In [1]:
```
from keras.models import load_model, Sequential
from keras.layers import Dense
```

```
Using TensorFlow backend.
```

In [15]:
```
import h5py
import numpy as np
```

Test Average Case

In [11]:
```
x = [[5,7,4,3,7,8,9,7,6,4]]
x = np.asarray(x)
a = model.predict(x)
a = np.rint(a)
a
```

Out[11]:
```
array([[ 3.,  4.,  4.,  5.,  6.,  7.,  7.,  7.,  8.,  9.]], dtype=float32)
```

Test Worst Case

In [12]:
```
x = [[9,8,7,6,5,4,3,2,1,0]]
x = np.asarray(x)
a = model.predict(x)
a = np.rint(a)
a
```

Out[12]:
```
array([[-0.,  1.,  2.,  3.,  4.,  5.,  6.,  7.,  8.,  9.]], dtype=float32)
```

Define the Network as 'model'

In [3]:
```
model = Sequential()

model.add(Dense(10, input_shape = (10,), activation = 'relu'))
model.add(Dense(20, activation = 'relu'))
model.add(Dense(40, activation = 'relu'))
model.add(Dense(80, activation = 'relu'))
model.add(Dense(40, activation = 'relu'))
model.add(Dense(20, activation = 'relu'))
model.add(Dense(10))


model.compile(loss = 'mean_squared_error',
              optimizer = 'nadam')
```

