To solve this problem, I use Python to import a python package for image processing (Scikit-image). With the support of this Scikit-image package, the problem becomes very simple. Because this Scikit-image package needs the NumPy array as image object to input, I also import the NumPy package to fix the input.

I implement both 4-connectivity and 8-connectivity, but I separate these two outputs into two different documents. So, I will show the code of 4-connectivity and 8-connectivity separately. Finally, I will explain how the package works in the third part.

1. 4-connectivity

Using the packages mentioned above to implement, we can get the following code

For output data, please kindly find the attached file " output_question_4_4connectivity.txt ".
For code details, please kindly find the attached file " Q4_4_connectivity.py " (Open with Python, Done by Pycharm). For the convenience of review, I have copied all the code of the document as follows.

```
from skimage import measure
# The key packages to solve this question

import numpy as np
# Use to fix the input

import random
# Need to use random() function to randomly create a matrix

L = 20    # The following sentence will be used

A = [[int(random.randint(0, 1)) for i in range(L)] for i in range(L)]
# Create a 20 x 20 matrix and randomly generate 0 and 1 in this matrix

A = np.array(A)
# Because the Scikit-image package needs the NumPy array as image object to input
# So, np.array() function can fix the input, changing {int} A into NumPy array

A = A.astype(bool)
# Because the label() function in the Scikit-image package needs to input.dtype == bool
# So, np.astype() function can fix the input again
# However, in the process of debugging, I found that it's OK not to use this statement.
# The label() function operate normally without A = A.astype(bool)

X = measure.label(A, background=0, return_num=False, connectivity=1)
# Use the label() function in the measure function of the Scikit-image package
# Implement 4-connectivity
```

```
f = open("output_question_4_4connectivity.txt", "w+")    # Control the output

for i in range(L):
        a = ''
        for j in range(L):
                a = a + ' ' + str(int(X[i][j]))
        print(a)
        f.write(a)
        f.write('\n')
        # Write the results into the document

f.close()    # Finish the control
```

2. 8-connectivity

Using the packages mentioned above to implement, we can get the following code

For output data, please kindly find the attached file " output_question_4_8connectivity.txt ".
For code details, please kindly find the attached file " Q4_8_connectivity.py " (Open with Python, Done by Pycharm). For the convenience of review, I have copied all the code of the document as follows.

```
from skimage import measure
# The key packages to solve this question

import numpy as np
# Use to fix the input

import random
# Need to use random() function to randomly create a matrix

L = 20    # The following sentence will be used

A = [[int(random.randint(0, 1)) for i in range(L)] for i in range(L)]
# Create a 20 x 20 matrix and randomly generate 0 and 1 in this matrix

A = np.array(A)
# Because the Scikit-image package needs the NumPy array as image object to input
# So, np.array() function can fix the input, changing {int} A into NumPy array

A = A.astype(bool)
# Because the label() function in the Scikit-image package needs to input.dtype == bool
# So, np.astype() function can fix the input again
# However, in the process of debugging, I found that it's OK not to use this statement.
# The label() function operate normally without A = A.astype(bool)
```

```
Y = measure.label(A, background=0, return_num=False, connectivity=2)
# Use the label() function in the measure function of the Scikit-image package
# Implement 8-connectivity

f = open("output_question_4_8connectivity.txt", "w+")    # Control the output

for i in range(L):
        a = ''
        for j in range(L):
                a = a + str(int(Y[i][j])) + ' '
        print(a)
        f.write(a)
        f.write('\n')
        # Write the results into the document

f.close()    # Finish the control
```

3. How the Scikit-image Package Work

The official explanation of this measure.label() function is as follows

```
Parameters
----------
input : ndarray of dtype int
        Image to label.
background : int, optional
        Consider all pixels with this value as background pixels, and label
        them as 0. By default, 0-valued pixels are considered as background
        pixels.
return_num : bool, optional
        Whether to return the number of assigned labels.
connectivity : int, optional
        Maximum number of orthogonal hops to consider a pixel/voxel
        as a neighbor.
        Accepted values are ranging from    1 to input.ndim. If ``None``, a full
        connectivity of ``input.ndim`` is used.

Returns
-------
labels : ndarray of dtype int
        Labeled array, where all connected regions are assigned the
        same integer value.
num : int, optional
        Number of labels, which equals the maximum label index and is only
        returned if return_num is `True`.
```

```
References
----------
.. [1] Christophe Fiorio and Jens Gustedt, "Two linear time Union-Find
       strategies for image processing", Theoretical Computer Science
       154 (1996), pp. 165-181.
.. [2] Kensheng Wu, Ekow Otoo and Arie Shoshani, "Optimizing connected
       component labeling algorithms", Paper LBNL-56864, 2005,
       Lawrence Berkeley National Laboratory (University of California),
       http://repositories.cdlib.org/lbnl/LBNL-56864

Examples
--------
>>> import numpy as np
>>> x = np.eye(3).astype(int)
>>> print(x)
[[1 0 0]
 [0 1 0]
 [0 0 1]]
>>> print(label(x, connectivity=1))
[[1 0 0]
 [0 2 0]
 [0 0 3]]
>>> print(label(x, connectivity=2))
[[1 0 0]
 [0 1 0]
 [0 0 1]]
>>> print(label(x, background=-1))
[[1 2 2]
 [2 1 2]
 [2 2 1]]
>>> x = np.array([[1, 0, 0],
...               [1, 1, 5],
...               [0, 0, 0]])
>>> print(label(x))
[[1 0 0]
 [1 1 2]
 [0 0 0]]
```

In a nutshell, Connectivity = 1 in the package corresponds to 4-connectivity in the question. Connectivity = 2 in the package corresponds to 8-connectivity in the question. As for the return_num parameter, It outputs how many connected components there are