۱.

10	10	20	20	20	20	20	20
10	10	20	20	20	20	20	20
20	20	20	20	30	20	20	30
10	10	20	30	30	20	30	30
20	20	30	30	30	30	30	30
20	30	30	30	30	30	30	20
20	30	30	30	30	30	30	20
20	20	20	30	20	20	30	20

گسترش (افزایش)

برای هر پیکسل مقدار جدیدش می‌شود ماکسیمم سطر بالایی و ستون چپی در یک همسایگی ۳x۳

10	10	10	10	10	10	10	10
10	10	10	10	10	10	10	10
10	10	10	10	20	10	10	10
10	10	10	10	10	10	10	10
10	10	10	10	10	10	10	10
10	10	10	10	10	10	10	10
10	10	10	10	10	10	10	10
20	20	20	20	10	10	10	10

سایش

برای هر پیکسل مقدار جدیدش می‌شود مینیمم سطر بالایی و ستون چپی در یک همسایگی ۳x۳

۲. می‌توانیم از ۸ عنصر ساختاری ۳x۳ مختلف استفاده کنیم که در همه‌ی این ۸ عنصر عضو وسطی ۱ باشد و در هر یک از این ۸ عنصر هربار یکی از همسایه‌های عضو وسطی لزوما صفر باشد (یعنی در عنصر ساختاری ۱- باشد)

-1	0	0
0	1	0
0	0	0

0	-1	0
0	1	0
0	0	0

0	0	-1
0	1	0
0	0	0

0	0	0
0	1	-1
0	0	0

0	0	0
-1	1	0
0	0	0

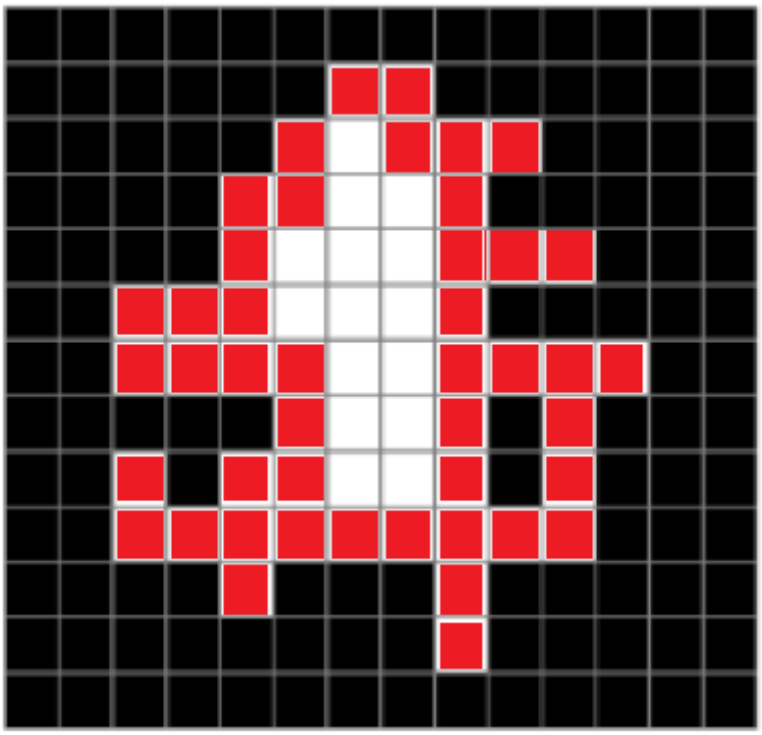
0	0	0
0	1	0
-1	0	0

0	0	0
0	1	0
0	-1	0

0	0	0
0	1	0
0	0	-1

عنصرهای ساختاری بالا می‌توانند ۲x۲ هم باشند ولی برای ساده‌تر مشخص شدن عنصر وسطی از ۳x۳ استفاده کردم.

با اعمال عملگر Hit-or-Miss برای ۸ عنصر ساختاری بالا و اجتماع گرفتن از نقاط ۱ حاصل (OR کردن حاصل‌ها) همانطور که در تصویر زیر مشخص شده هر نقطه‌ی سفیدی که یک همسایه سیاه داشته باشد به عنوان مرز مشخص می‌شود.



۳.الف

در تصویر داده شده پیکسل‌های روشن بک‌گراند هستند و پیکسل‌های تیره خود شی هستند. ابتدا با not کردن، جای سیاه و سفید را عوض کردم و بعد در انتهای کار دوباره not کردم تا مثل ابتدای کار شود.

برای پیاده‌سازی اعمال مورفولوژی از روابط موجود در اسلاید۱۷ و یک کرنل دایره‌ای به ابعاد ۳x۳ استفاده کردم.

```
In [32]: def S(A, k, B):
             A_kB = cv2.erode(A, B, iterations=k)
             A_kBoB = cv2.morphologyEx(A_kB, cv2.MORPH_OPEN, B)
             return cv2.subtract(A_kB, A_kBoB)
```

Implement this function to get the input image and return the skeleton of the input image.

```
In [39]: def get_skeleton(image):
             """
             Finds the skeleton of the input image.

             Parameters:
                 image (numpy.ndarray): The input image.

             Returns:
                 numpy.ndarray: The skeleton image.
                 numpy.ndarray: The parameters required for reconstructing image
             """
             res = np.zeros_like(image)
             params = None

             A = cv2.bitwise_not(image)
             B = cv2.getStructuringElement(cv2.MORPH_ELLIPSE,(3,3)) # kernel

             k = 0
             while cv2.countNonZero(cv2.erode(A, B, iterations=k)):
                 res = cv2.bitwise_or(res, S(A, k, B))
                 k += 1

             return cv2.bitwise_not(res), params
```

۳.ب

در هر مرحله $S_k(A)$ را در تابع get_skeleton ذخیره کردم و در یک لیست ریختم.

```
k = 0
while cv2.countNonZero(cv2.erode(A, B, iterations=k)):
    res = cv2.bitwise_or(res, S(A, k, B))
    k += 1

    params.append(S(A, k, B))
```

در تابع recons از این مقادیر ذخیره شده استفاده کردم و طبق رابطه‌ی موجود در پایین اسلاید۱۷ دوباره تصویر را بازسازی کردم. یعنی هربار S_k را kبار dilate کردم و تصاویر حاصل را اجتماع گرفتم (OR کردم)

```
B = cv2.getStructuringElement(cv2.MORPH_ELLIPSE,(3,3)) # kernel

res = np.zeros_like(image)

#Write your code here
k = 0
for param in params:
    res = cv2.bitwise_or(cv2.dilate(param, B, iterations=k), res)
    k += 1

return cv2.bitwise_not(res)
```

منابع:

سوال۳: Skeletonization in Python using OpenCV | by Neeramitra Reddy | Analytics Vidhya | Medium