Load the weights on the model

In [4]:
```
model.load_weights('version3.h5')
```

Define an array 'z' with 100, 10 digit random arrays for testing

In [6]:
```
z = np.random.randint(10, size =(100,10), dtype = 'int8')
```

In [7]:
```
z
```

Out[7]:
```
array([[5, 6, 9, 9, 8, 0, 6, 6, 7, 3],
       [5, 1, 5, 2, 6, 8, 2, 6, 7, 2],
       [7, 4, 5, 2, 9, 9, 1, 6, 2, 3],
       [9, 6, 0, 7, 0, 1, 6, 7, 8, 5],
       [7, 5, 6, 7, 0, 8, 4, 6, 2, 1],
       [5, 1, 1, 1, 6, 8, 5, 5, 6, 5],
       [2, 2, 7, 9, 5, 5, 8, 9, 5, 5],
       [8, 1, 3, 1, 5, 2, 8, 6, 3, 7],
       [1, 0, 4, 2, 0, 0, 2, 3, 7, 4],
       [4, 1, 7, 1, 5, 7, 5, 0, 0, 7],
       [1, 4, 0, 2, 8, 5, 1, 9, 2, 6],
       [6, 2, 7, 4, 9, 4, 1, 4, 1, 0],
       [5, 7, 4, 5, 8, 0, 9, 0, 8, 5],
       [5, 3, 2, 1, 7, 5, 4, 1, 9, 5],
       [7, 6, 8, 8, 0, 7, 0, 6, 7, 2],
       [3, 7, 7, 4, 0, 0, 8, 2, 2, 1],
       [4, 8, 5, 8, 2, 0, 9, 3, 4, 1],
       [4, 4, 3, 1, 6, 8, 7, 8, 2, 8],
       [5, 9, 7, 5, 7, 3, 8, 2, 9, 4],
       [4, 5, 8, 6, 0, 6, 7, 7, 4, 1],
       [5, 2, 9, 8, 9, 2, 2, 8, 4, 4],
       [1, 4, 6, 1, 1, 5, 3, 4, 7, 6],
       [3, 0, 6, 2, 6, 1, 2, 0, 3, 0],
       [1, 1, 0, 1, 2, 1, 8, 9, 6, 4],
       [2, 5, 2, 6, 2, 6, 1, 6, 1, 7],
       [3, 1, 9, 1, 2, 4, 5, 9, 8, 0],
       [8, 1, 6, 5, 9, 3, 8, 6, 8, 5],
       [7, 0, 2, 0, 9, 3, 5, 6, 4, 3],
       [3, 2, 1, 3, 5, 3, 8, 8, 4, 0],
       [1, 5, 5, 4, 0, 7, 1, 1, 4, 4],
       [9, 5, 6, 3, 5, 4, 0, 5, 6, 2],
       [7, 8, 6, 7, 2, 7, 1, 1, 8, 0],
       [6, 9, 8, 3, 0, 1, 7, 4, 2, 6],
       [3, 5, 0, 0, 9, 4, 6, 3, 4, 2],
       [8, 8, 8, 6, 6, 7, 9, 4, 6, 0],
       [2, 7, 3, 5, 4, 5, 9, 1, 3, 6],
       [0, 4, 7, 7, 1, 6, 2, 2, 3, 7],
       [0, 6, 6, 5, 7, 6, 9, 3, 1, 3],
       [5, 6, 5, 1, 3, 8, 3, 2, 5, 4],
       [5, 2, 3, 5, 8, 8, 4, 2, 2, 8],
       [8, 8, 7, 5, 1, 3, 2, 2, 3, 6],
       [5, 9, 1, 9, 5, 7, 1, 5, 3, 4],
       [1, 8, 1, 5, 9, 4, 1, 9, 6, 1],
