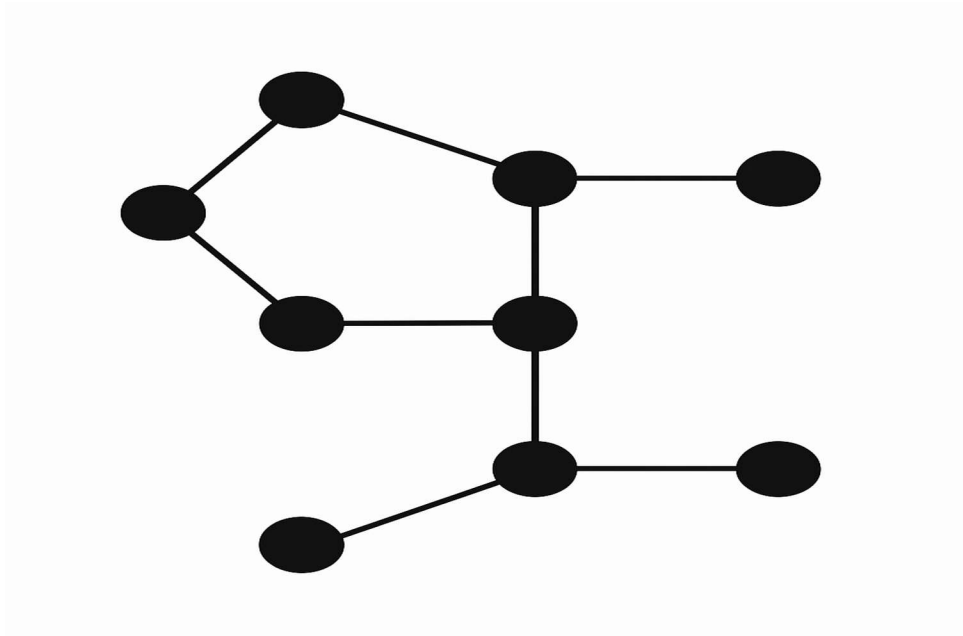


مولفه های متصل (connected components):

مفهوم **connected components** (اجزای متصل) در کتابخانه **OpenCV** ارتباط مستقیم و نزدیکی با نظریه گرافها (**Graph Theory**) دارد. در ادامه، ابتدا به بیان مفهوم اجزای متصل در گراف می پردازیم، سپس آن را به کاربرد در پردازش تصویر و پیاده سازی در **OpenCV** ربط می دهیم.

در نظریه گراف می توان گفت که یک **گراف** مجموعه ای از **رؤوس (نقاط)** و **یال ها (ارتباطات بین نقاط)** است. اگر بتوان از هر رأس به رأس دیگری با پیمایش یال ها رسید (مستقیم یا غیرمستقیم)، آن مجموعه از رؤوس یک **جزء متصل Connected Component** را تشکیل می دهد. به عبارتی یک **جزء متصل** در گراف، یک زیربخش از گراف است که همه ی نقاط درون آن به نوعی با هم ارتباط دارند، اما هیچ ارتباطی با نقاط بیرونی ندارند.



