

رسالة محمد

# مبانی بینایی کامپیوتر

مدرس: محمدرضا محمدی

بهار ۱۴۰۳

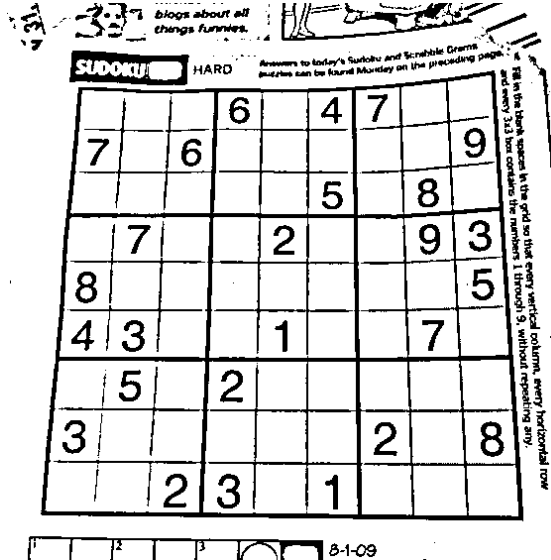
# ناحيه بندی تصوير

Image Segmentation

# آستانه گذاری افقی

```
dst = cv2.adaptiveThreshold(src, maxValue, adaptiveMethod, thresholdType, blockSize, C)
```

// src:	Source 8-bit single-channel image
// maxValue:	Non-zero value assigned to the pixels for which the condition is satisfied
// adaptiveMethod:	Adaptive thresholding algorithm to use (MEAN or GAUSSIAN)
// thresholdType:	Thresholding type that must be either THRESH_BINARY or THRESH_BINARY_INV
// blockSize:	Size of a pixel neighborhood that is used to calculate a threshold value
// C:	Constant subtracted from the mean or weighted mean
// dst:	Destination image of the same size and the same type as src



# استخراج ناحیه‌ها از تصویر باینری

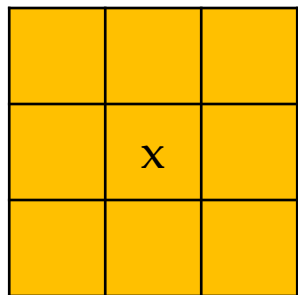
- با استفاده از روش‌های ذکر شده، یک تصویر دوسطحی بدست می‌آید
- حال باید پیکسل‌های مربوط به هر ناحیه را مشخص کنیم

0	0	0	1	1	0	0	0
0	0	1	1	1	1	0	0
0	0	0	1	0	1	0	0
0	1	0	0	0	0	0	1
1	1	0	0	0	1	1	1
1	1	0	0	1	1	0	0
1	0	0	0	0	0	1	1
0	0	0	0	0	0	1	0

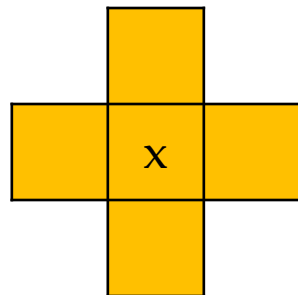
0	0	0	1	1	0	0	0
0	0	1	1	1	1	0	0
0	0	0	1	0	1	0	0
0	2	0	0	0	0	0	3
2	2	0	0	0	3	3	3
2	2	0	0	3	3	0	0
2	0	0	0	0	0	3	3
0	0	0	0	0	0	3	0

# اتصال پیکسل‌ها

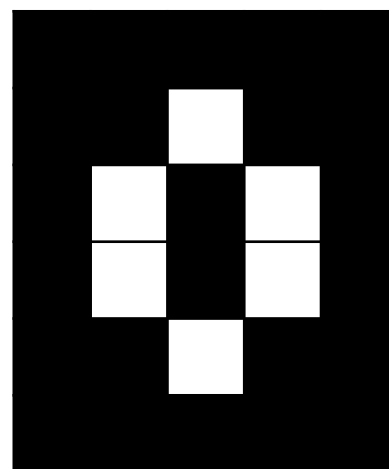
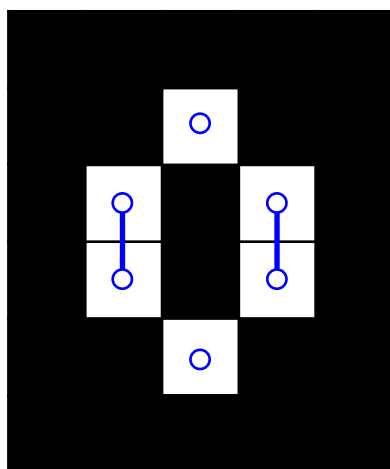
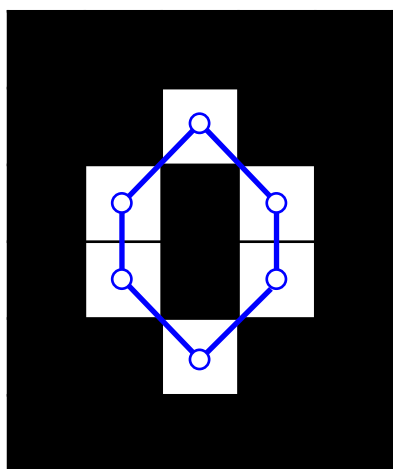
- کدام پیکسل‌ها به هم متصل هستند؟



8-connectivity

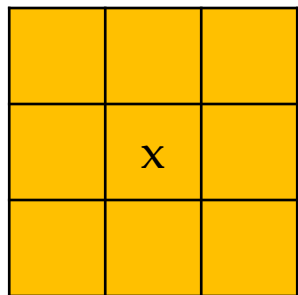


4-connectivity

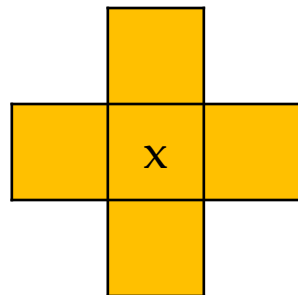


# اتصال پیکسل‌ها

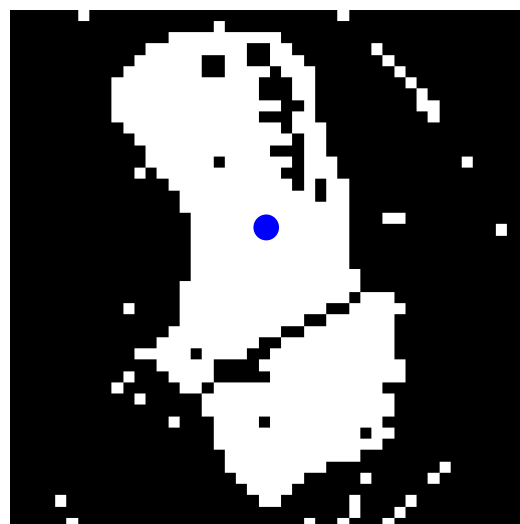
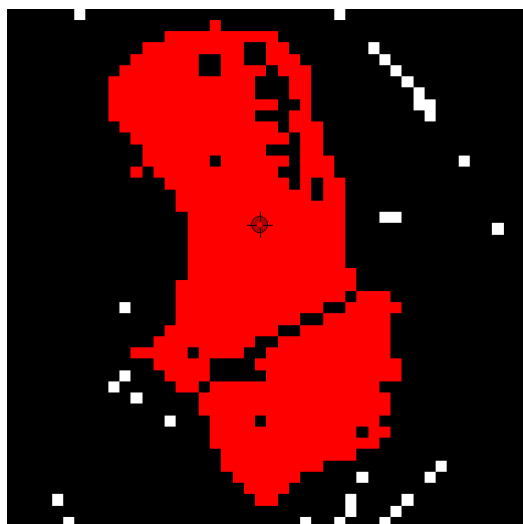
- کدام پیکسل‌ها به هم متصل هستند؟



