



# Tecnológico de Monterrey

**Grupo:**

TC3002B.201

**Materia:**

Desarrollo de aplicaciones avanzadas de ciencias computacionales

## **Evidencia Final Compiladores: Desarrollo de herramienta de soporte al proceso de análisis de imágenes**

**Integrantes:**

Salomon Dabbah Beracha - A01028445

Marco Antonio Torres - A01025334

Martín Palomares García - A01066569

**Fecha de entrega:**

03/05/24

<u>Reglas de la gramática implementada</u>	<u>3</u>
<u>1. Precedencia de operadores</u>	<u>5</u>
<u>2. Llamadas a funciones</u>	<u>6</u>
<u>3. Asignación de variables</u>	<u>7</u>
<u>4. Implementación de flujos de imágenes y 5. Aplicación de filtros de Open CV</u>	<u>8</u>
<u>6. Cada una de las nuevas características implementadas</u>	<u>11</u>
<u>a. Aceptar archivos y ejecutar el contenido</u>	<u>11</u>
<u>b. Todas las tareas entregadas (independientemente de la calificación si existiese alguna)</u>	<u>12</u>
<u>c. Aceptar cualquier función de numpy para manejo de matrices como np.where, np.mean, np.std. Al menos 9 de ellas.</u>	<u>14</u>
<u>d. Implementación de visualización de histogramas con opencv</u>	<u>17</u>
<u>e. Implementación de un algoritmo complejo como herramienta en el lenguaje: CannyEdgeDetection</u>	<u>18</u>
<u>f. Implementación de pruebas automatizadas. Uso de un marco de TESTING para probar todas las características de tu gramática</u>	<u>18</u>

## Reglas de la gramática implementada

Rule 0 S' -> assignment  
Rule 1 assignment -> VARIABLE SETTO expression  
Rule 2 assignment -> VARIABLE SETTO flow  
Rule 3 flow -> VARIABLE CONNECT flow\_functions  
Rule 4 flow\_functions -> flow\_function\_call CONNECT flow\_functions  
Rule 5 flow\_functions -> flow\_function\_call  
Rule 6 flow\_function\_call -> VARIABLE LPAREN params RPAREN  
Rule 7 assignment -> expression  
Rule 8 expression -> expression PLUS term  
Rule 9 expression -> expression MINUS term  
Rule 10 expression -> term  
Rule 11 expression -> string  
Rule 12 string -> STRING  
Rule 13 term -> term TIMES exponent  
Rule 14 term -> term DIVIDE exponent  
Rule 15 term -> exponent  
Rule 16 exponent -> factor EXP factor  
Rule 17 exponent -> factor  
Rule 18 exponent -> LPAREN expression RPAREN  
Rule 19 factor -> NUMBER  
Rule 20 factor -> VARIABLE  
Rule 21 factor -> function\_call  
Rule 22 function\_call -> VARIABLE LPAREN RPAREN  
Rule 23 function\_call -> VARIABLE LPAREN params RPAREN  
Rule 24 params -> params COMMA expression  
Rule 25 params -> expression

## Terminales

COMMA	: 24
CONNECT	: 3 4
DIVIDE	: 14
EXP	: 16
LPAREN	: 6 18 22 23
MINUS	: 9
NUMBER	: 19
PLUS	: 8
RPAREN	: 6 18 22 23
SETTO	: 1 2
STRING	: 12
TIMES	: 13

VARIABLE : 1 2 3 6 20 22 23  
error :