       [9, 8, 0, 8, 8, 7, 2, 9, 5, 4],
       [3, 4, 2, 3, 7, 6, 1, 0, 8, 7],
       [2, 4, 3, 4, 8, 7, 2, 8, 5, 6],
       [3, 0, 8, 6, 0, 3, 7, 5, 2, 6],
       [6, 2, 6, 9, 5, 7, 7, 2, 4, 6],
       [4, 4, 5, 1, 3, 5, 9, 9, 0, 5],
       [2, 3, 1, 7, 9, 5, 3, 0, 5, 0],
       [9, 7, 1, 4, 9, 6, 9, 8, 7, 4],
       [8, 9, 3, 0, 2, 1, 6, 7, 8, 9],
       [2, 7, 9, 5, 4, 2, 8, 9, 5, 9],
       [1, 7, 5, 0, 1, 7, 5, 0, 7, 6],
       [7, 9, 0, 8, 8, 2, 9, 5, 3, 1],
```

```
[6, 2, 5, 7, 8, 6, 9, 0, 3, 4],
[6, 8, 6, 9, 2, 4, 2, 3, 4, 0],
[1, 8, 4, 0, 2, 4, 2, 6, 5, 3],
[3, 1, 0, 7, 5, 9, 7, 7, 6, 8],
[3, 8, 4, 4, 1, 8, 9, 6, 5, 5],
[0, 0, 9, 3, 5, 5, 0, 5, 4, 8],
[7, 2, 1, 4, 0, 8, 4, 1, 3, 4],
[2, 2, 0, 0, 1, 6, 4, 4, 5, 6],
[5, 8, 9, 5, 9, 6, 1, 1, 1, 9],
[5, 7, 6, 8, 0, 6, 4, 6, 8, 7],
[2, 5, 7, 2, 8, 6, 9, 2, 4, 8],
[2, 9, 3, 1, 7, 6, 1, 6, 2, 5],
[8, 7, 4, 4, 5, 1, 6, 6, 9, 7],
[5, 6, 8, 4, 8, 3, 6, 1, 0, 9],
[4, 8, 4, 0, 0, 3, 5, 1, 6, 8],
[4, 8, 7, 6, 0, 7, 5, 8, 9, 4],
[8, 3, 6, 2, 0, 2, 9, 1, 4, 2],
[8, 9, 3, 0, 6, 5, 1, 6, 6, 9],
[5, 5, 9, 4, 3, 6, 2, 8, 0, 7],
[8, 2, 1, 9, 1, 1, 9, 8, 6, 1],
[6, 2, 9, 8, 4, 8, 6, 1, 9, 7],
[9, 4, 8, 7, 3, 0, 1, 0, 6, 3],
[1, 0, 4, 2, 9, 0, 3, 7, 7, 4],
[0, 9, 3, 7, 3, 8, 8, 3, 4, 2],
[0, 4, 0, 0, 2, 8, 5, 4, 4, 7],
[2, 2, 0, 1, 9, 3, 3, 2, 6, 7],
[1, 8, 2, 3, 2, 9, 6, 7, 5, 4],
[6, 6, 8, 5, 2, 3, 8, 1, 1, 5],
[1, 4, 1, 3, 4, 0, 6, 1, 2, 1],
[1, 0, 4, 2, 2, 2, 3, 7, 6, 1],
[2, 9, 6, 7, 6, 1, 4, 3, 9, 1],
[5, 6, 0, 2, 4, 0, 0, 0, 8, 0],
[5, 3, 5, 9, 4, 9, 0, 4, 7, 0],
[7, 8, 6, 5, 0, 5, 5, 6, 4, 0],
[0, 4, 1, 1, 9, 5, 2, 0, 3, 9],
[8, 7, 6, 2, 8, 4, 5, 4, 5, 8],
[1, 9, 6, 8, 6, 2, 0, 3, 5, 8],
[4, 5, 0, 4, 8, 7, 9, 7, 9, 0],
[6, 3, 6, 6, 6, 2, 2, 4, 9, 4],
[3, 4, 5, 2, 7, 7, 8, 7, 6, 1],
[1, 7, 9, 9, 1, 6, 7, 6, 5, 5],
[8, 0, 2, 4, 5, 4, 9, 7, 6, 4],
[2, 9, 4, 2, 6, 0, 7, 4, 4, 0],
[7, 5, 7, 0, 4, 1, 6, 2, 1, 3],
[2, 5, 0, 8, 6, 8, 8, 1, 4, 6]], dtype=int8)
```

Predict the sorted output from the model

(np.rint is used to round up the results of the sorted array predicted)

In [30]:
```
predicted_output = model.predict(z)

predicted_output = np.rint(predicted_output)

predicted_output = predicted_output.astype('int8')

predicted_output
```

Out[30]:
```
array([[0, 3, 5, 6, 6, 6, 7, 8, 9, 9],
       [1, 2, 2, 3, 5, 5, 6, 6, 7, 8],
       [1, 2, 2, 3, 4, 5, 6, 7, 9, 9],
       [0, 0, 2, 4, 6, 6, 7, 7, 8, 9],
       [0, 1, 2, 4, 5, 6, 6, 7, 7, 8],
       [1, 1, 1, 4, 6, 5, 5, 6, 6, 8],
       [2, 2, 4, 5, 5, 6, 7, 8, 9, 9],
       [1, 1, 2, 3, 3, 5, 6, 7, 8, 8],
       [0, 0, 0, 1, 2, 2, 3, 4, 4, 7],
       [0, 0, 0, 2, 4, 5, 5, 6, 7, 7],
       [0, 1, 1, 2, 2, 4, 5, 6, 8, 9],
       [0, 1, 1, 2, 3, 4, 5, 6, 7, 9],
       [0, 0, 4, 5, 5, 6, 7, 8, 8, 9],
       [1, 1, 2, 3, 4, 5, 5, 6, 7, 9],
       [0, 0, 2, 5, 7, 7, 7, 7, 8, 8],
       [0, 0, 1, 2, 2, 3, 4, 7, 7, 8],
       [0, 1, 2, 3, 4, 4, 5, 8, 8, 9],
       [1, 2, 3, 4, 4, 6, 7, 8, 8, 8],
       [2, 3, 4, 5, 5, 7, 7, 8, 9, 9],
       [0, 1, 4, 5, 5, 6, 6, 7, 7, 8],
       [2, 2, 3, 4, 4, 5, 7, 9, 9, 9],
       [1, 1, 1, 3, 4, 4, 5, 6, 6, 7],
       [0, 0, 0, 1, 2, 2, 2, 3, 6, 6],
       [0, 1, 1, 1, 1, 2, 4, 6, 8, 9],
       [1, 1, 2, 2, 2, 5, 6, 6, 7, 7],
       [0, 1, 1, 2, 3, 4, 5, 8, 9, 9],
       [1, 3, 5, 5, 6, 6, 8, 8, 8, 9],
       [0, 0, 2, 3, 3, 4, 5, 6, 7, 9],
       [0, 1, 2, 3, 3, 4, 4, 5, 8, 8],
       [0, 1, 1, 2, 3, 4, 4, 4, 5, 7],
       [0, 2, 3, 4, 5, 5, 5, 6, 6, 9],
       [0, 1, 1, 2, 5, 7, 7, 7, 8, 8],
       [0, 1, 2, 3, 4, 6, 6, 7, 8, 9],
       [0, 0, 2, 3, 3, 4, 4, 5, 6, 9],
       [0, 4, 5, 5, 7, 7, 8, 8, 8, 9],
       [1, 2, 3, 3, 4, 5, 5, 6, 7, 9],
       [0, 1, 2, 2, 3, 4, 6, 7, 7, 7],
       [0, 1, 2, 4, 5, 6, 6, 6, 7, 9],
       [1, 2, 3, 3, 4, 4, 5, 5, 6, 8],
       [2, 2, 2, 3, 4, 4, 6, 7, 8, 8],
       [1, 2, 2, 3, 3, 5, 6, 7, 8, 8],
       [1, 1, 3, 4, 4, 5, 5, 7, 9, 9],