8-connectivity

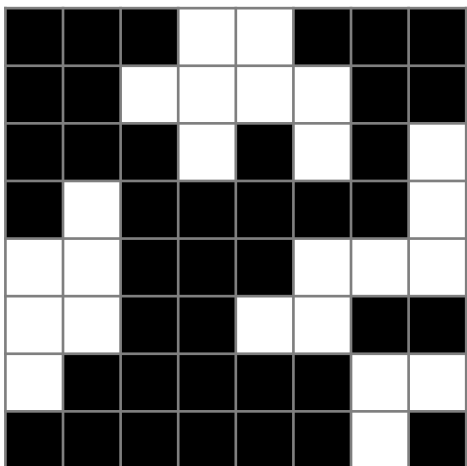


4-connectivity



# برچسب‌گذاری اجزاء متصل

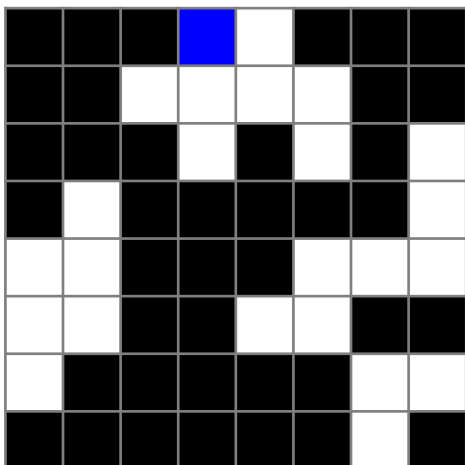
- چگونه اجزاء متصل در تصویر باینری را مشخص کنیم؟
- الگوریتم یک جزء در هر زمان
  - در این الگوریتم برای استخراج اجزاء، از پیکسل نخست تصویر شروع می‌کنیم و به هر پیکسل که رسیدیم پیکسل‌های متصل به آن را محاسبه می‌کنیم





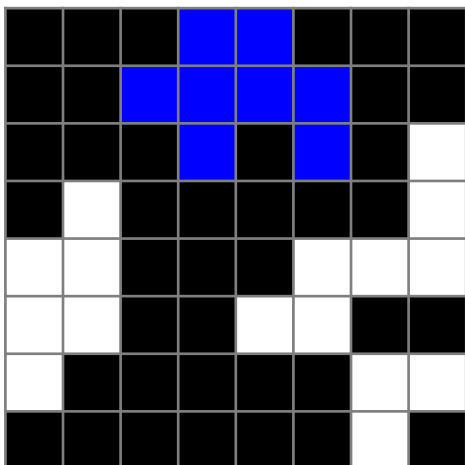
# برچسب گذاری اجزاء متصل

- چگونه اجزاء متصل در تصویر باینری را مشخص کنیم؟
- الگوریتم یک جزء در هر زمان
  - در این الگوریتم برای استخراج اجزاء، از پیکسل نخست تصویر شروع می کنیم و به هر پیکسل که رسیدیم پیکسل های متصل به آن را محاسبه می کنیم



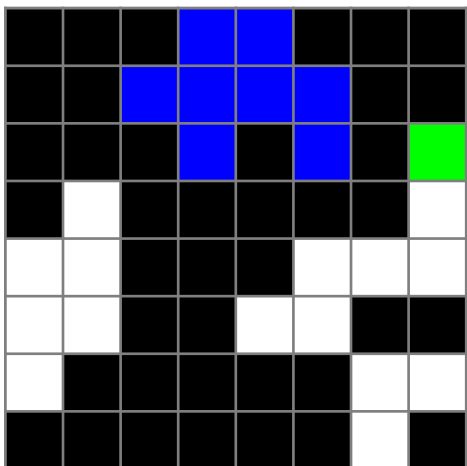
# برچسب گذاری اجزاء متصل

- چگونه اجزاء متصل در تصویر باینری را مشخص کنیم؟
- الگوریتم یک جزء در هر زمان
  - در این الگوریتم برای استخراج اجزاء، از پیکسل نخست تصویر شروع می کنیم و به هر پیکسل که رسیدیم پیکسل های متصل به آن را محاسبه می کنیم



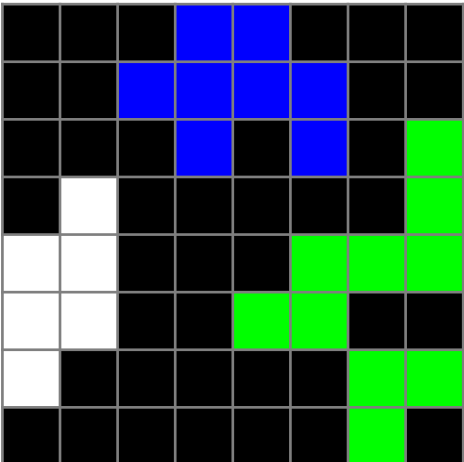
# برچسب گذاری اجزاء متصل

- چگونه اجزاء متصل در تصویر باینری را مشخص کنیم؟
- الگوریتم یک جزء در هر زمان
  - در این الگوریتم برای استخراج اجزاء، از پیکسل نخست تصویر شروع می کنیم و به هر پیکسل که رسیدیم پیکسل های متصل به آن را محاسبه می کنیم
  - سپس، این روند برای پیکسل های بعدی که هنوز برچسب نخورده اند تکرار می شود



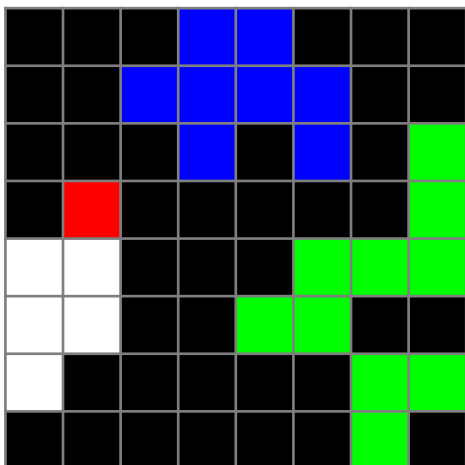
# برچسب‌گذاری اجزاء متصل

- چگونه اجزاء متصل در تصویر باینری را مشخص کنیم؟
- الگوریتم یک جزء در هر زمان
  - در این الگوریتم برای استخراج اجزاء، از پیکسل نخست تصویر شروع می‌کنیم و به هر پیکسل که رسیدیم پیکسل‌های متصل به آن را محاسبه می‌کنیم
  - سپس، این روند برای پیکسل‌های بعدی که هنوز برچسب نخورده‌اند تکرار می‌شود



