In [1]:

```
import os
import cv2
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

In [2]:

```
dataset = []
```

In [3]:

```
dataset = []  # List to store image and label pairs

# List of folder paths
folder_paths = [
    "/home/mirza/Desktop/Navttac/ML-1st/Fruits360/fruits-360-original-size/fruits
    "/home/mirza/Desktop/Navttac/ML-1st/Fruits360/fruits-360-original-size/fruits
    "/home/mirza/Desktop/Navttac/ML-1st/Fruits360/fruits-360-original-size/fruits
    "/home/mirza/Desktop/Navttac/ML-1st/Fruits360/fruits-360-original-size/fruits
    "/home/mirza/Desktop/Navttac/ML-1st/Fruits360/fruits-360-original-size/fruits
    "/home/mirza/Desktop/Navttac/ML-1st/Fruits360/fruits-360-original-size/fruits
    "/home/mirza/Desktop/Navttac/ML-1st/Fruits360/fruits-360-original-size/fruits
   "/home/mirza/Desktop/Navttac/ML-1st/Fruits360/fruits-360-original-size/fruits-
    "/home/mirza/Desktop/Navttac/ML-1st/Fruits360/fruits-360-original-size/fruits
    "/home/mirza/Desktop/Navttac/ML-1st/Fruits360/fruits-360-original-size/fruits
    "/home/mirza/Desktop/Navttac/ML-1st/Fruits360/fruits-360-original-size/fruits
    "/home/mirza/Desktop/Navttac/ML-1st/Fruits360/fruits-360-original-size/fruits
    "/home/mirza/Desktop/Navttac/ML-1st/Fruits360/fruits-360-original-size/fruits
    "/home/mirza/Desktop/Navttac/ML-1st/Fruits360/fruits-360-original-size/fruits
    "/home/mirza/Desktop/Navttac/ML-1st/Fruits360/fruits-360-original-size/fruits
    "/home/mirza/Desktop/Navttac/ML-1st/Fruits360/fruits-360-original-size/fruits
     "/home/mirza/Desktop/Navttac/ML-1st/Fruits360/fruits-360-original-size/fruit
     "/home/mirza/Desktop/Navttac/ML-1st/Fruits360/fruits-360-original-size/fruit
     "/home/mirza/Desktop/Navttac/ML-1st/Fruits360/fruits-360-original-size/fruit
     "/home/mirza/Desktop/Navttac/ML-1st/Fruits360/fruits-360-original-size/fruit
     "/home/mirza/Desktop/Navttac/ML-1st/Fruits360/fruits-360-original-size/fruit
     "/home/mirza/Desktop/Navttac/ML-1st/Fruits360/fruits-360-original-size/fruit
     "/home/mirza/Desktop/Navttac/ML-1st/Fruits360/fruits-360-original-size/fruit
     "/home/mirza/Desktop/Navttac/ML-1st/Fruits360/fruits-360-original-size/fruit

    # Add the rest of the folder paths here
]
```

In [4]:

```
# Iterate over the folder paths
for i in folder_paths:
    folder_name = os.path.basename(i)

    # Iterate over the images in the subdirectory
    for file_name in os.listdir(i):
        image_path = os.path.join(i, file_name)

        if os.path.isfile(image_path):  # Only consider files
            # Load the image using OpenCV
            image = cv2.imread(image_path, cv2.IMREAD_GRAYSCALE)

            # If the image was successfully loaded
            if image is not None:
                # Resize the grayscale image to 250X250 pixels
                resized_image = cv2.resize(image, (250, 250))

                # Flatten the image and append each pixel as a separate feature a
                flattened_image = resized_image.flatten().tolist()
                dataset.append(flattened_image + [folder_name])
```

In [5]:

```
"""Convert the dataset to a pandas DataFrame"""
df = pd.DataFrame(dataset, columns=[f'pixel_{i+1}' for i in range(250*250)] + ['l