## Descripción de las funciones implementadas como herramientas y accesorios a la gramática

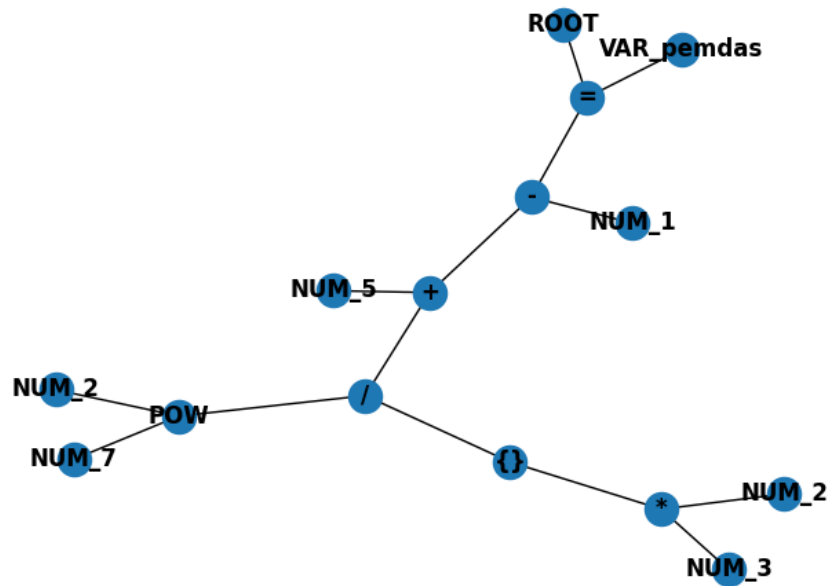
Función	Parámetro/s	Descripción
load_image	imagen	Carga una imagen al compilador y la lee.
save_image	nombre_del_archivo, imagen	Guarda la imagen en el dispositivo con el nombre seleccionado.
show_image	imagen	Despliega la imagen en el dispositivo.
search_cv2	nombre_de_la_funcion	Se usa para ver si una función existe en la librería de cv2, en caso de que exista, la función la regresa, de lo contrario arroja un error diciendo que no existe.
gen_matrix	filas, columnas, argumentos	Crea una matriz de x filas, por y columnas, y se le introducen los argumentos, es importante tener en cuenta que se tienen que introducir la cantidad de argumentos resultante de la multiplicación de x por y, por ejemplo, si la matriz sería de 3 x 3 tendría que tener 9 argumentos exactamente.
gen_vector	argumentos	Crea un vector que contiene los argumentos que se le proporcionen.
my_mean	argumentos	Obtiene la media de los argumentos proporcionados.
my_sum	argumentos	Obtiene la sumatoria de los argumentos proporcionados.
my_median	argumentos	Obtiene la mediana de los argumentos proporcionados.
my_std	argumentos	Obtiene la desviación estándar de los argumentos proporcionados.
my_max	argumentos	Obtiene el elemento más grande de los argumentos proporcionados.
my_min	argumentos	Obtiene el elemento más pequeño de los argumentos proporcionados.
my_sin	argumentos	Obtiene los senos de los argumentos proporcionados.
my_cos	argumentos	Obtiene los cosenos de los argumentos proporcionados.
my_tan	argumentos	Obtiene las tangentes de los argumentos proporcionados.
histogram	imagen	Convierte la imagen a escala de grises y calcula su histograma, que muestra la distribución de intensidades de píxeles de 0 (negro) a 255 (blanco).
canny_edge	imagen	Convierte la imagen a escala de grises y detecta los bordes utilizando el método de Canny. Luego, muestra la imagen original y la imagen de los bordes.
gray_scale	imagen	Carga la imagen y la convierte en blanco y negro.

## Demostraciones

### 1. Precedencia de operadores

$$5 + (3 * 2) / 7^2 - 1 = 202/49 = 4.1224489796$$

```
input>> pmdas = 5 + (3 * 2) / 7 ^ 2 - 1
From Node 14 []
From Node 1 []
From Node 2 []
From Node 3 []
From Node 4 [3, 2]
From Node 5 [6]
From Node 6 []
From Node 7 []
From Node 8 [7, 2]
From Node 9 [6, 49]
From Node 10 [5, 0.12244897959183673]
From Node 11 []
From Node 12 [5.122448979591836, 1]
From Node 13 ['pmdas', 4.122448979591836]
From Node 0 [4.122448979591836]
Result: 4.122448979591836
```