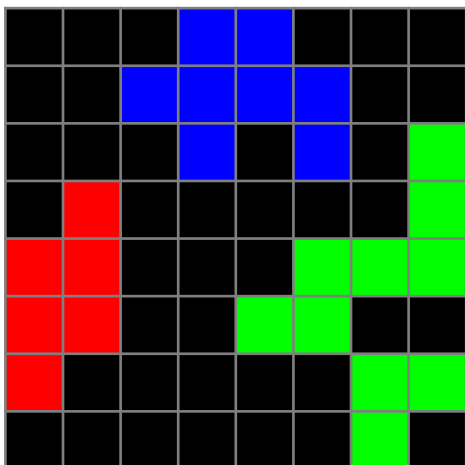
# برچسب گذاری اجزاء متصل

- چگونه اجزاء متصل در تصویر باینری را مشخص کنیم؟
- الگوریتم یک جزء در هر زمان
  - در این الگوریتم برای استخراج اجزاء، از پیکسل نخست تصویر شروع می کنیم و به هر پیکسل که رسیدیم پیکسل های متصل به آن را محاسبه می کنیم
  - سپس، این روند برای پیکسل های بعدی که هنوز برچسب نخورده اند تکرار می شود



# برچسب گذاری اجزاء متصل

- چگونه اجزاء متصل در تصویر باینری را مشخص کنیم؟
- الگوریتم یک جزء در هر زمان
  - در این الگوریتم برای استخراج اجزاء، از پیکسل نخست تصویر شروع می کنیم و به هر پیکسل که رسیدیم پیکسل های متصل به آن را محاسبه می کنیم
  - سپس، این روند برای پیکسل های بعدی که هنوز برچسب نخورده اند تکرار می شود



# استخراج یک ناحیه متصل

```
// Finding the connected component containing an  
object pixel p
```

## 1. Initialize

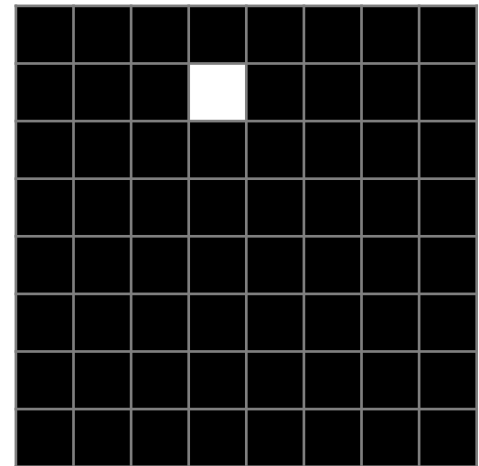
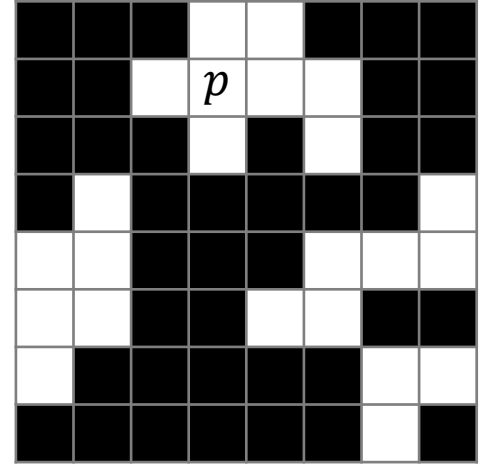
1. Create a result set  $S$  that contains only  $p$
2. Create a Visited flag at each pixel, and set it to be False except for  $p$
3. Initialize a queue (or stack)  $Q$  that contains only  $p$ .

## 2. Repeat until $Q$ is empty:

1. Pop a pixel  $x$  from  $Q$ .

$$S = \{(1,3)\}$$

$$Q = \{(1,3)\}$$



# استخراج یک ناحیه متصل

```
// Finding the connected component containing an  
object pixel p
```

## 1. Initialize

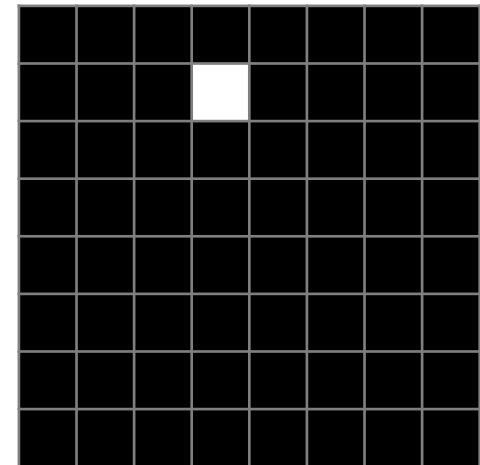
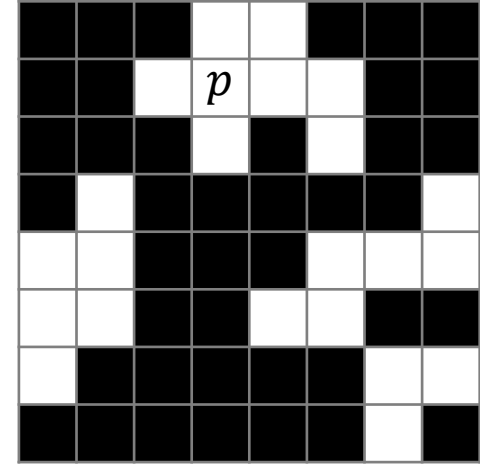
1. Create a result set  $S$  that contains only  $p$
2. Create a Visited flag at each pixel, and set it to be False except for  $p$
3. Initialize a queue (or stack)  $Q$  that contains only  $p$ .

## 2. Repeat until $Q$ is empty:

1. Pop a pixel  $x$  from  $Q$ .
2. For each unvisited object pixel  $y$  connected to  $x$ , add  $y$  to  $S$ , set its flag to be visited, and push  $y$  to  $Q$ .

$$S = \{(1,3)\}$$

$$Q = \{ \}$$





# استخراج یک ناحیه متصل

```
// Finding the connected component containing an  
object pixel p
```

## 1. Initialize

1. Create a result set  $S$  that contains only  $p$
2. Create a Visited flag at each pixel, and set it to be False except for  $p$
3. Initialize a queue (or stack)  $Q$  that contains only  $p$ .

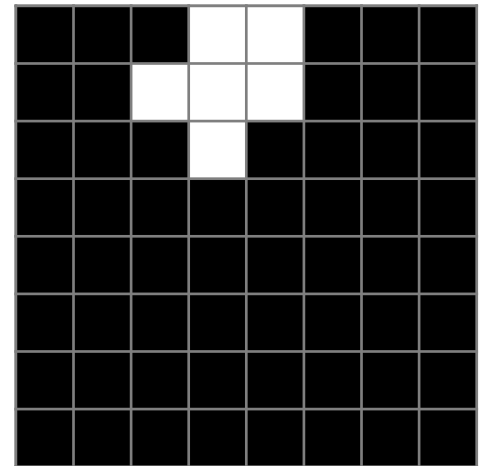
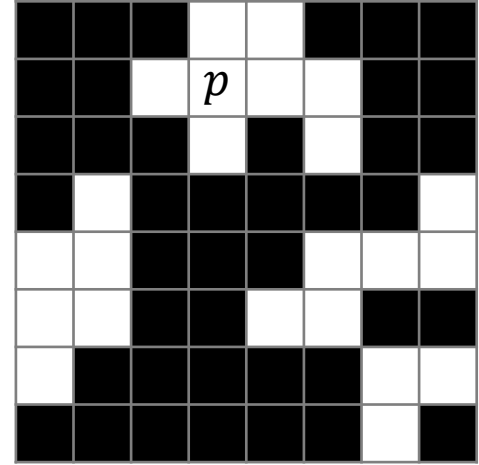
## 2. Repeat until $Q$ is empty:

1. Pop a pixel  $x$  from  $Q$ .
2. For each unvisited object pixel  $y$  connected to  $x$ , add  $y$  to  $S$ , set its flag to be visited, and push  $y$  to  $Q$ .