"""Print the DataFrame"""
df
```

Out[5]:

	pixel_1	pixel_2	pixel_3	pixel_4	pixel_5	pixel_6	pixel_7	pixel_8	pixel_9	pixel_10	.
0	255	255	255	255	255	255	255	255	255	255	.
1	255	255	255	255	255	255	255	255	255	255	.
2	255	255	255	255	255	255	255	255	255	255	.
3	255	255	255	255	255	255	255	255	255	255	.
4	255	255	255	255	255	255	255	255	255	255	.
...	...	...	...	...	...	...	...	...	...	...	.
6162	255	255	255	255	255	255	255	255	255	255	.
6163	255	255	255	255	255	255	255	255	255	255	.
6164	255	255	255	255	255	255	255	255	255	255	.
6165	255	255	255	255	255	255	255	255	255	255	.
6166	255	255	255	255	255	255	255	255	255	255	.

6167 rows × 62501 columns

In [6]:

```
# Extract the last column (label column) from the DataFrame
label_column = df.iloc[:, -1]

# Count the occurrences of each unique label

```

In [7]:

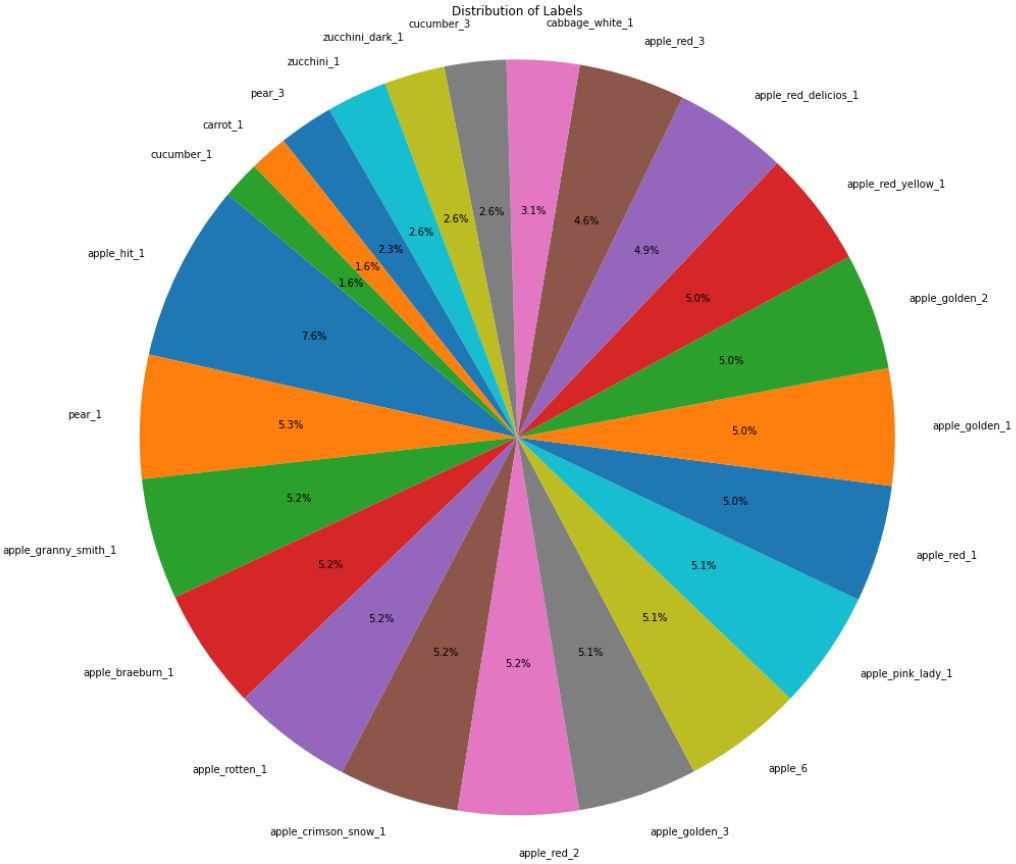
```
label_counts = label_column.value_counts()
label_counts
```

Out[7]:

```
apple_hit_1             468
pear_1                  326
apple_granny_smith_1    320
apple_braeburn_1        320
apple_rotten_1          319
apple_crimson_snow_1    318
apple_red_2             318
apple_golden_3          316
apple_6                 315
apple_pink_lady_1       313
apple_red_1             309
apple_golden_1          308
apple_golden_2          308
apple_red_yellow_1      308
apple_red_delicios_1    300
apple_red_3             281
cabbage_white_1         192
cucumber_3              163
zucchini_dark_1         160
zucchini_1              160
pear_3                  144
carrot_1                101
cucumber_1              100
Name: label, dtype: int64
```

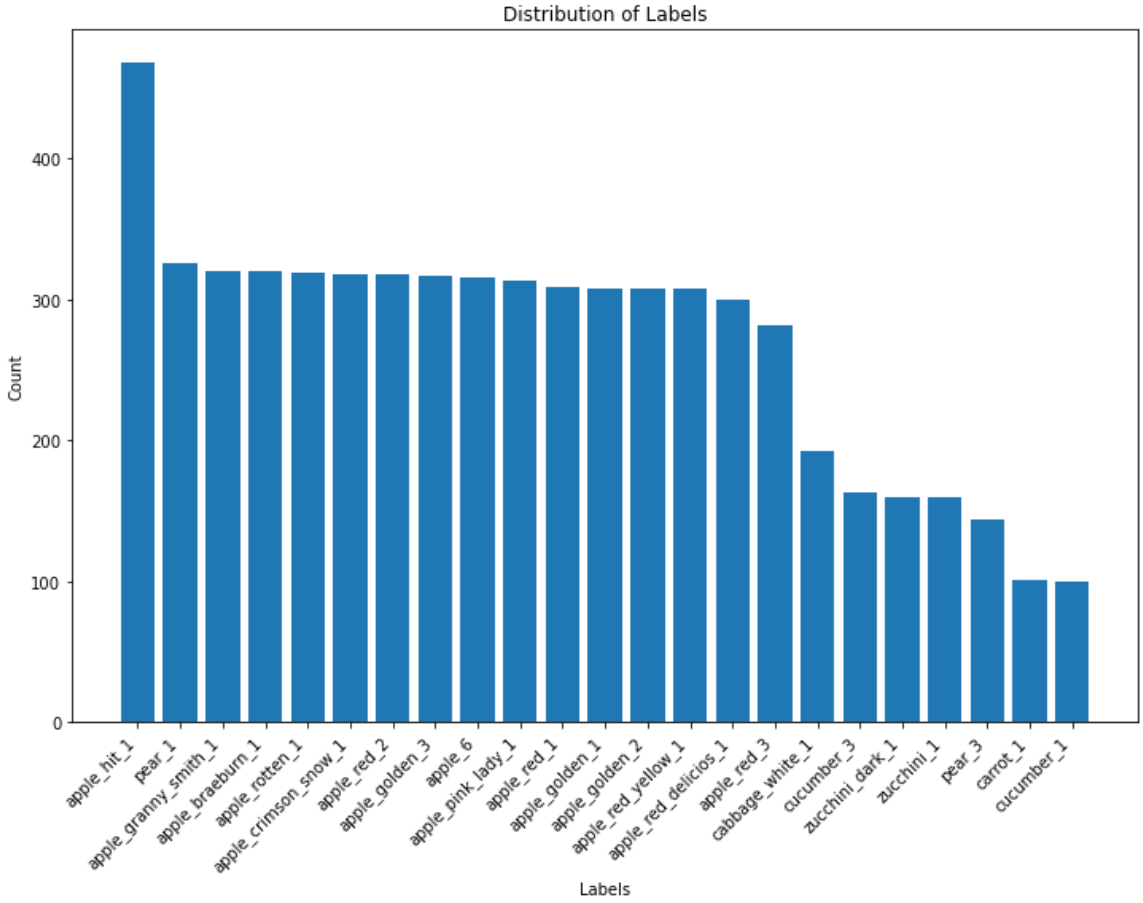
In [8]:

```
# Create a pie chart
plt.figure(figsize=(20, 15))
plt.pie(label_counts, labels=label_counts.index, autopct='%1.1f%%', startangle=14
plt.axis('equal')  # Equal aspect ratio ensures that pie is drawn as a circle.
plt.title('Distribution of Labels')
plt.show()
```



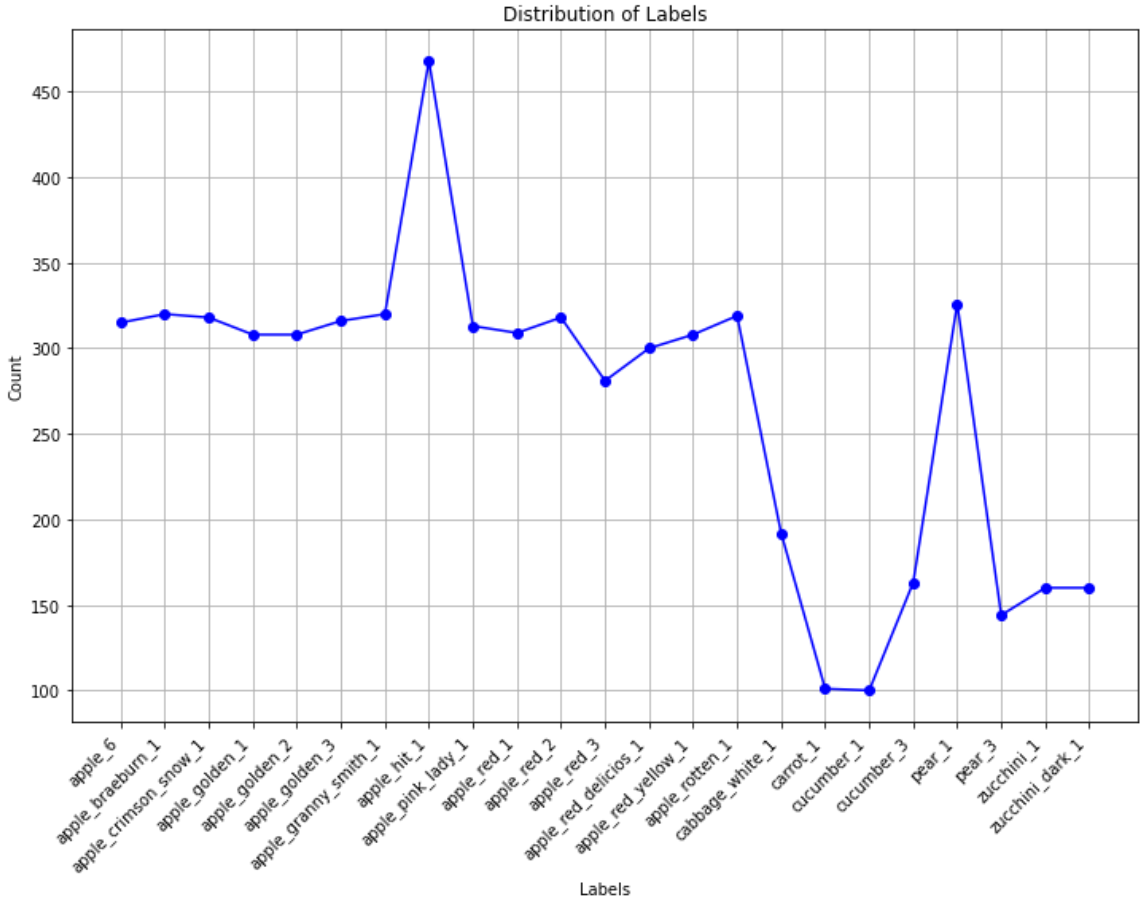
In [9]:

```
# Create a bar chart
plt.figure(figsize=(12, 8))
plt.bar(label_counts.index, label_counts.values)
plt.xlabel('Labels')
plt.ylabel('Count')
plt.title('Distribution of Labels')
plt.xticks(rotation=45, ha='right')
plt.show()
```



In [10]:

```
"""Sort the label counts by index (labels) for a more organized line graph"""
label_counts_sorted = label_counts.sort_index()
"""Create a line graph"""
"""Increase width to 12 inches and height to 8 inches"""
plt.figure(figsize=(12, 8))
plt.plot(label_counts_sorted.index, label_counts_sorted.values, marker='o', lines
plt.xlabel('Labels')
plt.ylabel('Count')
plt.title('Distribution of Labels')
"""Rotate x-axis labels for better visibility"""
plt.xticks(rotation=45, ha='right')
"""Add grid lines for better visualization"""
plt.grid(True)
plt.show()
```



In [11]:

```
df
```

Out[11]:

	pixel_1	pixel_2	pixel_3	pixel_4	pixel_5	pixel_6	pixel_7	pixel_8	pixel_9	pixel_10	.
0	255	255	255	255	255	255	255	255	255	255	.
1	255	255	255	255	255	255	255	255	255	255	.
2	255	255	255	255	255	255	255	255	255	255	.
3	255	255	255	255	255	255	255	255	255	255	.
4	255	255	255	255	255	255	255	255	255	255	.
...	...	...	...	...	...	...	...	...	...	...	.
6162	255	255	255	255	255	255	255	255	255	255	.
6163	255	255	255	255	255	255	255	255	255	255	.
6164	255	255	255	255	255	255	255	255	255	255	.
6165	255	255	255	255	255	255	255	255	255	255	.
6166	255	255	255	255	255	255	255	255	255	255	.

6167 rows × 62501 columns

In [12]:

```
# # Example of a heatmap for correlation between numerical columns
# plt.figure(figsize=(10, 8))
# corr_matrix = df.corr()  # Calculate correlation matrix for numerical columns
# plt.imshow(corr_matrix, cmap='coolwarm', interpolation='nearest')
# plt.colorbar()
# plt.xticks(range(len(corr_matrix)), corr_matrix.columns, rotation=45, ha='right
# plt.yticks(range(len(corr_matrix)), corr_matrix.columns)
# plt.title('Correlation Heatmap')
# plt.show()
```

In [13]:

```
"Normalize the pixel values between 0 and 1"
X=df.iloc[:,:62500]
X=X/255
```

In [14]:

```
X
```

Out[14]:

	pixel_1	pixel_2	pixel_3	pixel_4	pixel_5	pixel_6	pixel_7	pixel_8	pixel_9	pixel_10	.
0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	.
1	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	.
2	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	.
3	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	.
4	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	.
...	...	...	...	...	...	...	...	...	...	...	.
6162	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	.
6163	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	.
6164	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	.
6165	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	.
6166	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	.

6167 rows × 62500 columns

In [15]:

```
Y=df.iloc[:,-1]
Y
```

Out[15]:

```
0          apple_6
1          apple_6
2          apple_6
3          apple_6
4          apple_6
           ...
6162    zucchini_1
6163    zucchini_1
6164    zucchini_1
6165    zucchini_1
6166    zucchini_1
Name: label, Length: 6167, dtype: object
```

In [16]:

```
from sklearn.preprocessing import LabelEncoder
label_encoder = LabelEncoder()
# Fit and transform the labels into numeric values
Y_encoded = label_encoder.fit_transform(Y)
```

In [17]:

```
X
```

Out[17]:

	pixel_1	pixel_2	pixel_3	pixel_4	pixel_5	pixel_6	pixel_7	pixel_8	pixel_9	pixel_10	.
0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	.
1	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	.
2	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	.
3	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	.
4	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	.
...	...	...	...	...	...	...	...	...	...	...	.
6162	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	.
6163	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	.
6164	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	.
6165	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	.
6166	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	.

6167 rows × 62500 columns

In [18]:

```
y_series = pd.Series(Y_encoded, name='Target')

# Concatenate 'X' (features) and 'y_series' (target variable) along columns (axis
df = pd.concat([X, y_series], axis=1)

# Print the merged DataFrame to check the result
df
```

Out[18]:

	pixel_1	pixel_2	pixel_3	pixel_4	pixel_5	pixel_6	pixel_7	pixel_8	pixel_9	pixel_10	.
0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	.
1	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	.
2	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	.
3	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	.
4	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	.
...	...	...	...	...	...	...	...	...	...	...	.
6162	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	.
6163	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	.
6164	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	.
6165	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	.
6166	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	.

6167 rows × 62501 columns

In [19]:

```
from sklearn.model_selection import train_test_split
# Split the data into training and testing sets (80% training, 20% testing)
X_train, X_test, y_train, y_test = train_test_split(X, y_series, test_size=0.2, r
```

In [20]:

```
from sklearn.svm import SVC

# Create an SVM classifier
svm_classifier = SVC(kernel='linear', C=1.0)

# Train the SVM classifier
svm_classifier.fit(X_train, y_train)
```

Out[20]:

```
SVC
SVC(kernel='linear')
```

In [21]:

```
from sklearn.metrics import accuracy_score, classification_report, confusion_matr

# Predict the labels for the test set
y_pred = svm_classifier.predict(X_test)

# Calculate the accuracy of the model
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)

# Print classification report
print(classification_report(y_test, y_pred))

# Print confusion matrix
conf_matrix = confusion_matrix(y_test, y_pred)
print("Confusion Matrix:")
print(conf_matrix)
```

```
Accuracy: 1.0
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        75
           1       1.00      1.00      1.00        75
           2       1.00      1.00      1.00        50
           3       1.00      1.00      1.00        75
           4       1.00      1.00      1.00        64
           5       1.00      1.00      1.00        57
           6       1.00      1.00      1.00        68
           7       1.00      1.00      1.00       101
           8       1.00      1.00      1.00        61
           9       1.00      1.00      1.00        61
          10       1.00      1.00      1.00        53
          11       1.00      1.00      1.00        58
          12       1.00      1.00      1.00        58
          13       1.00      1.00      1.00        56
          14       1.00      1.00      1.00        61
          15       1.00      1.00      1.00        30
          16       1.00      1.00      1.00        21
          17       1.00      1.00      1.00        20
          18       1.00      1.00      1.00        33
          19       1.00      1.00      1.00        65
          20       1.00      1.00      1.00        32
          21       1.00      1.00      1.00        28
          22       1.00      1.00      1.00        32

    accuracy                           1.00      1234
   macro avg       1.00      1.00      1.00      1234
weighted avg       1.00      1.00      1.00      1234

Confusion Matrix:
[[ 75   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0]
 [  0  75   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0]
 [  0   0  50   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0]
 [  0   0   0  75   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0]
 [  0   0   0   0  64   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0]
 [  0   0   0   0   0  57   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0]
 [  0   0   0   0   0   0  68   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0]
 [  0   0   0   0   0   0   0 101   0   0   0   0   0   0   0   0   0
    0   0   0   0   0]
 [  0   0   0   0   0   0   0   0  61   0   0   0   0   0   0   0   0
    0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0  61   0   0   0   0   0   0   0
    0   0
```

```
     0   0   0   0   0]
  [  0   0   0   0   0   0   0   0   0   0  53   0   0   0   0   0
0   0
     0   0   0   0   0]
  [  0   0   0   0   0   0   0   0   0   0   0  58   0   0   0   0
0   0
     0   0   0   0   0]
  [  0   0   0   0   0   0   0   0   0   0   0   0  58   0   0   0
0   0
     0   0   0   0   0]
  [  0   0   0   0   0   0   0   0   0   0   0   0   0  56   0   0
0   0
     0   0   0   0   0]
  [  0   0   0   0   0   0   0   0   0   0   0   0   0   0  61   0
0   0
     0   0   0   0   0]
  [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0  30
0   0
     0   0   0   0   0]
  [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0  2
1   0
     0   0   0   0   0]
  [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0  20
     0   0   0   0   0]
  [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0
    33   0   0   0   0]
  [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0
     0  65   0   0   0]
  [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0
     0   0  32   0   0]
  [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0
     0   0   0  28   0]
  [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0
     0   0   0   0  32]]
```

In [ ]:

In [ ]: