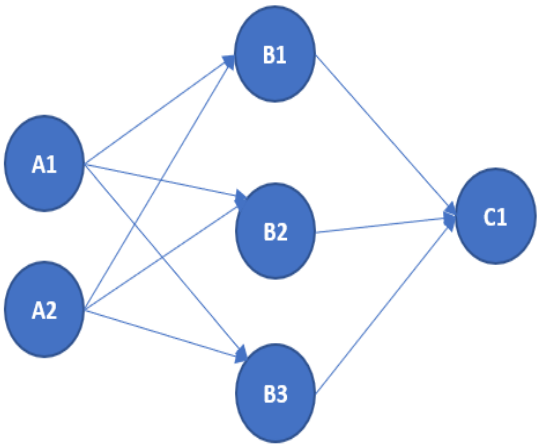
INITIALIZATION AND NETWORK CONFIGURATION

Most of us are unaware of neuroid networks. Therefore, the network configuration and initialization team chose to work on a simple network which makes it simple for everyone to understand.

The User will enter the input on the front-end page as no. of neuroid in the input layer, hidden layer and output layer and we take the input from vector and produce the matrices. So, Basically we are working on 2 neuroid in input layer,3 neuroid in the hidden layer and 1 neuroid in the output layer.




```
lst=[]
n=int(input("Enter no.of layers :"))
for i in range(0,n):
    m=int(input())
    lst.append(m)
print(lst)
maxdim = max(lst)
out = np.zeros((len(lst),maxdim))
for i in range(len(out)):
    out[i][:lst[i]]=1
print("The First Matrix is: \n")
print(out.T)
```

Here the matrix 1 is generated as follows which is the truth table of the network.

```
Enter no.of layers :3
2
3
1
[2, 3, 1]
The First Matrix is:
[[1. 1. 1.]
 [1. 1. 0.]
 [0. 1. 0.]]
```

```
dim = sum(lst)
second_matrix = np.zeros((dim,dim))
rownum = 0
col = lst[0]
for j in range(len(lst)-1):
r = lst[j]
c = lst[j+1]
for j in range(rownum,rownum+r):
second_matrix[j][col:col+c]=1
rownum = rownum+r
col = col+c
print("The Second Matrix is: \n")
print(second_matrix)
```

This is the second matrix generation which shows the connections in the network. If there is a Connection from one neuroid to another it represents as 1 and if no connection it represents as Zero.

```
umbr=np.random.uniform(0,1,size=(sum(lst),1))
beta=np.random.randint(0,10,size=(sum(lst),1))
kr=np.random.randint(-5,4,size=(sum(lst),1))
maxcount=np.random.randint(10,90,size=(sum(lst),1))
matrix=np.concatenate((umbr.T,beta.T,kr.T,maxcount.T)).T
print("The Third Matrix is: \n")
print(matrix)
```

Here, third matrix is generated. Umbr , Beta, kr and maxcount are generated randomly. And the size of the matrices will be based on the no.of neuroids in the network.

```
vector=np.random.randint(0,2,size=(sum(lst),sum(lst)))
count=0
rev_vector = vector[::-1]
for r in range(0,len(rev_vector[0]-1)):
    for c in range(0,len(rev_vector[0]-1)):
        if rev_vector[r][c] == 1:
            count = count+1
        if count > feedback:
            rev_vector[r][c] = 0

vector = rev_vector[::-1]
print("The Vector is : \n")
print(vector)
for x in range(len(vector)):
    for y in range(len(output[0])):
        for z in range(len(output)):
            if (output[x][z]==1.0):
                continue
```

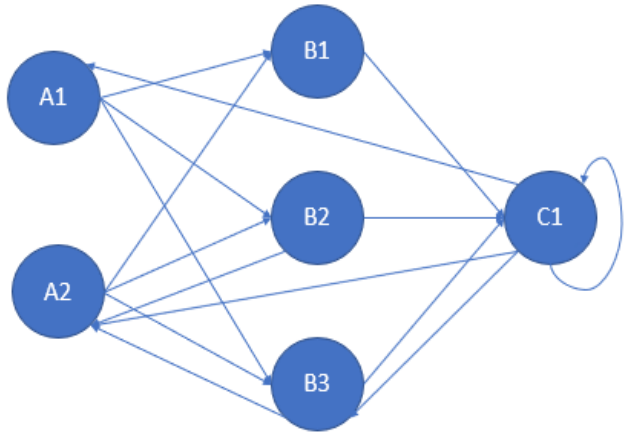
```
            else:
               mul=vector[x][z]*output[z][y]
            if mul>0.0:
               output[x][z]=1.0

      for i in range(2, len(output[0])-1):
         for j in range(2, len(output)-1):
            output[i][j]=0.0

      print("Matrix after Multiplication: \n")
      print(output)
```

