se propone realizar ciclos anidados para la creación de las cadenas con números binarios, y validaciones con contadores donde verificaremos si es impar o par, dependiendo del número random generado entre uno o cero por ello se le asigna porcentaje 2, teniendo esto se proponen las bibliotecas incluye `<stdio.h>` incluye `<math.h>` incluye `<stdlib.h>`, ahora se crearan 3 ficheros donde, uno será para todas las cadenas, otra para los que logran paridad, y otro donde no se logra la paridad aparte los habremos con `w+` para abrir y crear. Ahora crearemos variables para el control de los for, contadores de control para la imparidad de ceros y unos, como también un array para el almacenamiento del número binario. Con la operación `pow` calculamos el 10 a la 6 para comenzar nuestro ciclo anidado, donde nos referimos con esto es que el primer for va ser el control para el numero de cadenas que va generar para el random de números binarios. Ahora dentro del primer ciclo for (para el numero de cadenas que se generaran de los numeros binarios aleatorios), se crea otro ciclo encargado para crear estas cadenas y ver la imparidad antes de salir de cada ciclo, es decir que el for se repetirá 64 veces por el numero random obtenido y genera el array. Abrimos el fichero1 para almacenar esta cadena que se genera `fprintf` y cuando termine de crear los 64 números le damos un salto de línea y no los concatene, después de esto validamos si es un cero o un uno lo que genera el numero binario y lo guardamos en un contador de unos o ceros según el caso, nos salimos del primer ciclo para obtener la imparidad de la cadena generada en ese ciclo, entonces primero validamos si el número del contador 0 su residuo con 2 es cero, es par, después validamos el contador de 1 si su residuo es de igual forma con 2 0, hay una paridad lógica, en caso de que alguno de los dos no se cumpla hay imparidad, si es esto se abre el fichero 3 de impuridad y creamos otro ciclo para extraer los valores y exportarlo. En caso de que se cumplieran las dos abrimos el fichero dos, hacemos el ciclo de la cadena y exportamos ese valor. Por cada ciclo creado de cadena esperamos 1 segundo y una vez terminados los 10 a la 6 de forma automática concluye el programa. Y al final de cada ciclo generado de la cadena reiniciamos la cadena para no tener errores o numero enormes.

0.2 CODIGO

```
#include <stdio.h>
#include <math.h>
#include <stdlib.h>
```

```

void main(){

/*CREACION DE FICHEROS*/
FILE *fichero;
    fichero = fopen("Cadenas.txt", "w+");
    if(fichero == NULL ) {
        printf("No fue posible abrir/crear el archivo\n");
        return;
    }
FILE *fichero2;
    fichero2 = fopen("Validas.txt", "w+");
    if(fichero2 == NULL ) {
        printf("No fue posible abrir/crear el archivo\n");
        return;
    }
FILE *fichero3;
    fichero3 = fopen("no_validas.txt", "w+");
    if(fichero3 == NULL ) {
        printf("No fue posible abrir/crear el archivo\n");
        return;
    }

    int j=0, n=0, i=0;
    char unos = 0;
    char ceros = 0;
    int numero[65];

int generador = pow(10,6);

for (i = 0;i < generador ; i++)
{
    printf("\nel numero binario es:");
for (j = 0;j < 64 ; j++)
    {
        numero[j] = rand() % 2;
        printf("%d",numero[j]);

        fprintf(fichero,"%d",numero[j]);

if(numero[j] == 1){
    unos++;
}else{
    ceros++;
}
}
}

```

```

    }
    fprintf(fichero, "\n");
    /**/
    if(unos % 2 == 0){
        if(ceros % 2 == 0)
        {
            for (j = 0; j < 64 ; j++){
                fprintf(fichero2, "%d", numero[j]);
            }

            fprintf(fichero2, "\n");
        }else{
            for (j = 0; j < 64 ; j++){
                fprintf(fichero3, "%d", numero[j]);
            }

            fprintf(fichero3, "\n");
        }

        /**/
    } else{
        for (j = 0; j < 64 ; j++){
            fprintf(fichero3, "%d", numero[j]);
        }

        fprintf(fichero3, "\n");
    }

    delay(10000);
    unos = 0;
    ceros = 0;
}

return;
}

```

0.3 FUNCIONAMIENTO Y PRUEBAS

Ahora como se puede observar al ejecutar el programa tardo demasiado ya que tenía el delay de un segundo por lo que se opto a reducirlo y correrlo, de esta manera como dice al final se crean las 1000000 cadenas de números binarios de 64 binarios aleatorios. Entonces cumplimos la primera parte de crearlas.