       [1, 1, 1, 2, 4, 5, 6, 8, 9, 9],
       [0, 2, 3, 5, 7, 8, 8, 8, 9, 9],
       [0, 1, 2, 3, 3, 4, 6, 7, 7, 8],
       [2, 2, 3, 4, 4, 5, 6, 7, 8, 8],
       [0, 0, 2, 3, 3, 5, 6, 6, 7, 8],
       [2, 2, 4, 5, 6, 6, 7, 7, 7, 9],
       [0, 1, 3, 4, 5, 5, 4, 5, 9, 9],
```

```
[0, 0, 1, 2, 3, 3, 5, 6, 7, 9],
[1, 4, 4, 5, 7, 7, 8, 9, 9, 9],
[0, 1, 2, 3, 6, 7, 8, 8, 9, 9],
[2, 2, 4, 5, 5, 7, 8, 9, 9, 9],
[0, 0, 0, 2, 4, 5, 6, 7, 7, 7],
[0, 1, 2, 3, 5, 7, 8, 8, 9, 9],
[0, 2, 3, 4, 5, 6, 6, 7, 8, 9],
[0, 2, 3, 3, 4, 4, 5, 7, 8, 9],
[0, 1, 2, 2, 3, 4, 4, 5, 6, 8],
[0, 1, 3, 5, 6, 7, 7, 7, 8, 9],
[1, 3, 4, 4, 4, 5, 6, 8, 8, 9],
[0, 0, 1, 2, 4, 5, 5, 5, 8, 9],
[0, 1, 1, 2, 3, 4, 4, 4, 7, 8],
[0, 0, 1, 2, 2, 4, 4, 5, 6, 6],
[1, 0, 2, 4, 5, 6, 8, 9, 9, 9],
[0, 4, 5, 6, 6, 6, 7, 8, 8, 8],
[2, 2, 2, 4, 5, 6, 7, 8, 8, 9],
[1, 1, 2, 2, 3, 5, 6, 6, 7, 9],
[1, 3, 4, 5, 6, 6, 7, 7, 8, 9],
[0, 1, 3, 4, 5, 6, 7, 8, 8, 9],
[0, 0, 1, 3, 4, 4, 5, 6, 8, 8],
[0, 3, 5, 5, 6, 7, 7, 8, 8, 9],
[0, 1, 2, 2, 2, 3, 4, 6, 8, 9],
[0, 1, 3, 5, 6, 6, 6, 8, 9, 9],
[0, 2, 3, 4, 5, 5, 6, 7, 8, 9],
[1, 1, 1, 1, 2, 6, 8, 9, 9, 9],
[1, 2, 4, 6, 6, 7, 8, 8, 9, 9],
[0, 0, 1, 3, 3, 4, 6, 7, 8, 9],
[0, 0, 1, 2, 3, 3, 5, 6, 7, 9],
[0, 2, 3, 3, 3, 4, 7, 8, 8, 9],
[0, 0, 0, 2, 4, 4, 4, 5, 7, 8],
[0, 1, 2, 2, 2, 2, 4, 6, 7, 9],
[1, 2, 2, 3, 4, 5, 6, 7, 8, 9],
[1, 1, 2, 3, 5, 5, 6, 6, 8, 8],
[0, 1, 1, 1, 1, 2, 3, 4, 4, 6],
[0, 1, 1, 2, 2, 2, 3, 4, 6, 7],
[1, 1, 2, 3, 4, 6, 6, 7, 9, 9],
[0, 0, 0, 0, 1, 2, 4, 5, 6, 8],
[0, 0, 3, 4, 4, 5, 5, 7, 9, 9],
[0, 0, 4, 6, 5, 5, 6, 6, 7, 8],
[0, 0, 1, 1, 2, 3, 4, 5, 9, 9],
[2, 3, 4, 5, 5, 6, 7, 8, 8, 8],
[0, 1, 2, 3, 5, 6, 7, 8, 8, 9],
[0, 0, 3, 5, 5, 7, 8, 8, 9, 9],
[2, 3, 3, 4, 4, 6, 6, 6, 6, 9],
[1, 2, 3, 4, 5, 6, 7, 7, 7, 8],
[1, 1, 5, 6, 6, 6, 6, 7, 9, 9],
[0, 2, 4, 4, 4, 5, 6, 7, 8, 9],
[0, 0, 1, 3, 3, 4, 5, 6, 7, 9],
[0, 1, 1, 2, 3, 4, 5, 6, 7, 7],
[0, 1, 2, 4, 5, 6, 6, 7, 8, 8]], dtype=int8)
```