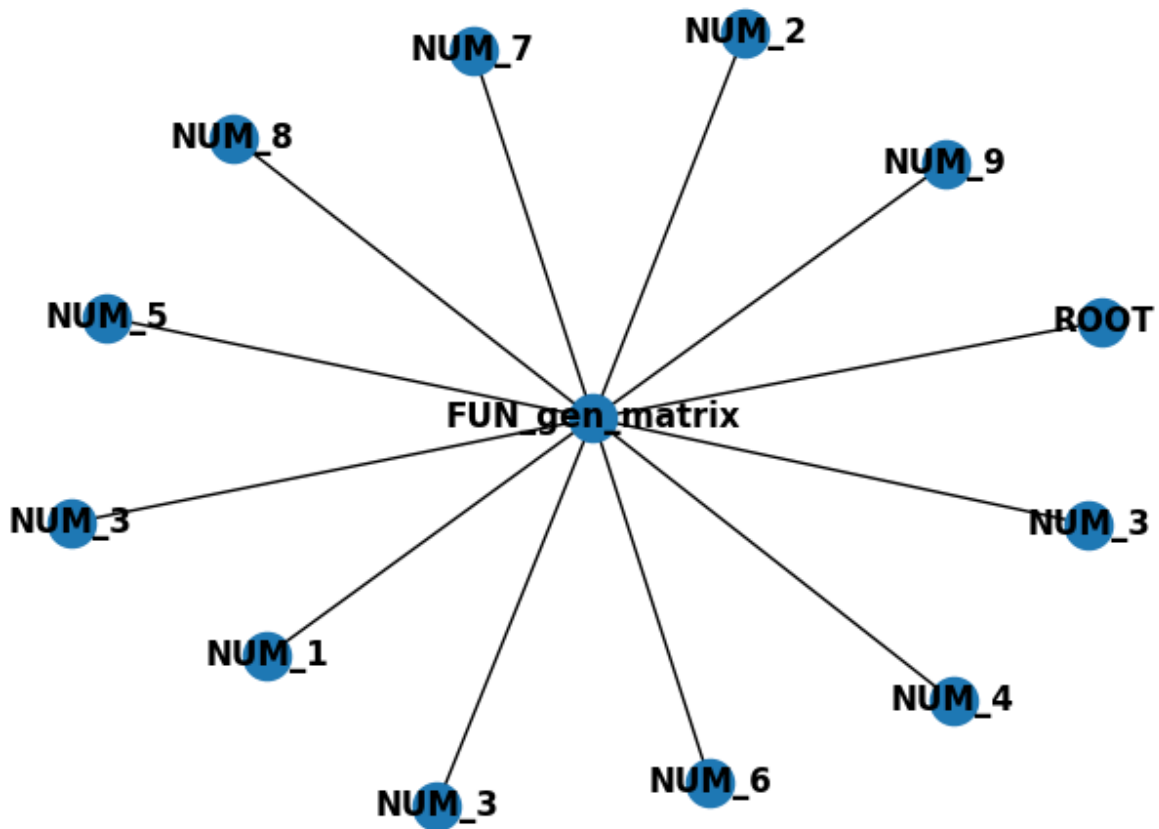
## 2. Llamadas a funciones

```

input>> gen_matrix(3,3,1,2,3,4,5,6,7,8,9)
[{'type': 'NUMBER', 'label': 'NUM_3', 'value': 3, 'counter': 1}, {'type': 'NUMBER', 'label': 'NUM_3', 'value': 3, 'counter': 2}, {'t
ype': 'NUMBER', 'label': 'NUM_1', 'value': 1, 'counter': 3}, {'type': 'NUMBER', 'label': 'NUM_2', 'value': 2, 'counter': 4}, {'type'
: 'NUMBER', 'label': 'NUM_3', 'value': 3, 'counter': 5}, {'type': 'NUMBER', 'label': 'NUM_4', 'value': 4, 'counter': 6}, {'type': 'N
UMBER', 'label': 'NUM_5', 'value': 5, 'counter': 7}, {'type': 'NUMBER', 'label': 'NUM_6', 'value': 6, 'counter': 8}, {'type': 'NUMBE