## 3. Output $S$

$$S = \{(1,3), (0,3), (1,2), \dots\}$$

$$Q = \{(0,3), (1,2), \dots\}$$



# استخراج یک ناحیه متصل

```
// Finding the connected component containing an  
object pixel p
```

## 1. Initialize

1. Create a result set  $S$  that contains only  $p$
2. Create a Visited flag at each pixel, and set it to be False except for  $p$
3. Initialize a queue (or stack)  $Q$  that contains only  $p$ .

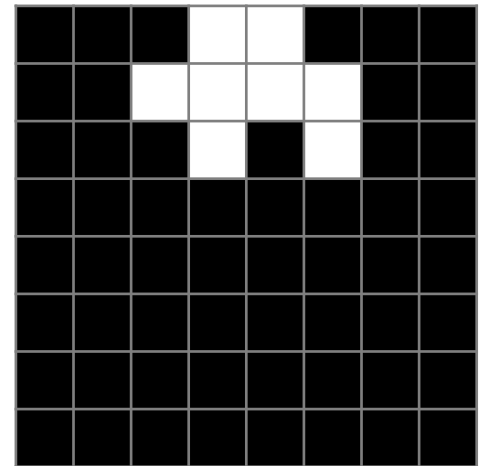
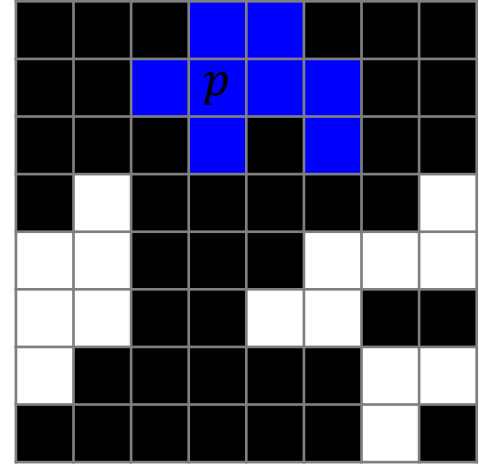
## 2. Repeat until $Q$ is empty:

1. Pop a pixel  $x$  from  $Q$ .
2. For each unvisited object pixel  $y$  connected to  $x$ , add  $y$  to  $S$ , set its flag to be visited, and push  $y$  to  $Q$ .

## 3. Output $S$

$$S = \{(1,3), (0,3), (1,2), \dots\}$$

$$Q = \{ \}$$



# استخراج یک ناحیه متصل

```
// Finding the connected component containing an  
object pixel p
```

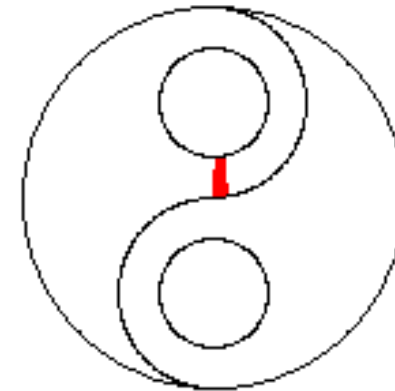
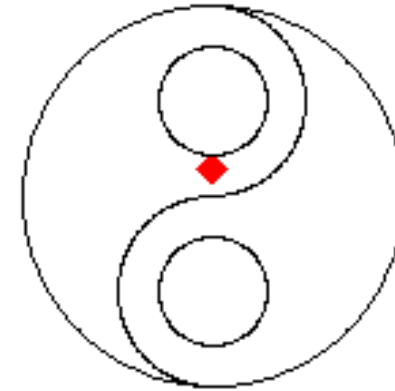
## 1. Initialize

1. Create a result set  $S$  that contains only  $p$
2. Create a Visited flag at each pixel, and set it to be False except for  $p$
3. Initialize a queue (or stack)  $Q$  that contains only  $p$ .

## 2. Repeat until $Q$ is empty:

1. Pop a pixel  $x$  from  $Q$ .
2. For each unvisited object pixel  $y$  connected to  $x$ , add  $y$  to  $S$ , set its flag to be visited, and push  $y$  to  $Q$ .

## 3. Output $S$



# رشد ناحیه

- هدف از این الگوریتم استخراج ناحیه مربوط به یک شیء در تصویر است که یک نقطه از آن را می دانیم
- به نقطه اولیه بذر یا seed گفته می شود



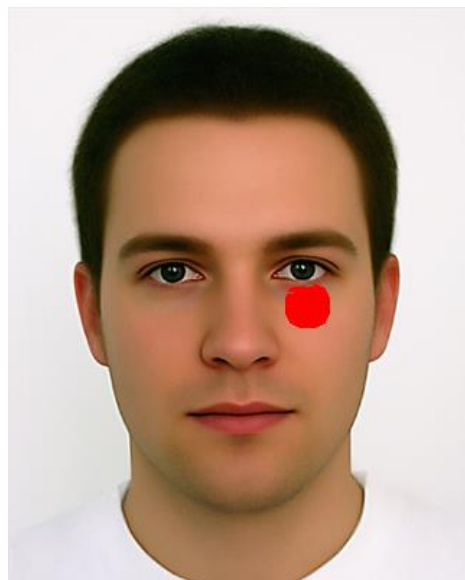
# رشد ناحیه

- هدف از این الگوریتم استخراج ناحیه مربوط به یک شیء در تصویر است که یک نقطه از آن را می دانیم
- به نقطه اولیه بذر یا seed گفته می شود



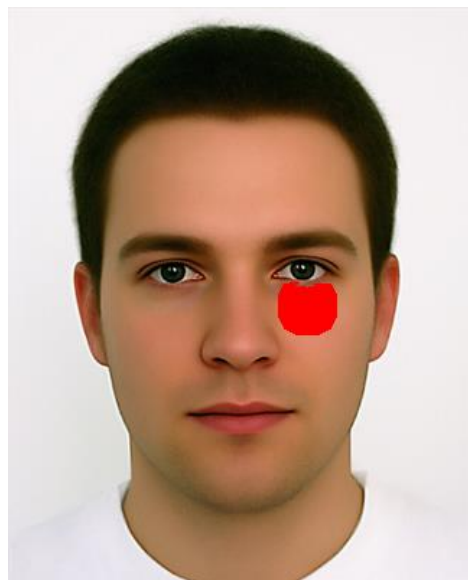
# رشد ناحیه

- هدف از این الگوریتم استخراج ناحیه مربوط به یک شیء در تصویر است که یک نقطه از آن را می دانیم
- به نقطه اولیه بذر یا seed گفته می شود



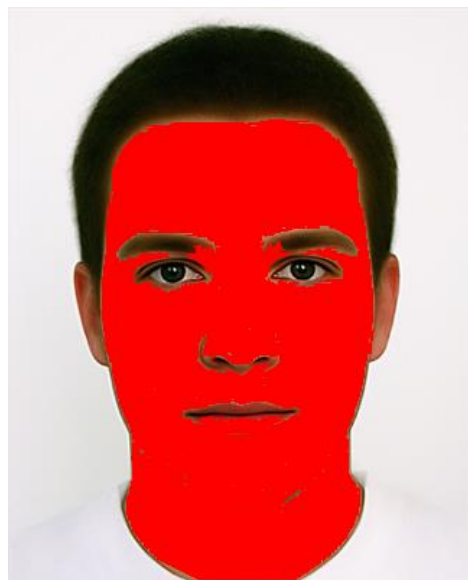
# رشد ناحیه

- هدف از این الگوریتم استخراج ناحیه مربوط به یک شیء در تصویر است که یک نقطه از آن را می دانیم
- به نقطه اولیه بذر یا seed گفته می شود