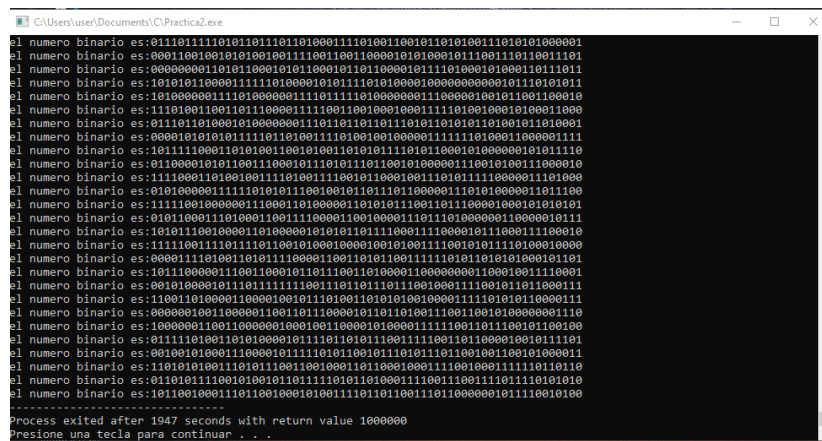


Figure 2: Compilación de código

Por consiguiente para verificar si se generaron los archivos, veremos cuantas cadenas se generaron en total en cada fichero, validas, no valida y cadenas.

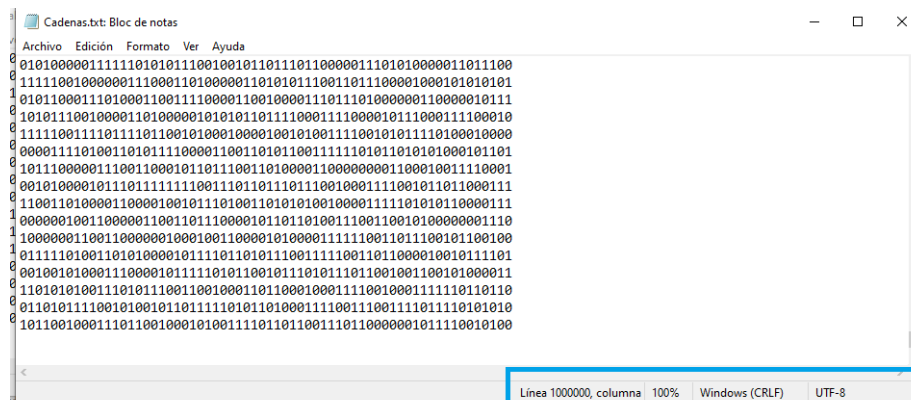


Figure 3: Cadenas

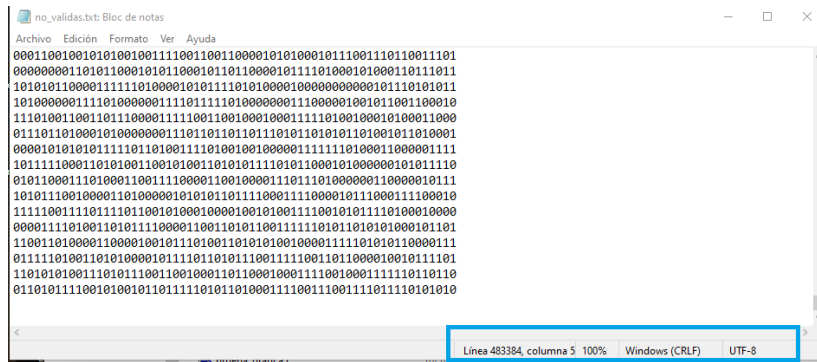


Figure 4: no validas

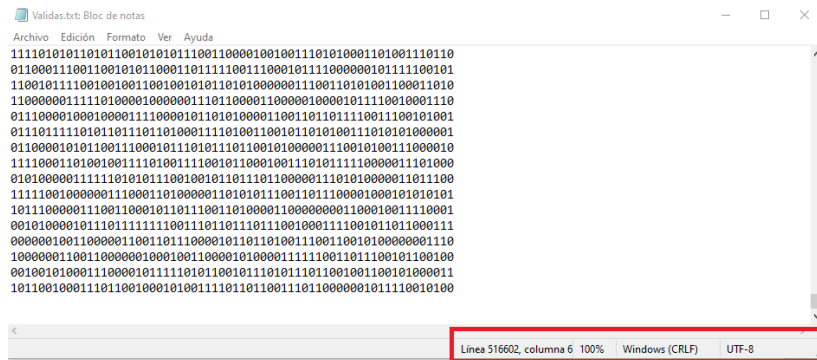


Figure 5: Validas

Después sumamos estas cantidades las de validas y no valida y deben dar el numero de cadenas 1,000,000.



Figure 6: Calculadora

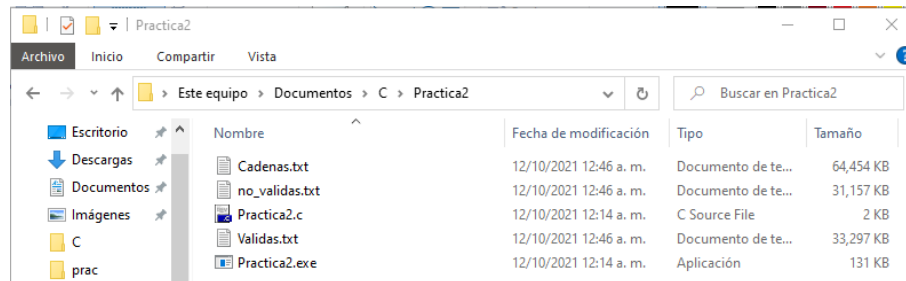


Figure 7: Archivo

GRAFOS

Dado que C no tiene un entorno bien determinado se opta por usar Python para los diagramas del automata - grafos y su función. A continuación se obtiene el código y capturas de pantalla.

0.4 PARIDAD

```
import random
import math
from tkinter import*

archivo = open('datos3.txt','w')

print("Bienvenido al Automata paridad binaria.")
```



```

opc = input("Que desea hacer? \n 1)Ingresar cadena. \n 2)Generar cadena\n")

if opc=="1":
    eleccion = input("Ingrese la cadena a evaluar \n")
    palabra = str(eleccion)

else:
    rand = random.randrange(10000000000000000000)
    palabra = str(bin(rand)[2:])

print ("Cadena a evaluar:",palabra)

c=0 #estado
for x in palabra:

    #estado 0
    if (c == 0 and x == "0"):
        c = 1
        archivo.write("Se paso del estado q0 al q1 \n")
        continue
    if (c == 0 and x == "1"):
        c = 3
        archivo.write("Se paso del estado q0 al q1 \n")
        continue

    #estado 1
    if (c == 1 and x == "0"):
        c = 0
        archivo.write("Se paso del estado q1 al q0 \n")
        continue
    if (c == 1 and x == "1"):
        c = 2
        archivo.write("Se paso del estado q1 al q2 \n")
        continue

    #estado 2
    if (c == 2 and x == "1"):
        c = 1
        archivo.write("Se paso del estado q2 al q1 \n")
        continue
    if (c == 2 and x == "0"):
        c = 3
        archivo.write("Se paso del estado q2 al q3 \n")
        continue

    #estado 3

```

```

if (c == 3 and x == "0"):
    c = 2
    archivo.write("Se paso del estado q3 al q2 \n")
    continue
if (c == 3 and x == "1"):
    c = 0
    archivo.write("Se paso del estado q3 al q0 \n")
    continue
archivo.close

print (c)
if (c == 0):
    print("Se llego al estado final. Se ha generado archivo con historia (datos3.txt) \n La cade
else:
    print("No se llego al estado final. Se ha generado archivo con historia (datos3.txt) \n La c

ventana = Tk()
canv = Canvas(ventana,width=400,height=700)
ventana.geometry("400x700")

ventana.title('Automata Paridad')

p= Label(ventana,text="Automata de paridad. \n Cadena:"+palabra).place(x=10,y=10)

#q0
p0= Label(ventana,text="q0").place(x=55,y=105)

if (c==0):
    canv.create_oval(20,70,110,160, fill="green")
else:
    canv.create_oval(20,70,110,160, fill="blue")
    canv.create_oval(35,85,95,145)
    canv.create_oval(90,145,100,155, fill="black")
    n1= Label(ventana,text="0").place(x=190,y=35)
    canv.create_oval(50,155,60,165, fill="black")
    n2= Label(ventana,text="0").place(x=190,y=185)

#q1
p1= Label(ventana,text="q1").place(x=305,y=105)

if(c==1):
    canv.create_oval(270,70,360,160, fill="red")
else:
    canv.create_oval(270,70,360,160, fill="blue")
    canv.create_oval(280,75,290,85, fill="black")
    n3= Label(ventana,text="1").place(x=255,y=230)

```

```

canv.create_oval(300,155,310,165, fill="black")
n4= Label(ventana,text="1").place(x=360,y=230)

#e2
p2= Label(ventana,text="q2").place(x=305,y=355)

if(c==2):
    canv.create_oval(270,320,360,410, fill="red")
else:
    canv.create_oval(270,320,360,410, fill="blue")
    canv.create_oval(280,325,290,335, fill="black")
    n5= Label(ventana,text="0").place(x=190,y=285)
    canv.create_oval(320,315,330,325, fill="black")
    n6= Label(ventana,text="0").place(x=190,y=435)

#e3
p3= Label(ventana,text="q3").place(x=55,y=355)
if(c==3):
    canv.create_oval(20,320,110,410, fill="red")
else:
    canv.create_oval(20,320,110,410, fill="blue")
    canv.create_oval(90,395,100,405, fill="black")
    n7= Label(ventana,text="1").place(x=10,y=230)
    canv.create_oval(70,315,80,325, fill="black")
    n8= Label(ventana,text="1").place(x=110,y=230)

#lineas
#0-1
xy1 = 95, 60, 286, 105
canv.create_arc(xy1, start=0, extent=180, style="arc")
#1-0
xy2 = 95, 180, 286, 115
canv.create_arc(xy2, start=0, extent=-180, style="arc")

#1-2
xy3 = 286, 160, 356, 320
canv.create_arc(xy3, start=-90, extent=180, style="arc")
#2-1
xy4 = 276, 160, 336, 320
canv.create_arc(xy4, start=90, extent=180, style="arc")

#2-3
xy5 = 95, 430, 286, 365
canv.create_arc(xy5, start=0, extent=-180, style="arc")
#3-2
xy6 = 95, 310, 286, 355

```

```

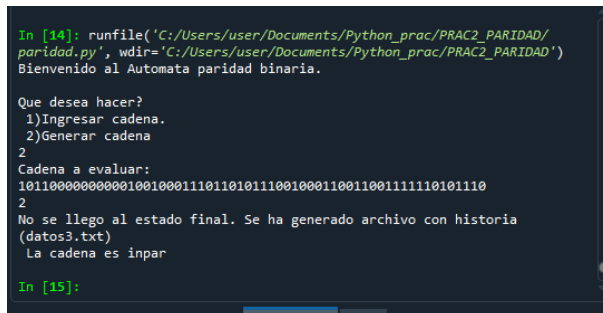
canv.create_arc(xy6, start=0, extent=180, style="arc")

#3-1
xy7 = 36, 160, 106, 320
canv.create_arc(xy7, start=-90, extent=180, style="arc")
#1-3
xy8 = 26, 160, 96, 320
canv.create_arc(xy8, start=90, extent=180, style="arc")

if(c==0):
p= Label(ventana,text="La cadena es par").place(x=10,y=460)
else:
ip= Label(ventana,text="La cadena es impar").place(x=10,y=460)

canv.place(x=0,y=0)
ventana.mainloop()

```



```

In [14]: runfile('C:/Users/user/Documents/Python_prac/PRAC2_PARIDAD/
paridad.py', wdir='C:/Users/user/Documents/Python_prac/PRAC2_PARIDAD')
Bienvenido al Automata paridad binaria.

Que desea hacer?
1)Ingresar cadena.
2)Generar cadena
2
Cadena a evaluar:
10110000000001001000111011010111001000110011001111110101110
2
No se lleo al estado final. Se ha generado archivo con historia
(datos3.txt)
La cadena es impar

In [15]:

```

Figure 8: Para observar mejor lo que hace ponemos la cadena o generamos una aleatoria

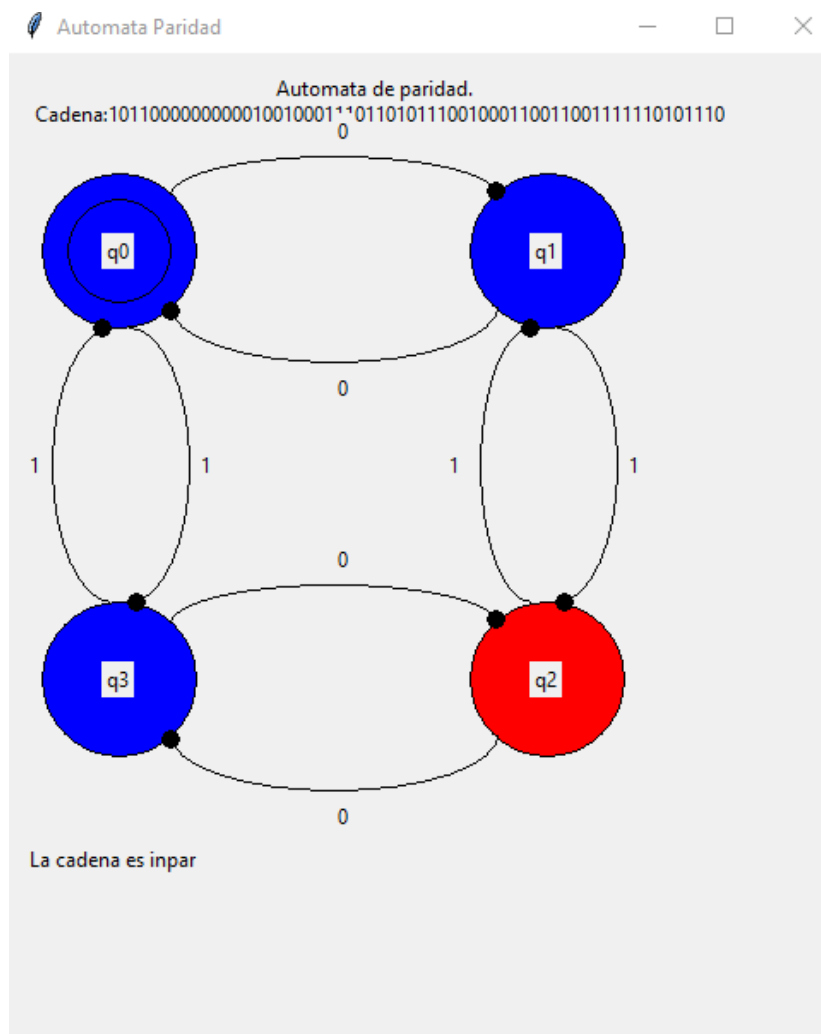


Figure 9: GRAFO

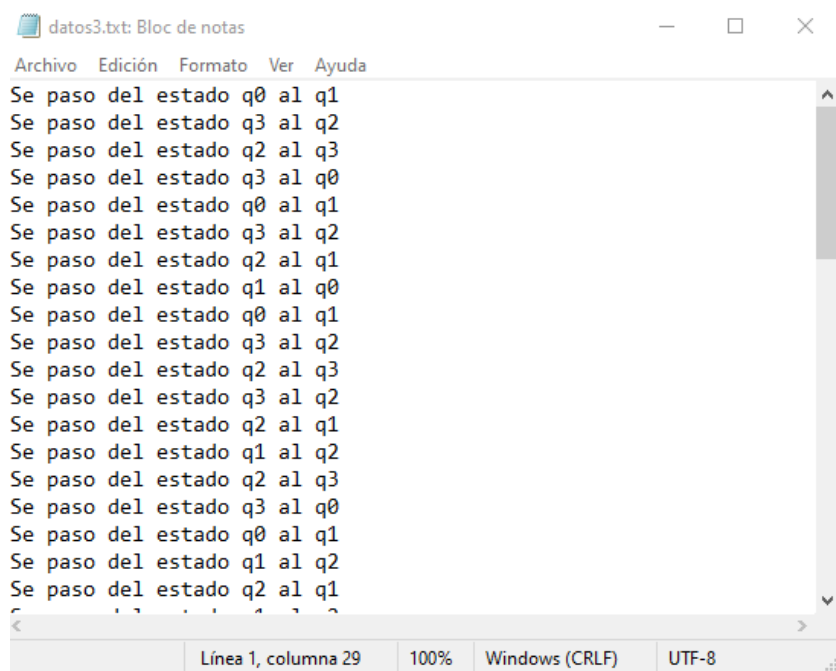


Figure 10: Se genera un Txt de los movimientos realizados

0.5 PROTOCOLO

```
import random
import math
from tkinter import*
#####

#Graficos

ventana = Tk()
canv = Canvas(ventana,width=800,height=300)
ventana.geometry("800x300")

ventana.title('Automata Paridad')

p= Label(ventana,text="Cadena a verificar. \n Cadena:").place(x=10,y=10)

#q0
p0= Label(ventana,text="Ready").place(x=50,y=105)

canv.create_oval(20,70,110,160, fill="blue")
```

```

canv.create_oval(35,85,95,145)
n1= Label(ventana,text="data in").place(x=185,y=35)

canv.create_oval(40,155,50,165, fill="black")
canv.create_oval(90,145,100,155, fill="black")

n2= Label(ventana,text="Start").place(x=25,y=185)

n2= Label(ventana,text="ack").place(x=185,y=175)

#q1
p1= Label(ventana,text="Sending").place(x=290,y=105)
canv.create_oval(270,70,360,160, fill="blue")

canv.create_oval(280,75,290,85, fill="black")

canv.create_oval(312,65,322,75, fill="black")

n1= Label(ventana,text="timeout").place(x=300,y=35)

#lineas

canv.create_line(35, 185, 45, 160, width=2, fill='black')

xy1 = 95, 60, 286, 105
canv.create_arc(xy1, start=0, extent=180, style="arc")

xy1 = 95, 125, 286, 170
canv.create_arc(xy1, start=0, extent=-180, style="arc")

xy2 = 316, 40, 356, 120
canv.create_arc(xy2, start=-25, extent=200, style="arc")

canv.place(x=0,y=0)
ventana.mainloop()

```

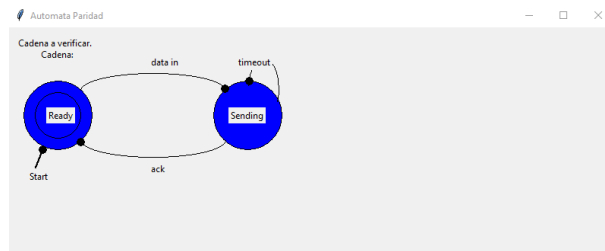


Figure 11: GRAFO

CONCLUSION

Para esta practica vimos como puede trabajar el autómata con respecto a una cadena de números binarios, como también determinar si es una cadena con imparidad o no. Por consiguiente en la realización de esta se tuvieron diferentes problemáticas que al final se resuelven, primero fue pensar en cómo ordenar los números binarios de tal forma que no se dieran saltos innecesarios o que se pegaran entre ellas, por ello usamos dos for de control como se menciona antes uno para crear la cadena y otra por el numero de cadenas a crear, entonces teniendo esto en cuenta se quito la idea de crear las cadenas en un archivo y después extraerlas para analizarlas una por una, optando por la idea de los dos for anidados podemos en un ciclo validar su imparidad antes de pasar a la siguiente cadena. Ahora tenemos los ficheros se pensaba que después de tener todas las cadenas extraer valores con imparidad o no, pero de igual forma en ese for anidado ya vamos almacenando si es imparidad o no, gracias a los contadores. Después de este análisis tuvimos algunos problemas para el arreglo de números pero solo fue un sencillo error ya que la cadena no entraba. Bien para la mejora de la practica podría quedar una implementación grafica con el código pero no de tantas cadenas si no con pocas para ver lo que pasa, o bien dado que es un programa que para mi maquina exige algo de proceso tarda la realización, sin embargo se concluye bien. Y finalmente se comprende mejor el autómata para la implementación.