R', 'label': 'NUM_7', 'value': 7, 'counter': 9}, {'type': 'NUMBER', 'label': 'NUM_8', 'value': 8, 'counter': 10}, {'type': 'NUMBER',
'label': 'NUM_9', 'value': 9, 'counter': 11}]

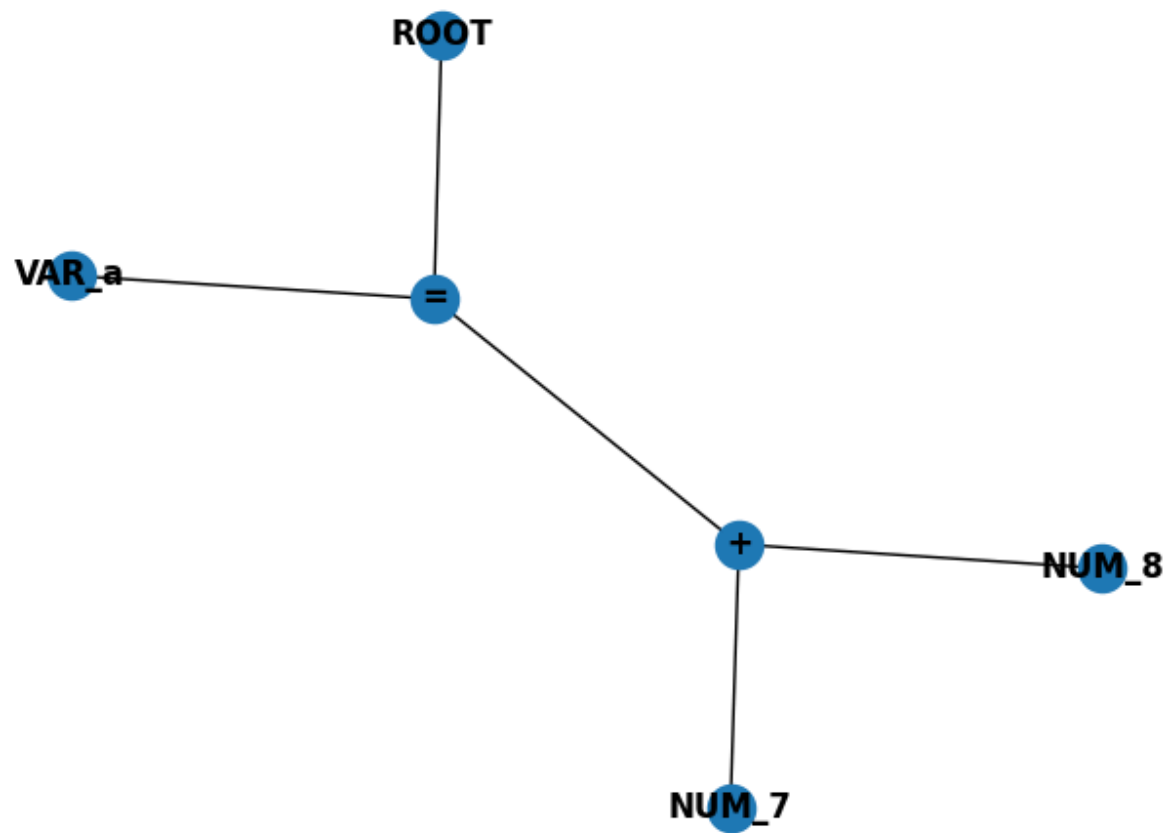
From Node 1 []
From Node 2 []
From Node 3 []
From Node 4 []
From Node 5 []
From Node 6 []
From Node 7 []
From Node 8 []
From Node 9 []
From Node 10 []
From Node 11 []
From Node 12 [3, 3, 1, 2, 3, 4, 5, 6, 7, 8, 9]
**** HERE Function call: gen_matrix ****
From Node 0 [array([[1, 2, 3],
[4, 5, 6],
[7, 8, 9]])]

```



### 3. Asignación de variables

```
input>> a = 7 + 8  
From Node 5 []  
From Node 1 []  
From Node 2 []  
From Node 3 [7, 8]  
From Node 4 ['a', 15]  
From Node 0 [15]  
Result: 15
```



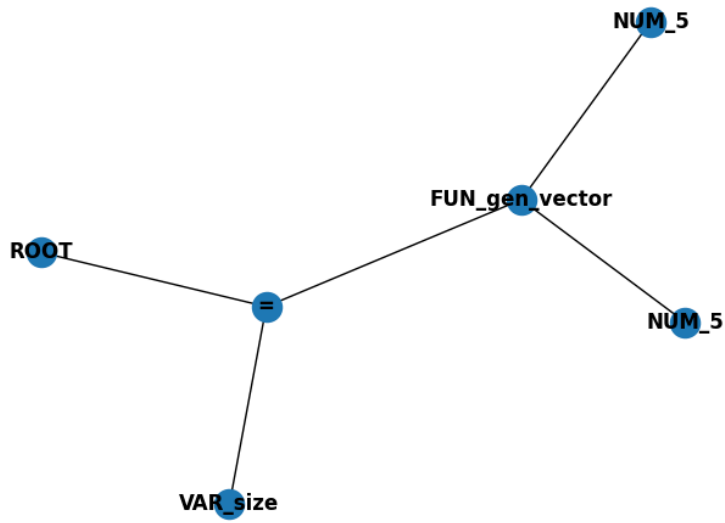
4. Implementación de flujos de imágenes y 5. Aplicación de filtros de Open CV



```

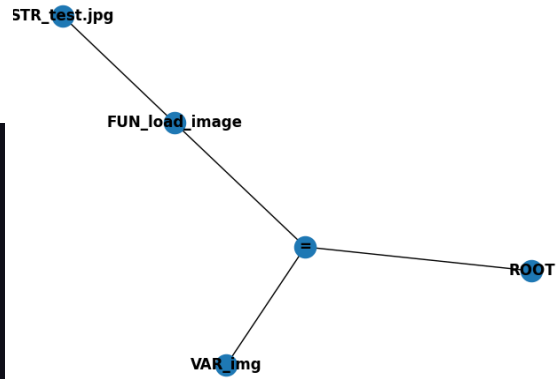
input>> size = gen_vector(5,5)
[{'type': 'NUMBER', 'label': 'NUM_5', 'value': 5, 'counter': 1}, {'type': 'NUMBER', 'label': 'NUM_5', 'value': 5, 'counter': 2}]
From Node 5 []
From Node 1 []
From Node 2 []
From Node 3 [5, 5]
**** HERE Function call: gen_vector ****
From Node 4 ['size', array([5, 5])]
From Node 0 [array([5, 5])]
  
```





```

input>> img = load_image("test.jpg")
[{'type': 'STRING', 'label': 'STR_test.jpg', 'value': 'test.jpg', 'counter': 1}]
From Node 4 []
From Node 1 []
From Node 2 ['test.jpg']
**** HERE Function call: load_image ****
From Node 3 ['img', array([[252, 243, 233],
                           [252, 243, 233],
                           ...],
                           [ 1, 2, 0],
                           [ 1, 2, 0],
                           [ 1, 2, 0]),
              [253, 244, 234],
              [253, 244, 234],
              [253, 244, 234],
              ...]]
  
```



```

graph TD
    ROOT((ROOT)) --- EQ((=))
    EQ --- ff_blur((ff_blur))
    EQ --- VAR_img2((VAR_img2))
    ff_blur --- VAR_size((VAR_size))
    ff_blur --- VAR_img((VAR_img))
  
```

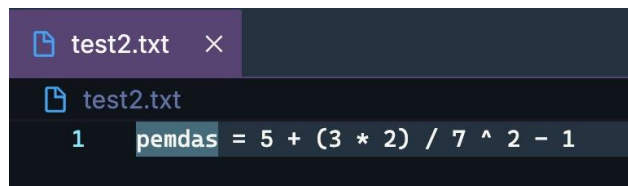
```

graph TD
    ROOT((ROOT)) --> FUN_show_image((FUN_show_image))
    FUN_show_image --> VAR_img2((VAR_img2))
  
```



6. Cada una de las nuevas características implementadas

a. Aceptar archivos y ejecutar el contenido



```
test2.txt ×  
test2.txt  
1 pmdas = 5 + (3 * 2) / 7 ^ 2 - 1
```

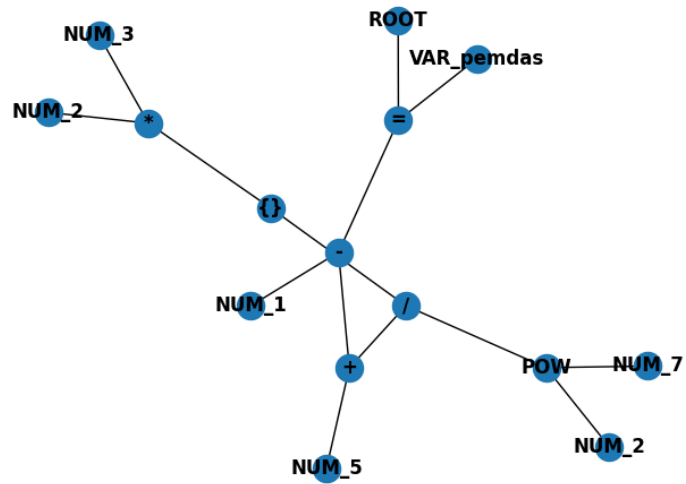


```
fileImport = True
```

```

From Node 14 []
From Node 1 []
From Node 2 []
From Node 3 []
From Node 4 [3, 2]
From Node 5 [6]
From Node 6 []
From Node 7 []
From Node 8 [7, 2]
From Node 9 [6, 49]
From Node 10 [5, 0.12244897959183673]
From Node 11 []
From Node 12 [5.122448979591836, 1]
From Node 13 ['pemdass', 4.122448979591836]
From Node 0 [4.122448979591836]
Result: 4.122448979591836

```



- b. Todas las tareas entregadas (independientemente de la calificación si existiese alguna)



### **Conjuntos - notacion**

4 de abr 10 pts

---



### **Creacion de lenguajes**

5 de abr 10 pts

---



### **Lenguajes específicos**

8 de abr 10 pts

---



### **Expresiones regulares**

8 de abr 10 pts

---



### **Automatas**

11 de abr 10 pts

---



### **Mi parser multiplica y divide**

12 de abr 10 pts

---



### **Gramáticas libres de contexto**

15 de abr 10 pts

---



### **Top Down Analyzer**

18 de abr 10 pts

---

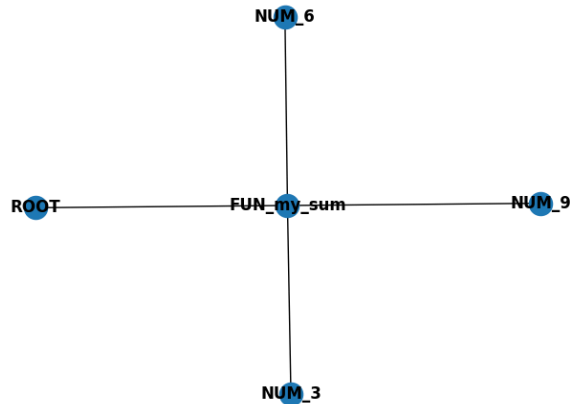


### **Bottom Up Analyzer**

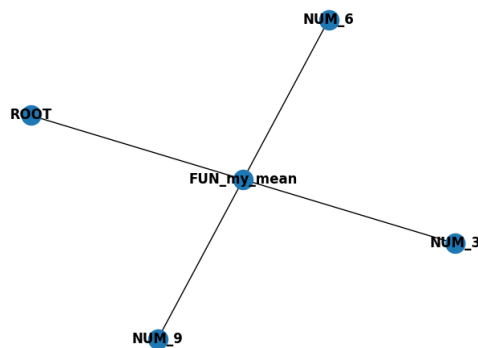
23 de abr 10 pts

- c. Aceptar cualquier función de numpy para manejo de matrices como np.where, np.mean, np.std. Al menos 9 de ellas.

```
input>> my_sum(3,6,9)
[{'type': 'NUMBER', 'label': 'NUM_3', 'value': 3, 'counter': 1}, {'type': 'NUMBER', 'label': 'NUM_6', 'value': 6, 'counter': 2}, {'t
ype': 'NUMBER', 'label': 'NUM_9', 'value': 9, 'counter': 3}]
From Node 1 []
From Node 2 []
From Node 3 []
From Node 4 [3, 6, 9]
**** HERE Function call: my_sum ****
From Node 0 [18]
```



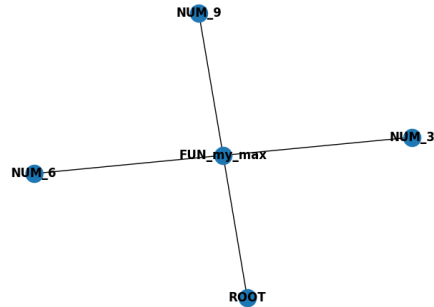
```
input>> my_mean(3,6,9)
[{'type': 'NUMBER', 'label': 'NUM_3', 'value': 3, 'counter': 1}, {'type': 'NUMBER', 'label': 'NUM_6', 'value': 6, 'counter': 2}, {'t
ype': 'NUMBER', 'label': 'NUM_9', 'value': 9, 'counter': 3}]
From Node 1 []
From Node 2 []
From Node 3 []
From Node 4 [3, 6, 9]
**** HERE Function call: my_mean ****
From Node 0 [6.0]
```



```

input>> my_max(3,6,9)
[{'type': 'NUMBER', 'label': 'NUM_3', 'value': 3, 'counter': 1}, {'type': 'NUMBER', 'label': 'NUM_6', 'value': 6, 'counter': 2}, {'type': 'NUMBER', 'label': 'NUM_9', 'value': 9, 'counter': 3}]
From Node 1 []
From Node 2 []
From Node 3 []
From Node 4 [3, 6, 9]
**** HERE Function call: my_max ****
From Node 0 [9]

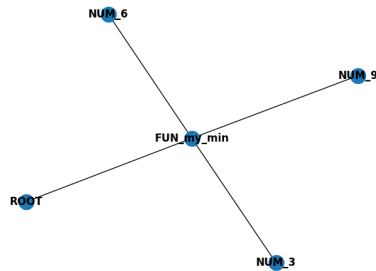
```



```

input>> my_min(3,6,9)
[{'type': 'NUMBER', 'label': 'NUM_3', 'value': 3, 'counter': 1}, {'type': 'NUMBER', 'label': 'NUM_6', 'value': 6, 'counter': 2}, {'type': 'NUMBER', 'label': 'NUM_9', 'value': 9, 'counter': 3}]
From Node 1 []
From Node 2 []
From Node 3 []
From Node 4 [3, 6, 9]
**** HERE Function call: my_min ****
From Node 0 [3]

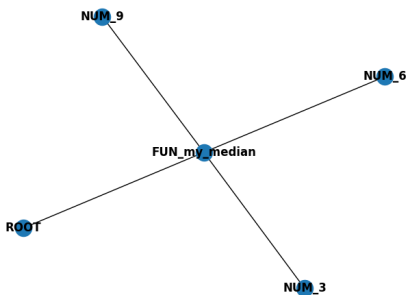
```



```

input>> my_median(3,6,9)
[{'type': 'NUMBER', 'label': 'NUM_3', 'value': 3, 'counter': 1}, {'type': 'NUMBER', 'label': 'NUM_6', 'value': 6, 'counter': 2}, {'type': 'NUMBER', 'label': 'NUM_9', 'value': 9, 'counter': 3}]
From Node 1 []
From Node 2 []
From Node 3 []
From Node 4 [3, 6, 9]
**** HERE Function call: my_median ****
From Node 0 [6.0]

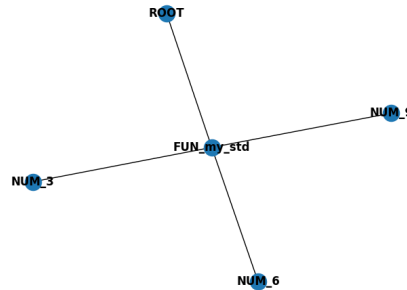
```



```

input>> my_std(3,6,9)
[{'type': 'NUMBER', 'label': 'NUM_3', 'value': 3, 'counter': 1}, {'type': 'NUMBER', 'label': 'NUM_6', 'value': 6, 'counter': 2}, {'t
ype': 'NUMBER', 'label': 'NUM_9', 'value': 9, 'counter': 3}]
From Node 1 []
From Node 2 []
From Node 3 []
From Node 4 [3, 6, 9]
**** HERE Function call: my_std ****
From Node 0 [2.449489742783178]

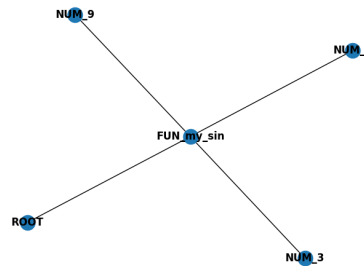
```



```

input>> my_sin(3,6,9)
[{'type': 'NUMBER', 'label': 'NUM_3', 'value': 3, 'counter': 1}, {'type': 'NUMBER', 'label': 'NUM_6', 'value': 6, 'counter': 2}, {'t
ype': 'NUMBER', 'label': 'NUM_9', 'value': 9, 'counter': 3}]
From Node 1 []
From Node 2 []
From Node 3 []
From Node 4 [3, 6, 9]
**** HERE Function call: my_sin ****
From Node 0 [array([ 0.14112001, -0.2794155 , 0.41211849])]

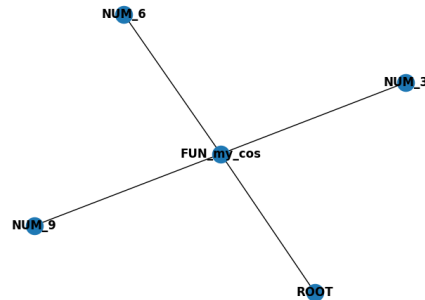
```



```

input>> my_cos(3,6,9)
[{'type': 'NUMBER', 'label': 'NUM_3', 'value': 3, 'counter': 1}, {'type': 'NUMBER', 'label': 'NUM_6', 'value': 6, 'counter': 2}, {'t
ype': 'NUMBER', 'label': 'NUM_9', 'value': 9, 'counter': 3}]
From Node 1 []
From Node 2 []
From Node 3 []
From Node 4 [3, 6, 9]
**** HERE Function call: my_cos ****
From Node 0 [array([-0.9899925 , 0.96017029, -0.91113026])]

```

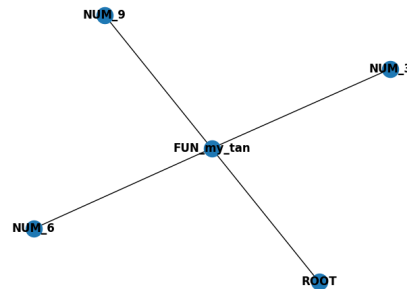




```

input>> my_tan(3,6,9)
[{'type': 'NUMBER', 'label': 'NUM_3', 'value': 3, 'counter': 1}, {'type': 'NUMBER', 'label': 'NUM_6', 'value': 6, 'counter': 2}, {'t
ype': 'NUMBER', 'label': 'NUM_9', 'value': 9, 'counter': 3}]
From Node 1 []
From Node 2 []
From Node 3 []
From Node 4 [3, 6, 9]
**** HERE Function call: my_tan ****
From Node 0 [array([-0.14254654, -0.29100619, -0.45231566])]

```



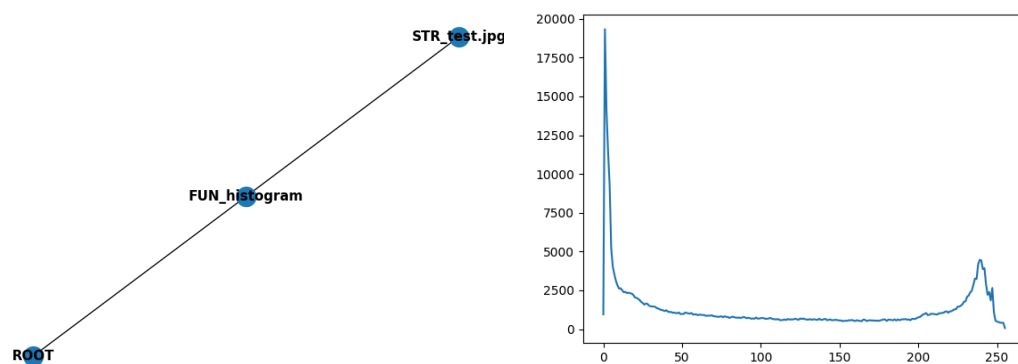
d. Implementación de visualización de histogramas con opencv



```

input>> histogram("test.jpg")
[{'type': 'STRING', 'label': 'STR_test.jpg', 'value': 'test.jpg', 'counter': 1}]
From Node 1 []
From Node 2 ['test.jpg']
**** HERE Function call: histogram ****
From Node 0 [None]

```



- e. Implementación de un algoritmo complejo como herramienta en el lenguaje:  
CannyEdgeDetection

```
input>> canny_edge("test.jpg")
[{'type': 'STRING', 'label': 'STR_test.jpg', 'value': 'test.jpg', 'counter': 1}]
From Node 1 []
From Node 2 ['test.jpg']
**** HERE Function call: canny_edge ****
From Node 0 [None]
```

Original Image



Edge Image



- f. Implementación de pruebas automatizadas. Uso de un marco de TESTING para probar todas las características de tu gramática

Run pytest

tests #8

Summary

Jobs

- test

Run details

Usage

Workflow file

test

succeeded 4 days ago in 18s

Search logs

- Set up job 0s
- Run actions/checkout@v2 1s
- Set up Python 0s
- Install dependencies 13s
- Run tests 2s
- Post Set up Python 0s
- Post Run actions/checkout@v2 0s
- Complete job 0s

<https://github.com/MartinPaGarcia/Desarrollo-de-aplicaciones-avanzadas/actions/runs/8841733482/job/24279282690>