Here, vector is multiplied with second matrix to get the back propagation's. The user enters the feedback percent between the 0 to 30.If the user enters 30% then the vector will have the 30% of the Matrix size ones in the vector. And then the matrix will have more number of back propagation's. If the user enters 0% then there will be no ones in the vector, therefore there will be no back propagations also. The best case we found is the percentage must be 10 to 20.For example, let us take 20% the vector is as follows.

```
[[0 0 0 0 0 0]
 [0 0 0 0 0 0]
 [0 0 0 0 0 0]
 [0 1 0 0 0 0]
 [0 1 0 0 0 1]
 [1 1 0 0 1 1]]
```

There are 7 ones in the matrix. The matrix with back propagation is as follows.

```
[[0. 0. 1. 1. 1. 0.]
 [0. 0. 1. 1. 1. 0.]
 [0. 0. 0. 0. 0. 1.]
 [0. 1. 0. 0. 0. 1.]
 [0. 1. 0. 0. 0. 1.]
 [1. 1. 0. 0. 1. 1.]]
```



This is how the neuroid network looks with back propagations.
After completion of generation of matrices,we need to make them functional.So,the data set team will provide us with AND gate inputs.We need to read them and work on it.

```
data = pd.read_excel("AND gate inputs.xlsx",sheet_name='Data 1')
A =data["Input A"]
ran = len(A)
B =data["Input B"]
weights1 = [0 for i in range(len(A))]
weights2 = [0 for i in range(len(B))]
plt.plot(A)
plt.plot(B)
plt.show()
A1 = Neuroid(umbr=matrix[0][0], beta=matrix[0][1], kr=matrix[0][2], maxcount=matrix[0][3], t=1)
A2 = Neuroid(umbr=matrix[1][0], beta=matrix[1][1], kr=matrix[1][2], maxcount=matrix[1][3], t=1)
y1 = A1.run_neuroid(inputs= A, weights = weights1)
y2 = A2.run_neuroid(inputs= B, weights = weights2)
plt.plot(y1)
plt.plot(y2)
plt.show()
B1 = Neuroid(umbr=matrix[2][0], beta=matrix[2][1], kr=matrix[2][2], maxcount=matrix[2][3], t=1)
inputs_combined = [x-y for x,y in zip(y1,y2)]
weights3 = [0 for i in range(len(inputs_combined))]
y3 = B1.run_neuroid(inputs=inputs_combined, weights=weights3)
B2 = Neuroid(umbr=matrix[3][0], beta=matrix[3][1], kr=matrix[3][2], maxcount=matrix[3][3], t=1)
inputs_combined = [x-y for x,y in zip(y1,y2)]
weights3 = [0 for i in range(len(inputs_combined))]
y4 = B2.run_neuroid(inputs=inputs_combined, weights=weights3)
B3 = Neuroid(umbr=matrix[4][0], beta=matrix[4][1], kr=matrix[4][2], maxcount=matrix[4][3], t=1)
inputs_combined = [x-y for x,y in zip(y1,y2)]
weights3 = [0 for i in range(len(inputs_combined))]
y5 = B3.run_neuroid(inputs=inputs_combined, weights=weights3)
plt.plot(y3)
plt.plot(y4)
plt.plot(y5)
plt.show()
C1 = Neuroid(umbr=matrix[5][0], beta=matrix[5][1], kr=matrix[5][2], maxcount=matrix[5][3], t=1)
inputs_combined = [x-y-z for x,y,z in zip(y3,y4,y5)]
weights3 = [0 for i in range(len(inputs_combined))]
y6 = C1.run_neuroid(inputs=inputs_combined, weights=weights3)
plt.plot(y6)
plt.show()
```

So, we are reading the data from the excel sheet and passing into our code. A1 is the first row of the third matrix and A2 is the second row of the third matrix and it works similarly for B1,B2,B3 and C1 also. The output from A1, A2 are taken as y1 and y2 and these y1 and y2 are passed as input to the B1,B2,B3 and the output from these are taken as y3, y4,y5 and sent as inputs to C1.The y6 is the combined value all of these.We are generating the plot for these y values.

The y6 values are as follows.
[0, -3.0, -3.0, -3.0, -3.0, -3.0, -3.0, -3.0, -3.0, -3.0, -3.0, -3.0, -3.0, -3.0, -3.0, -3.0, -3.0, -3.0, -0.6, -0.2727272727272727, -0.2727272727272727, -0.2727272727272727, -0.2727272727272727, -0.2727272727272727, -0.2727272727272727, -0.2727272727272727, -0.2727272727272727, -0.2727272727272727, -0.3333333333333333, -0.3333333333333333, -0.3333333333333333, -0.3333333333333333, -0.3333333333333333, -0.3333333333333333, -0.6, -0.6, -0.6, -0.6, -0.6, -0.6, -0.6, -0.6, -0.6, -0.6, -0.6, -0.6, -0.6, -0.2727272727272727, -0.2727272727272727, -0.2727272727272727, -0.2727272727272727, -0.2727272727272727, -0.2727272727272727, -0.6, -0.6, -0.6, -0.6, -0.6, -0.6, -0.6, -0.6, -0.6, -0.6, -0.6, -0.6, -0.6, -0.6, -0.6, -0.6, -0.6, -0.6]

These y6 values are sent to the Dataset team for the further process.