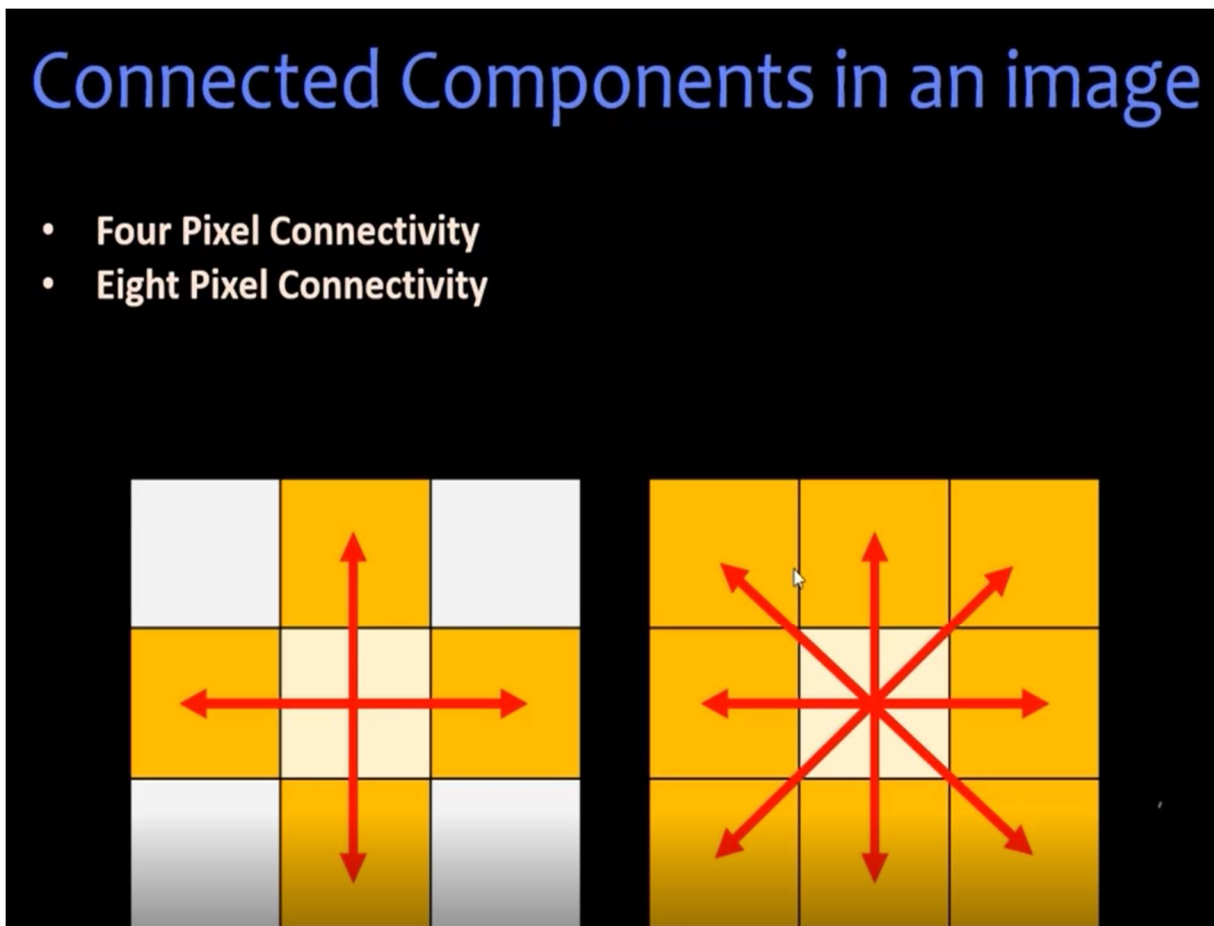
در پردازش تصویر، یک تصویر (مثلاً سیاه و سفید یا دودویی) را می توان به عنوان یک گراف در نظر گرفت که در آن هر پیکسل یک رأس است و یال ها بین پیکسل های مجاور (طبق تعریف همسایگی 4-تایی یا 8-تایی) وجود دارند.

در تعریف همسایگی می توان گفت:

4-connected فقط بالا، پایین، چپ و راست

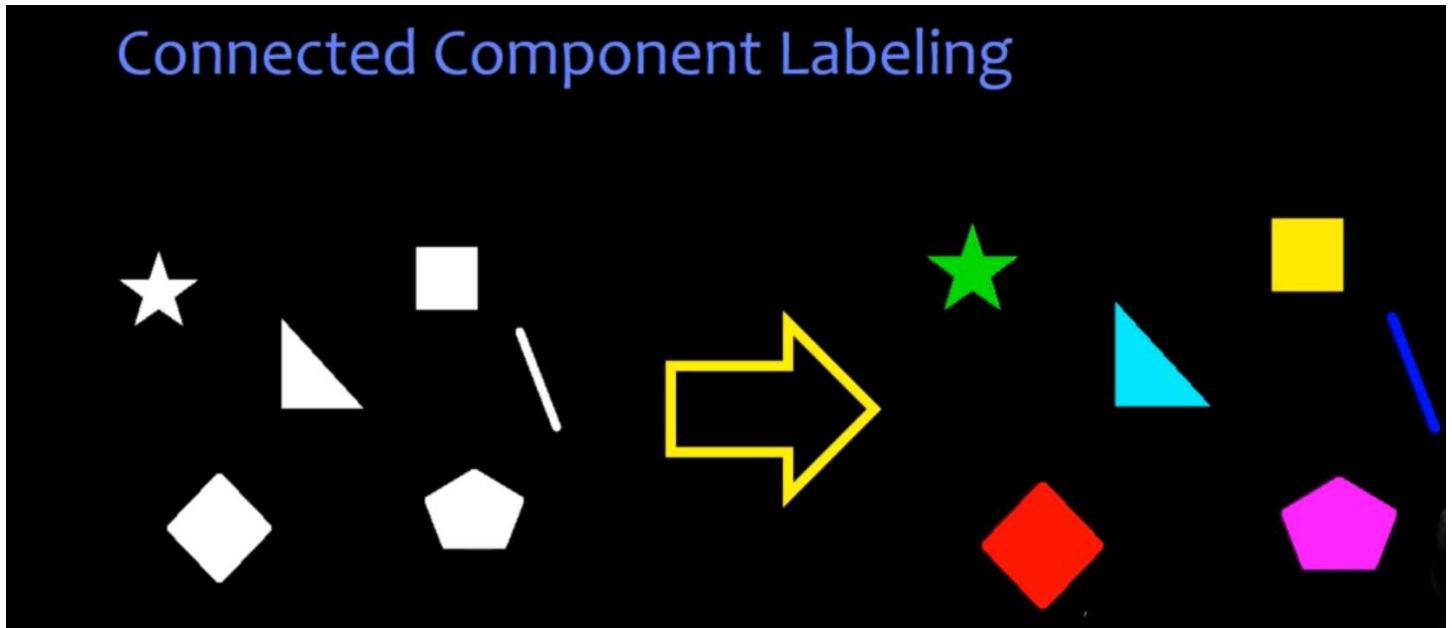
8-connected بالا، پایین، چپ، راست، و چهار گوشه

به عبارتی ما می توانیم به شکل خیلی ساده برای بررسی میزان اتصال همسایه ها به هم شبیه یک گراف از دو مفهوم چهار یا هشت همسایه در کنار هم استفاده کنیم.



بنابراین می توان گفت نواحی پیوسته ای از پیکسل ها که دارای مقدار مشابه (مثلاً همه سفید یا همه سیاه) هستند و از طریق همسایگی مشخص به هم متصل اند، یک جزء متصل را تشکیل می دهند.

بنابراین اگر این عملیات را بر روی اشکال مختلف اعمال کنیم می توانیم انتظار داشته باشیم که برای هر شکل که با استفاده از این مولفه به دست آمده است یک رنگ خاص اختصاص بیابد.



به طور کلی از **connected components** برای انجام عملیات زیر استفاده می شود:

- تشخیص و شمارش اشیاء در تصویر
- حذف نویزهای کوچک (با حذف اجزای کوچکتر از یک آستانه)
- تحلیل ساختاری تصویر
- بخش‌بندی تصویر Image Segmentation

مثال:

با استفاده از کد زیر می توان عملیات مولفه های متصل را انجام داد:

```
import cv2
import matplotlib.pyplot as plt
import numpy as np

# read image as gray
img = cv2.imread('images/shapes.png',0)

# threshold
_, binary = cv2.threshold(img, 127, 255, cv2.THRESH_BINARY)

# run connected
num_labels, labels4 = cv2.connectedComponents(binary,connectivity = 4)
num_labels, labels8 = cv2.connectedComponents(binary,connectivity = 8)

# Normalize labels
normal4 = cv2.normalize(labels4, None,0, 255, cv2.NORM_MINMAX).astype(np.uint8)
normal8 = cv2.normalize(labels8, None,0, 255, cv2.NORM_MINMAX).astype(np.uint8)

# Coloring each component
color4 = cv2.applyColorMap(normal4, cv2.COLORMAP_JET)
color8 = cv2.applyColorMap(normal8, cv2.COLORMAP_JET)

# plot
plt.figure(figsize=[10,8])
plt.subplot(131);plt.imshow(binary, cmap='gray');plt.title('Original')
plt.subplot(132);plt.imshow(color4);plt.title('connected4')
plt.subplot(133);plt.imshow(color8);plt.title('connected8')
```

بیا ببینیم این کد را با هم بررسی کنیم:


در این کد ما هر دو شرط 4 و 8 همسایگی را به عنوان مثال مطرح نموده ایم. اما انتخاب هر یک از این دو مولفه بستگی به عکس مورد نظر دارد.


در مرحله اول عکس را به شکل خاکستری خوانده ایم و عملیات threshold را روی آن اعمال کرده ایم تا مطمئن شویم عکس باینری است چرا که عملیات مولفه های متصل فقط بر روی عکس های باینری اجرا میشود.

در مرحله بعد از فانکشن مربوط به عملیات مولفه های متصل استفاده نموده ایم که در آن یکبار از 4 همسایگی و یکبار از 8 همسایگی استفاده نموده ایم:

```
num_labels, labels4 = cv2.connectedComponents(binary, connectivity = 4)
num_labels, labels8 = cv2.connectedComponents(binary, connectivity = 8)
```


در این خط:

connectedComponents  به دنبال اجزای سفید متصل در تصویر باینری می گردد.

 دو نوع اتصال بررسی می شود:

• **connectivity=4** → فقط بالا، پایین، چپ، راست همسایه هستند.

• **connectivity=8** → همسایگان قطری هم حساب می شوند.

 خروجی:

- num_labels: تعداد اجزای متصل پیدا شده + 1 (برچسب 0 برای زمینه) به عبارتی اگر در تصویر 3 ناحیه مجزا داشته باشیم مقدار این متغیر برابر با 4 است که شامل 3 جسم و یک پس زمینه خواهد بود.
 - labels4, labels8: تصاویری با برچسب عددی برای هر جزء جداگانه به عبارتی مقدار داخل این متغیر یک آرایه است با ابعادی برابر با ابعاد تصویر ورودی که در آن هر پیکسل با توجه به اینکه متعلق به کدام جز متصل است یک عدد صحیح یا لیبل دریافت می کند:
- در این آرایه پیکسل هایی که به پس زمینه تعلق دارند 0 و سایر اجزا مقدار 1 را دارند.

```
• normal4 = cv2.normalize(labels4, None, 0, 255, cv2.NORM_MINMAX).astype(np.uint8)
• normal8 = cv2.normalize(labels8, None, 0, 255, cv2.NORM_MINMAX).astype(np.uint8)
```

در مرحله بعد با هدف نمایش عکس جدید و ویژوال کردن آن از این خط کد استفاده می شود که توسط آن ماتریس های lable4 و lable8 به بازه 0 تا 255 و به صورت یک عکس نرمال می شوند. بنابراین این تابع از OpenCV مقدارهای موجود در ماتریس labels4 را از بازه ی اصلی شان (مثلاً بین 0 تا 3 یا هر مقدار دیگر) به بازه ی جدید بین 0 و 255 نگاشت (Normalize) می کند.

labels4 ماتریسی از برچسب های اجزای متصل که از تابع connectedComponents آمده.

None به OpenCV می گوید که خروجی را در یک ماتریس جدید برگرداند (و نه روی ماتریس موجود overwrite کند).

255, 0 بازه‌ی مقصد برای نرمال‌سازی.

cv2.NORM_MINMAX روش نرمال‌سازی → مقدار کمینه را به 0 و بیشینه را به 255 نگاشت می‌کند، و سایر مقادیر را به صورت خطی بین آن دو نگه می‌دارد.

```
color4 = cv2.applyColorMap(normal4, cv2.COLORMAP_JET)
color8 = cv2.applyColorMap(normal8, cv2.COLORMAP_JET)
```

در نهایت با استفاده از این کد یک colormap بر روی تصویر خاکستری اجاد کرده و آن را رنگی می‌کنیم.

- normal4 تصویری 8 بیتی در بازه‌ی 0 تا 255 (که در مرحله‌ی قبل با cv2.normalize از labels4 به دست آمده).
- cv2.COLORMAP_JET یک colormap از نوع "JET" که به هر مقدار خاکستری، یک رنگ از طیف آبی → سبز → زرد → قرمز اختصاص می‌دهد.

خروجی:

- color4: یک تصویر رنگی (BGR) که نمای بصری جذاب‌تری از اجزای متصل را ارائه می‌دهد. هر برچسب رنگ متفاوتی خواهد داشت (بسته به مقدار نرمال شده آن).