# رشد ناحیه

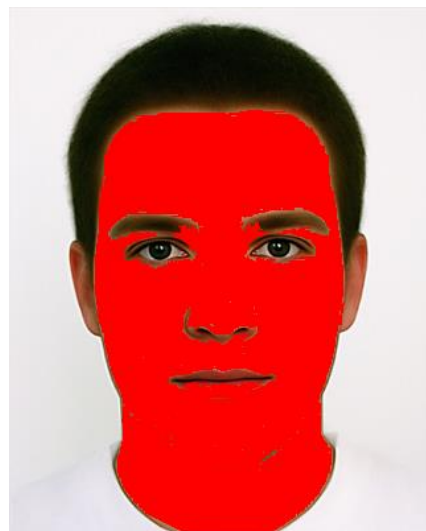
- هدف از این الگوریتم استخراج ناحیه مربوط به یک شیء در تصویر است که یک نقطه از آن را می دانیم
- به نقطه اولیه بذر یا seed گفته می شود





# رشد ناحیه

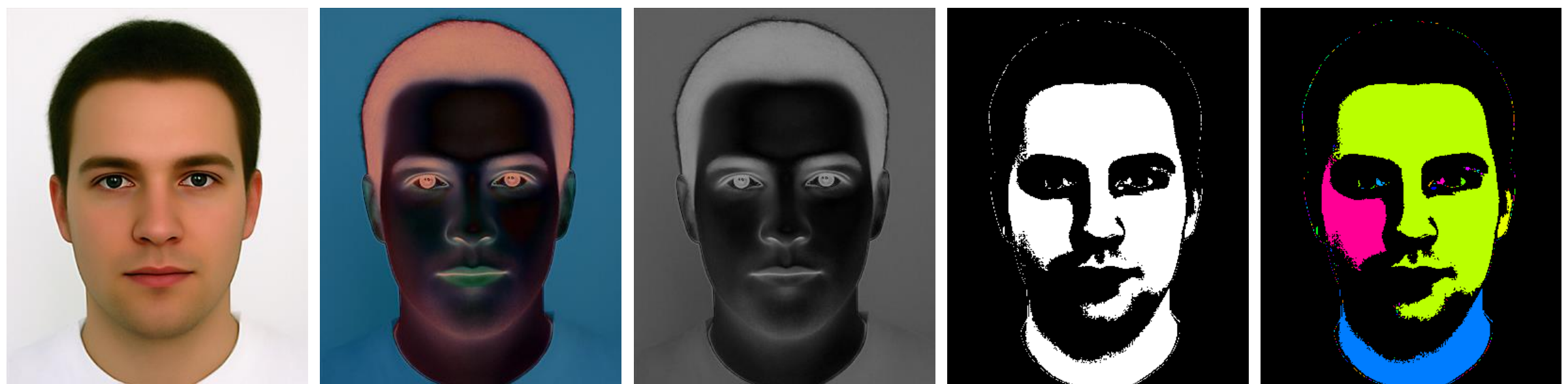
- الگوریتم رشد ناحیه مشابه با استخراج یک جزء متصل در تصویر باینری است
- تفاوت با تصویر باینری آن است که مقادیر پیکسل‌ها باینری نیستند و حتی می‌توانند رنگی باشند
- در پیاده‌سازی، تفاوت اصلی در این است که پیکسل‌های همسایه به چه شرطی به ناحیه اضافه شوند؟
- باید محتوای مشابهی داشته باشد که معادل با اختلاف کم است
- اختلاف با چه معیاری سنجیده شود؟



# معیار اختلاف برای رشد ناحیه

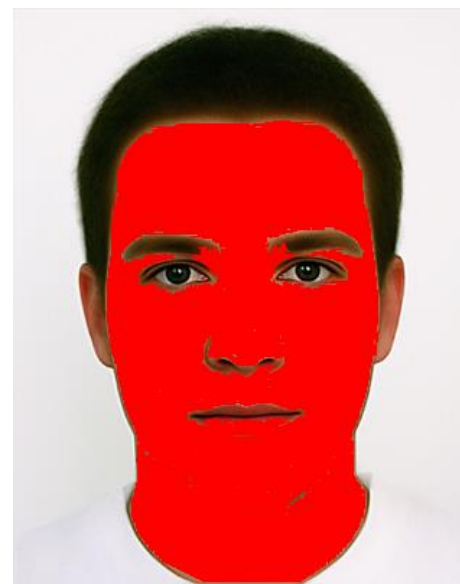
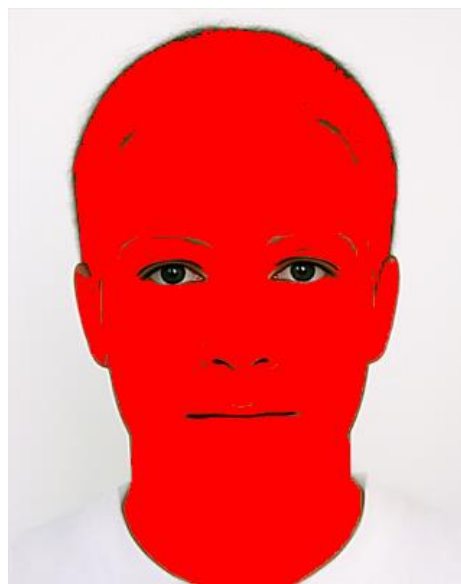
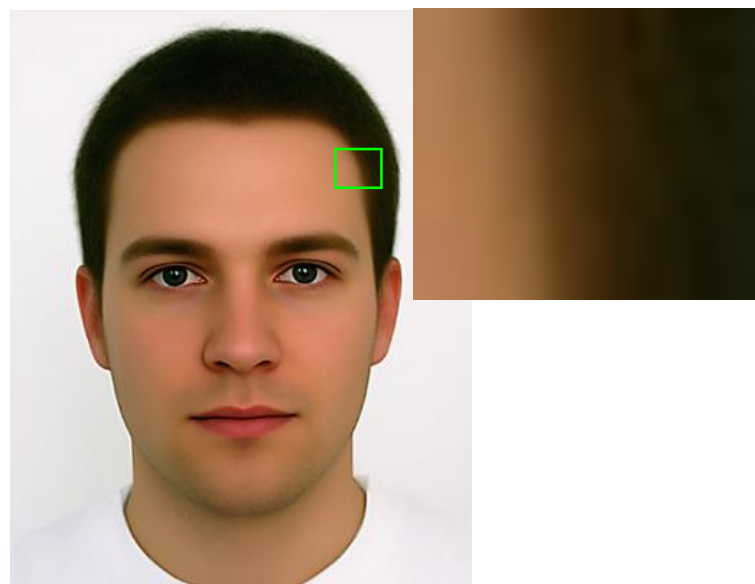
- می‌توان رنگ پیکسل مورد نظر را با رنگ پیکسل بذر مقایسه کرد و اگر اختلاف آنها از حدی کمتر بود به ناحیه اضافه شوند

- این روش معادل با این است که ابتدا تصویر را بر اساس اختلاف با رنگ مورد نظر باینری کرده و سپس ناحیه متصل به این پیکسل را استخراج کنیم



# معیار اختلاف برای رشد ناحیه

- می‌توان مقایسه را بجای پیکسل بذر با پیکسل‌های مجاور انجام داد
  - به این حالت رشد محلی (در برابر رشد سراسری) گفته می‌شود
  - این روش برای حالت‌هایی که مرز ضعیف وجود دارد دچار نشت می‌شود

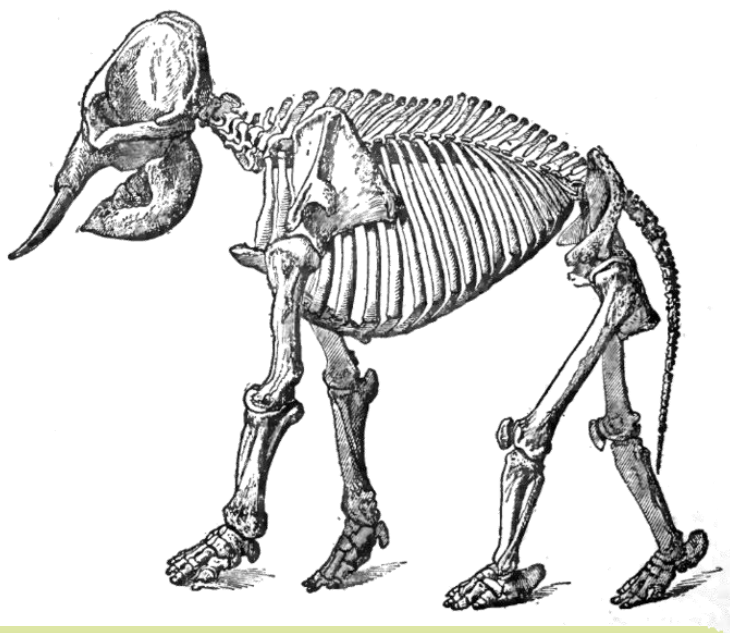


# پردازش‌های مورفولوژی

## Morphological Image Processing

# مورفولوژی

- مورفولوژی (ریخت‌شناسی) شاخه‌ای از علم زیست‌شناسی است که به مطالعه شکل ظاهری و ویژگی‌های ساختاری خاص حیوانات و گیاهان می‌پردازد
- پردازش‌های مورفولوژی به ابزار و روش‌هایی گفته می‌شود که برای استخراج اجزای مفید تصویر نظیر مرزها و گوشه‌ها استفاده می‌شود
- عملگرهای مورفولوژی اغلب برای تصاویر باینری استفاده می‌شوند

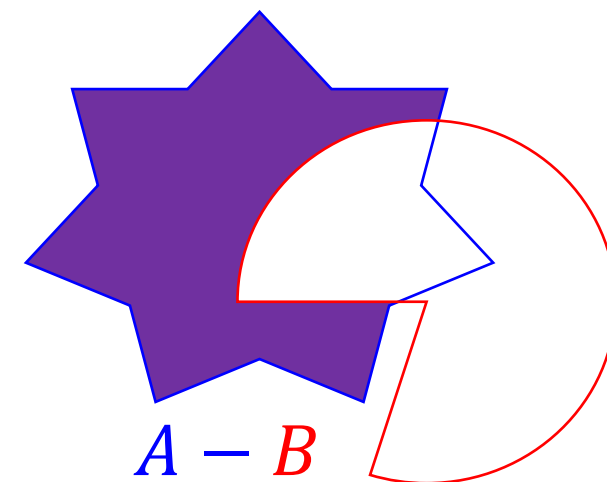
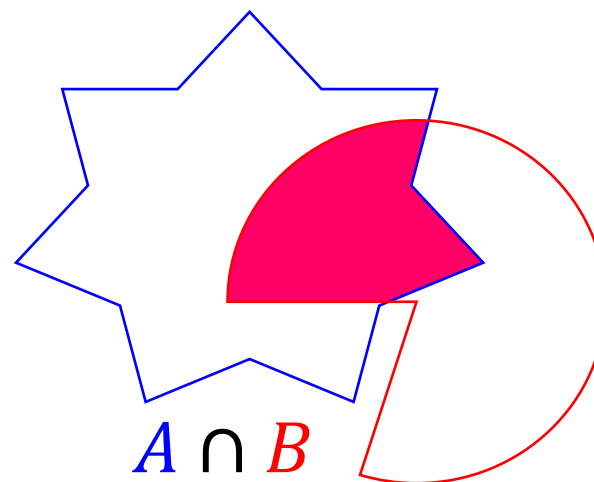
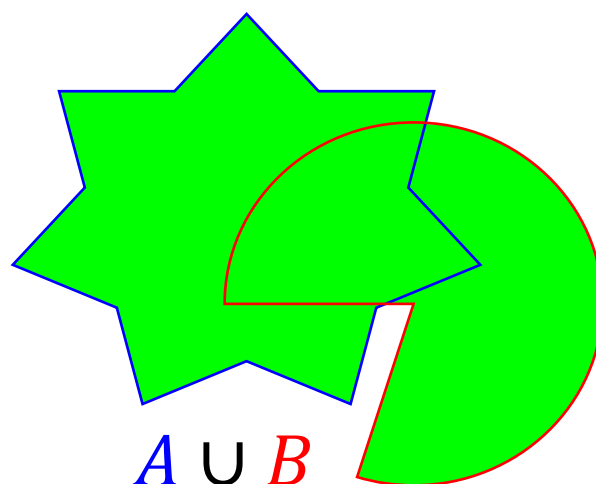
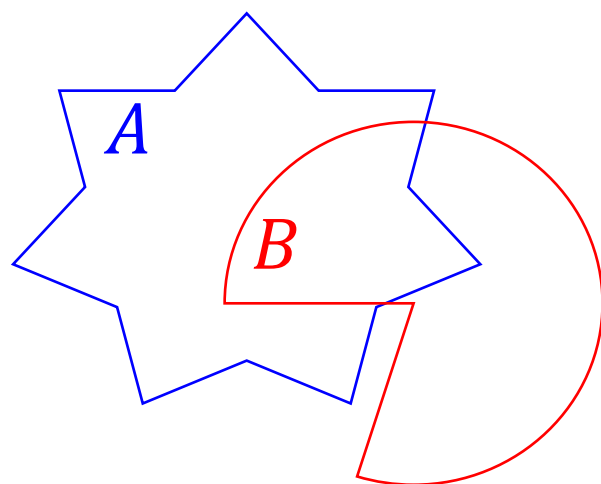


# نظریه مجموعه‌ها

- اگر  $A$  یک مجموعه در  $Z^2$  و  $a = (a_1, a_2)$  یک عنصر از این مجموعه باشد، از نماد  $a \in A$  استفاده می‌کنیم
- و اگر  $a$  یک عنصر از  $A$  نباشد، نماد  $a \notin A$  را استفاده می‌کنیم
- مجموعه بدون عضو، مجموعه تهی نامیده می‌شود با نماد  $\emptyset$
- اگر تمام عناصر مجموعه  $A$  در مجموعه  $B$  وجود داشته باشند، در آن صورت  $A$  زیرمجموعه  $B$  است و با نماد  $A \subseteq B$  نشان داده می‌شود

# نظریه مجموعه‌ها

- اجتماع مجموعه‌های  $A$  و  $B$  شامل تمام عناصر این دو مجموعه است
- اشتراک مجموعه‌های  $A$  و  $B$  تنها شامل عناصر مشترک در دو مجموعه است
- تفاضل مجموعه  $A$  از مجموعه  $B$  شامل عناصری از  $A$  است که در  $B$  وجود ندارند  $A - B = A \cap B^c$
- مکمل مجموعه  $A$  شامل تمام عناصری است که در مجموعه  $A$  وجود ندارند و با  $A^c$  نشان داده می‌شود



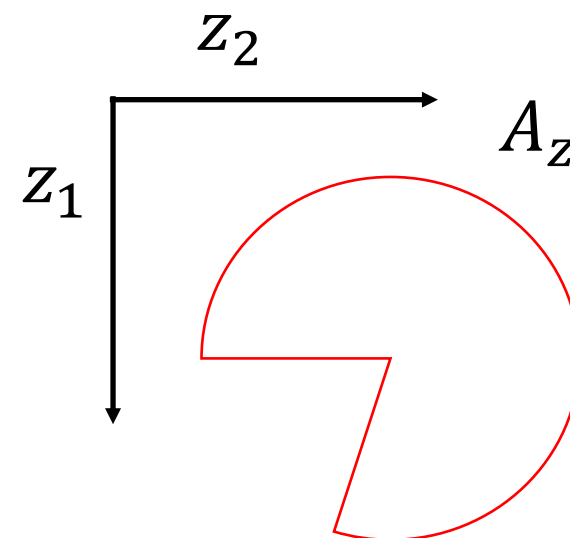
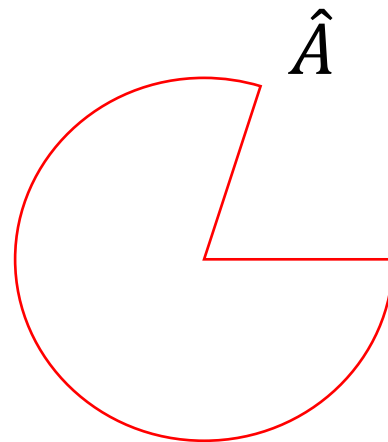
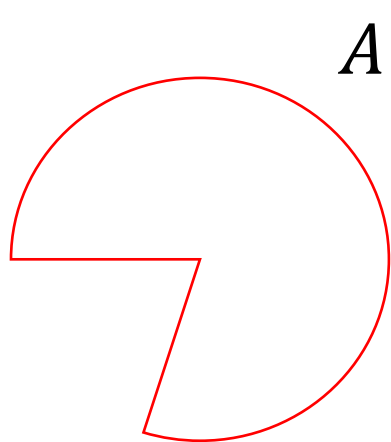
## نظریه مجموعه‌ها

- انعکاس مجموعه  $A$  به صورت زیر تعریف می‌شود

$$\hat{A} = \{w | w = -a, \text{ for } a \in A\}$$

- انتقال مجموعه  $A$  به اندازه نقطه  $z = (z_1, z_2)$  عبارت است از

$$A_z = \{w | w = a + z, \text{ for } a \in A\}$$





# عملگر گسترش

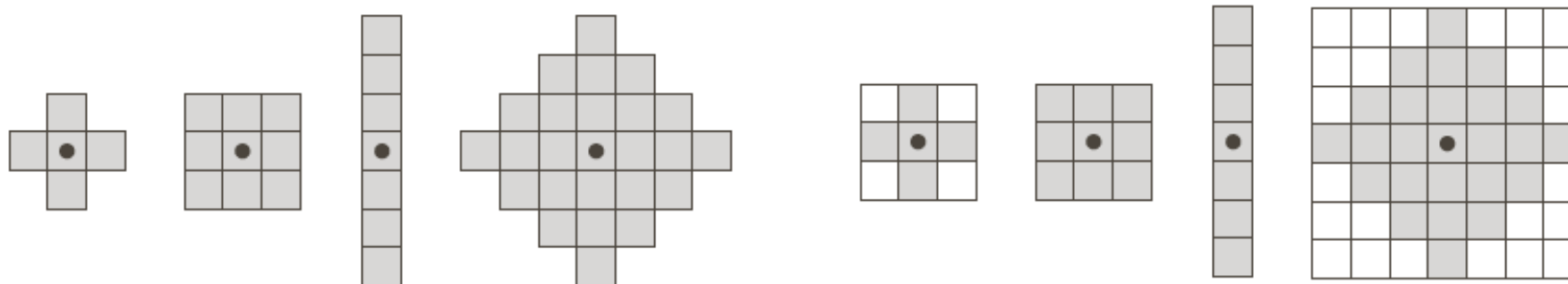
- عملگر گسترش (dilate) برای گسترش مجموعه  $A$  توسط  $B$  به صورت زیر تعریف می‌شود:

$$A \oplus B = \{z \mid (\hat{B})_z \cap A \neq \emptyset\}$$

- این رابطه به مفهوم بدست آوردن انعکاس  $B$  حول مرکز (لنگر) خودش و جابجایی آن به اندازه  $z$  است که اگر این نسخه از  $B$  دارای اشتراک با  $A$  بود،  $z$  جزء مجموعه جدید خواهد بود

# عنصر ساختاری

- به مجموعه  $B$  در عملگر گسترش (و عملگرهای بعدی) عنصر ساختاری (Structuring Element) گفته می‌شود که انتخاب مناسب آن نتیجه مستقیم در عملکرد عملگرها دارد



## مثال: گسترش 1D

$$A \oplus B = \{z \mid (\hat{B})_z \cap A \neq \emptyset\}$$

Input image

0	1	0	0	0	0	1	0	0	1	1	0
---	---	---	---	---	---	---	---	---	---	---	---



Structuring Element

1	1	1
---	---	---



Output Image

1										
---	--	--	--	--	--	--	--	--	--	--

## مثال: گسترش 1D

$$A \oplus B = \{z \mid (\hat{B})_z \cap A \neq \emptyset\}$$

Input image

0	1	0	0	0	0	1	0	0	1	1	0
---	---	---	---	---	---	---	---	---	---	---	---



Structuring Element

1	1	1
---	---	---



Output Image

1	1								
---	---	--	--	--	--	--	--	--	--

## مثال: گسترش 1D

$$A \oplus B = \{z \mid (\hat{B})_z \cap A \neq \emptyset\}$$

Input image

0	1	0	0	0	0	1	0	0	1	1	0
---	---	---	---	---	---	---	---	---	---	---	---



Structuring Element

1	1	1
---	---	---



Output Image

1	1	0							
---	---	---	--	--	--	--	--	--	--

# مثال: گسترش 1D

$$A \oplus B = \{z \mid (\hat{B})_z \cap A \neq \emptyset\}$$

Input image

0	1	0	0	0	0	1	0	0	1	1	0
---	---	---	---	---	---	---	---	---	---	---	---



Structuring Element

1	1	1
---	---	---



Output Image

1	1	0	0						
---	---	---	---	--	--	--	--	--	--

## مثال: گسترش 1D

$$A \oplus B = \{z \mid (\hat{B})_z \cap A \neq \emptyset\}$$

Input image

0	1	0	0	0	0	1	0	0	1	1	0
---	---	---	---	---	---	---	---	---	---	---	---



Structuring Element

1	1	1
---	---	---



Output Image

1	1	0	0	1					
---	---	---	---	---	--	--	--	--	--

## مثال: گسترش 1D

$$A \oplus B = \{z \mid (\hat{B})_z \cap A \neq \emptyset\}$$

Input image

0	1	0	0	0	0	1	0	0	1	1	0
---	---	---	---	---	---	---	---	---	---	---	---



Structuring Element

1	1	1
---	---	---

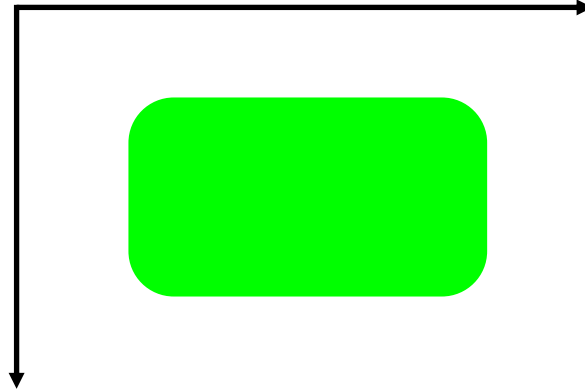
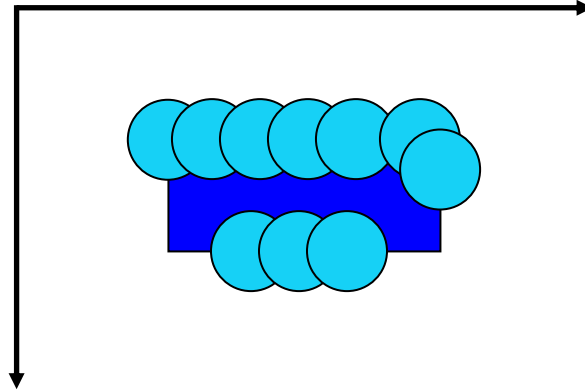
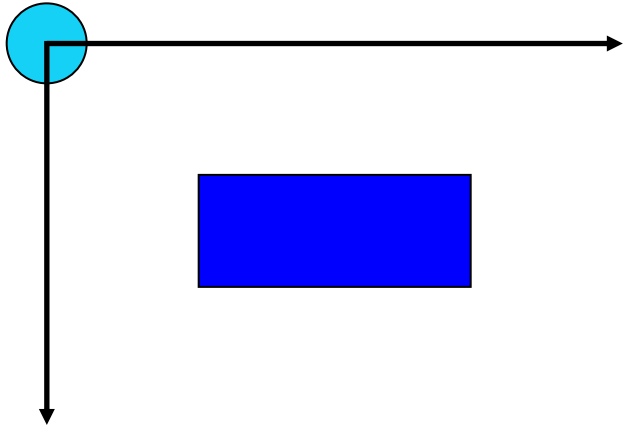


Output Image

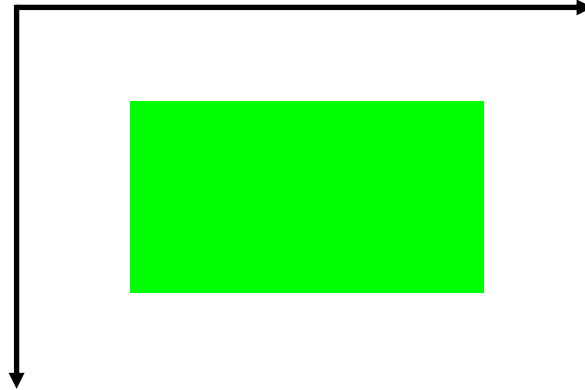
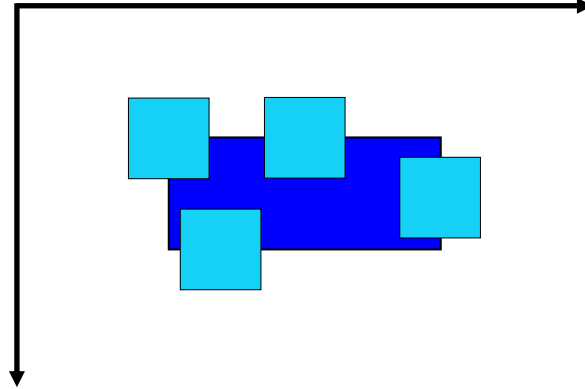
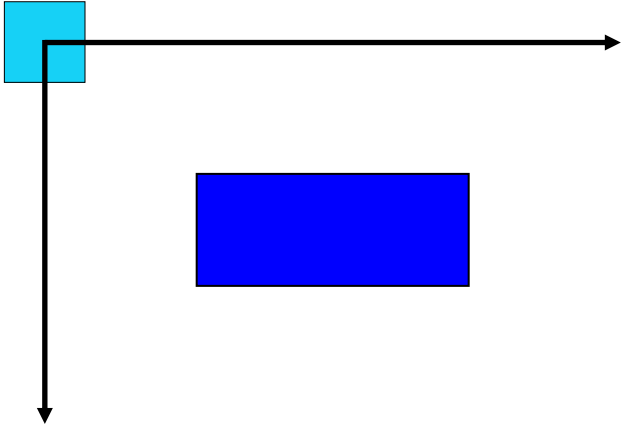
1	1	0	0	1	1	1	1	1	1
---	---	---	---	---	---	---	---	---	---



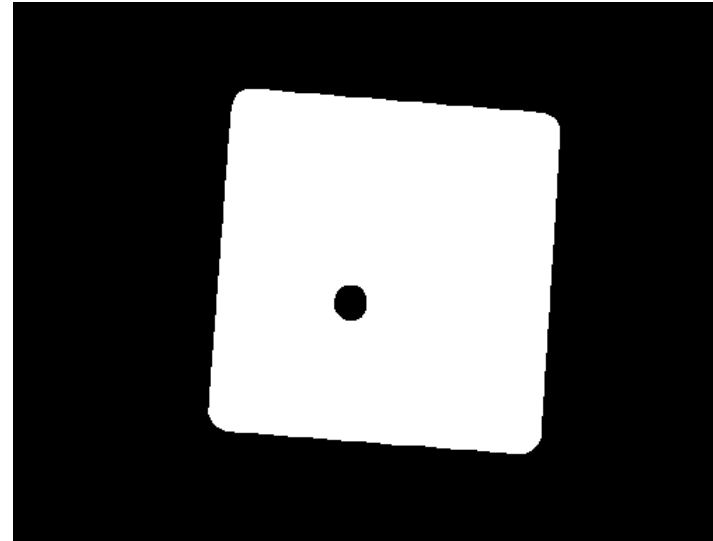
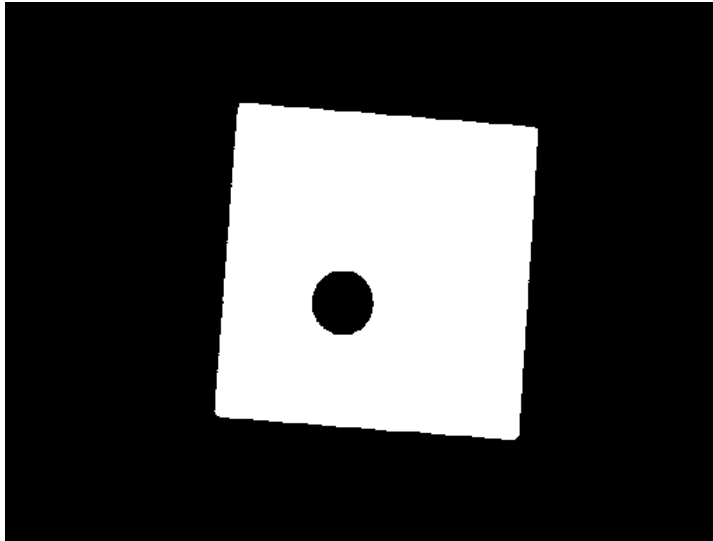
# مثال: گسترش 2D



## مثال: گسترش 2D



## مثال: گسترش 2D



0	0	1	1	1	0	0
0	1	1	1	1	1	0
1	1	1	1	1	1	1
1	1	1	1	1	1	1
1	1	1	1	1	1	1
0	1	1	1	1	1	0
0	0	1	1	1	0	0