Use np.sort to sort the same array 'z' using the numpy library

In [31]:
```
expected_output = np.sort(z)
expected_output
```

Out[31]:
```
array([[0, 3, 5, 6, 6, 6, 7, 8, 9, 9],
       [1, 2, 2, 2, 5, 5, 6, 6, 7, 8],
       [1, 2, 2, 3, 4, 5, 6, 7, 9, 9],
       [0, 0, 1, 5, 6, 6, 7, 7, 8, 9],
       [0, 1, 2, 4, 5, 6, 6, 7, 7, 8],
       [1, 1, 1, 5, 5, 5, 5, 6, 6, 8],
       [2, 2, 5, 5, 5, 5, 7, 8, 9, 9],
       [1, 1, 2, 3, 3, 5, 6, 7, 8, 8],
       [0, 0, 0, 1, 2, 2, 3, 4, 4, 7],
       [0, 0, 1, 1, 4, 5, 5, 7, 7, 7],
       [0, 1, 1, 2, 2, 4, 5, 6, 8, 9],
       [0, 1, 1, 2, 4, 4, 4, 6, 7, 9],
       [0, 0, 4, 5, 5, 5, 7, 8, 8, 9],
       [1, 1, 2, 3, 4, 5, 5, 5, 7, 9],
       [0, 0, 2, 6, 6, 7, 7, 7, 8, 8],
       [0, 0, 1, 2, 2, 3, 4, 7, 7, 8],
       [0, 1, 2, 3, 4, 4, 5, 8, 8, 9],
       [1, 2, 3, 4, 4, 6, 7, 8, 8, 8],
       [2, 3, 4, 5, 5, 7, 7, 8, 9, 9],
       [0, 1, 4, 4, 5, 6, 6, 7, 7, 8],
       [2, 2, 2, 4, 4, 5, 8, 8, 9, 9],
       [1, 1, 1, 3, 4, 4, 5, 6, 6, 7],
       [0, 0, 0, 1, 2, 2, 3, 3, 6, 6],
       [0, 1, 1, 1, 1, 2, 4, 6, 8, 9],
       [1, 1, 2, 2, 2, 5, 6, 6, 6, 7],
       [0, 1, 1, 2, 3, 4, 5, 8, 9, 9],
       [1, 3, 5, 5, 6, 6, 8, 8, 8, 9],
       [0, 0, 2, 3, 3, 4, 5, 6, 7, 9],
       [0, 1, 2, 3, 3, 3, 4, 5, 8, 8],
       [0, 1, 1, 1, 4, 4, 4, 5, 5, 7],
       [0, 2, 3, 4, 5, 5, 5, 6, 6, 9],
       [0, 1, 1, 2, 6, 7, 7, 7, 8, 8],
       [0, 1, 2, 3, 4, 6, 6, 7, 8, 9],
       [0, 0, 2, 3, 3, 4, 4, 5, 6, 9],
       [0, 4, 6, 6, 6, 7, 8, 8, 8, 9],
       [1, 2, 3, 3, 4, 5, 5, 6, 7, 9],
       [0, 1, 2, 2, 3, 4, 6, 7, 7, 7],
       [0, 1, 3, 3, 5, 6, 6, 6, 7, 9],
       [1, 2, 3, 3, 4, 5, 5, 5, 6, 8],
       [2, 2, 2, 3, 4, 5, 5, 8, 8, 8],
       [1, 2, 2, 3, 3, 5, 6, 7, 8, 8],
       [1, 1, 3, 4, 5, 5, 5, 7, 9, 9],
       [1, 1, 1, 1, 4, 5, 6, 8, 9, 9],
       [0, 2, 4, 5, 7, 8, 8, 8, 9, 9],
       [0, 1, 2, 3, 3, 4, 6, 7, 7, 8],
       [2, 2, 3, 4, 4, 5, 6, 7, 8, 8],
       [0, 0, 2, 3, 3, 5, 6, 6, 7, 8],
       [2, 2, 4, 5, 6, 6, 6, 7, 7, 9],
       [0, 1, 3, 4, 4, 5, 5, 5, 9, 9],
       [0, 0, 1, 2, 3, 3, 5, 5, 7, 9],
       [1, 4, 4, 6, 7, 7, 8, 9, 9, 9],
       [0, 1, 2, 3, 6, 7, 8, 8, 9, 9],
       [2, 2, 4, 5, 5, 7, 8, 9, 9, 9],
       [0, 0, 1, 1, 5, 5, 6, 7, 7, 7],
```

```
[0, 1, 2, 3, 5, 7, 8, 8, 9, 9],
[0, 2, 3, 4, 5, 6, 6, 7, 8, 9],
[0, 2, 2, 3, 4, 4, 6, 6, 8, 9],
[0, 1, 2, 2, 3, 4, 4, 5, 6, 8],
[0, 1, 3, 5, 6, 7, 7, 7, 8, 9],
[1, 3, 4, 4, 5, 5, 6, 8, 8, 9],
[0, 0, 0, 3, 4, 5, 5, 5, 8, 9],
[0, 1, 1, 2, 3, 4, 4, 4, 7, 8],
[0, 0, 1, 2, 2, 4, 4, 5, 6, 6],
[1, 1, 1, 5, 5, 6, 8, 9, 9, 9],
[0, 4, 5, 6, 6, 6, 7, 7, 8, 8],
[2, 2, 2, 4, 5, 6, 7, 8, 8, 9],
[1, 1, 2, 2, 3, 5, 6, 6, 7, 9],
[1, 4, 4, 5, 6, 6, 7, 7, 8, 9],
[0, 1, 3, 4, 5, 6, 6, 8, 8, 9],
[0, 0, 1, 3, 4, 4, 5, 6, 8, 8],
[0, 4, 4, 5, 6, 7, 7, 8, 8, 9],
[0, 1, 2, 2, 2, 3, 4, 6, 8, 9],
[0, 1, 3, 5, 6, 6, 6, 8, 9, 9],
[0, 2, 3, 4, 5, 5, 6, 7, 8, 9],
[1, 1, 1, 1, 2, 6, 8, 8, 9, 9],
[1, 2, 4, 6, 6, 7, 8, 8, 9, 9],
[0, 0, 1, 3, 3, 4, 6, 7, 8, 9],
[0, 0, 1, 2, 3, 4, 4, 7, 7, 9],
[0, 2, 3, 3, 3, 4, 7, 8, 8, 9],
[0, 0, 0, 2, 4, 4, 4, 5, 7, 8],
[0, 1, 2, 2, 2, 3, 3, 6, 7, 9],
[1, 2, 2, 3, 4, 5, 6, 7, 8, 9],
[1, 1, 2, 3, 5, 5, 6, 6, 8, 8],
[0, 1, 1, 1, 1, 2, 3, 4, 4, 6],
[0, 1, 1, 2, 2, 2, 3, 4, 6, 7],
[1, 1, 2, 3, 4, 6, 6, 7, 9, 9],
[0, 0, 0, 0, 0, 2, 4, 5, 6, 8],
[0, 0, 3, 4, 4, 5, 5, 7, 9, 9],
[0, 0, 4, 5, 5, 5, 6, 6, 7, 8],
[0, 0, 1, 1, 2, 3, 4, 5, 9, 9],
[2, 4, 4, 5, 5, 6, 7, 8, 8, 8],
[0, 1, 2, 3, 5, 6, 6, 8, 8, 9],
[0, 0, 4, 4, 5, 7, 7, 8, 9, 9],
[2, 2, 3, 4, 4, 6, 6, 6, 6, 9],
[1, 2, 3, 4, 5, 6, 7, 7, 7, 8],
[1, 1, 5, 5, 6, 6, 7, 7, 9, 9],
[0, 2, 4, 4, 4, 5, 6, 7, 8, 9],
[0, 0, 2, 2, 4, 4, 4, 6, 7, 9],
[0, 1, 1, 2, 3, 4, 5, 6, 7, 7],
[0, 1, 2, 4, 5, 6, 6, 8, 8, 8]], dtype=int8)
```

Compute 'Error' between the 'expected_output' and the 'predicted_output'

In [33]:
```
error = expected_output - predicted_output

error = error**2

error
```

Out[33]:
```
array([[0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
       [0, 0, 0, 1, 0, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
       [0, 0, 1, 1, 0, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
       [0, 0, 0, 1, 1, 0, 0, 0, 0, 0],
       [0, 0, 1, 0, 0, 1, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
       [0, 0, 1, 1, 0, 0, 0, 1, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 1, 0, 1, 0, 0, 0],
       [0, 0, 0, 0, 0, 1, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 1, 0, 0],
       [0, 0, 0, 1, 1, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 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, 0],
       [0, 0, 1, 0, 0, 0, 1, 1, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0, 1, 0, 0, 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, 0, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0, 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, 0, 0, 1, 1, 0, 0, 1, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 1, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
       [0, 0, 1, 1, 1, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
       [0, 0, 1, 1, 0, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 1, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 1, 1, 1, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 1, 0, 0, 0, 0, 0],
       [0, 0, 0, 1, 0, 0, 0, 0, 0, 0],
       [0, 0, 1, 0, 0, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 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, 0, 0, 1, 0, 1, 0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 1, 0, 0],
       [0, 0, 0, 1, 0, 0, 0, 0, 0, 0],
```

```
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
       [0, 0, 1, 1, 1, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
       [0, 0, 1, 0, 0, 0, 1, 1, 0, 0],
       [0, 0, 0, 0, 0, 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],
       [0, 0, 1, 1, 0, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
       [0, 1, 1, 1, 0, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 1, 0, 0],
       [0, 0, 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, 0, 0, 0],
       [0, 0, 0, 0, 0, 0, 1, 0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
       [0, 1, 1, 0, 0, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 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, 0, 0, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 1, 1, 1, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 1, 1, 0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 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],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
       [0, 0, 0, 1, 0, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
       [0, 1, 0, 0, 0, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0, 1, 0, 0, 0],
       [0, 0, 1, 1, 0, 0, 1, 0, 0, 0],
       [0, 1, 0, 0, 0, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
       [0, 0, 0, 1, 0, 0, 1, 0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
       [0, 0, 1, 1, 1, 0, 1, 0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 1, 0, 0]], dtype=int8)
```

In [34]:
```
n = np.count_nonzero(error)
n
```

Out[34]: 82

Out of 1000 numbers in the expected (correct) sort, 82 are erroneous by a margin of 1.

However, this could also be because of 'rint' not rounding numbers perfectly.

In [ ]: