2018222_CV_Midsem

March 1, 2021

0.0.1 CSE 344 - Computer Vision Midsem Exam

Name : Arka Sarkar **Roll Number :** 2018222

[154]:
```
import numpy as np
import cv2
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
from scipy.spatial import distance
from sklearn import metrics
```

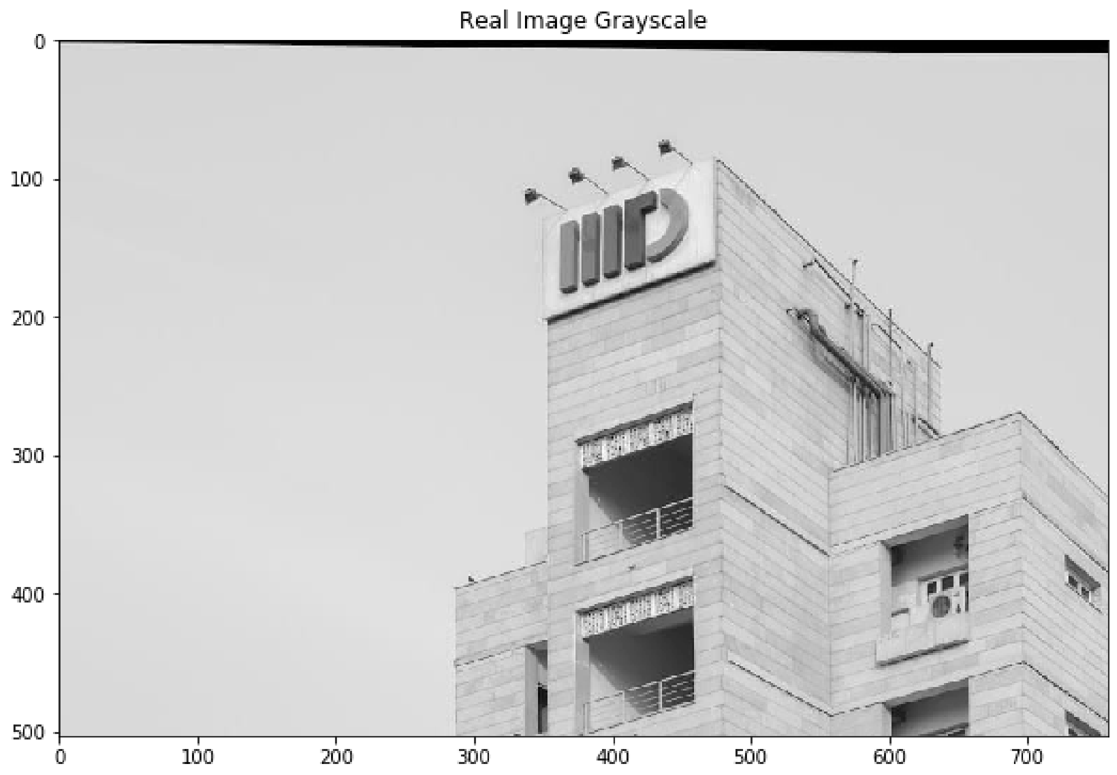
0.0.2 Question 1

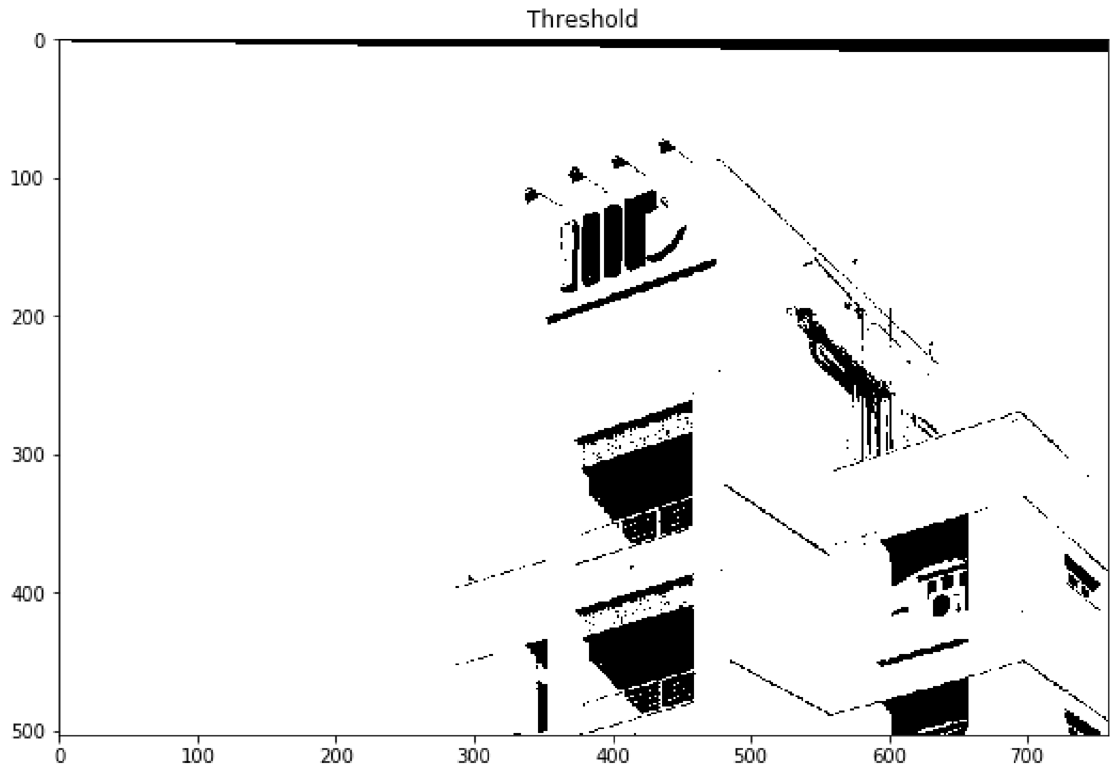
Just like Otsu's method, write a program to find an optimal threshold value that minimizes the sum of the Total Sum of Squares (TSS) of the two classes formed by the threshold. Use the given grayscale image to output your results.

[155]:
```
def otsu_tts(image):
    min_cost = float('inf')
    threshold = 0

    for i in range(1,256):
        v0 = img[img < i]
        s0 = np.sum((v0 - np.mean(v0))**2)
        w0 = len(img[img < i])
        v1 = img[img >= i]
        s1 = np.sum((v1 - np.mean(v1))**2)
        w1 = len(img[img >= i])

        cost = s0 + s1
        if(cost < min_cost):
            min_cost = cost
            threshold = i

    return threshold
```

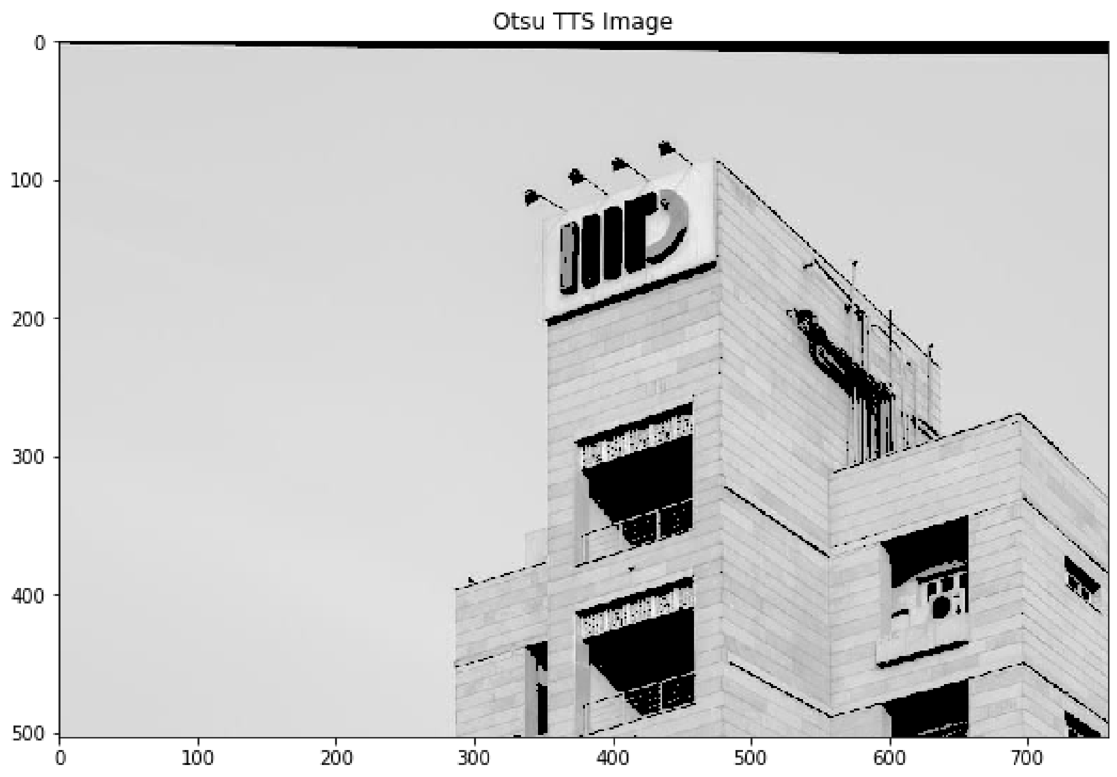
[156]:
```
def select_foreground(img, s = 0):
    if(s == 0):
        m,n = img.shape
        m0 = int(0.15*m)
        n0 = int(0.15*n)
        c0 = 0
        c1 = 0
        for i in range(m//2 - m0, m//2 + m0):
            for j in range(n//2 - n0, n//2 + n0):
                if(img[i,j] == 0):
                    c0 = c0 + 1
                else:
                    c1 = c1 + 1
        if(c0 > c1):
            return 0
        else :
            return 1
    else :

        m,n = img.shape

        m0 = 10
        n0 = 10
        c0 = 0
        c1 = 0
        for i in range(m):
            for j in range(n):
                if (i < m0):
                    if(img[i,j] == 0):
                        c0 = c0 + 1
                    else:
                        c1 = c1 + 1

                elif (i > m-m0):
                    if(img[i,j] == 0):
                        c0 = c0 + 1
                    else:
                        c1 = c1 + 1
                elif (j < n0):
                    if(img[i,j] == 0):
                        c0 = c0 + 1
                    else:
                        c1 = c1 + 1
                elif (j > n-n0):
                    if(img[i,j] == 0):
                        c0 = c0 + 1
                    else:
                        c1 = c1 + 1
        if(c0 > c1):
            return 0
        else :
            return 1
```

[167]:
```
img = cv2.imread("iiitd1.png",2)

plt.figure(figsize=(10,10))
plt.imshow(img, cmap = 'gray')
plt.title("Real Image Grayscale")
plt.show()

img.astype(int)
thres = otsu_tts(img)

img[img<thres] = 0
img[img >= thres] = 1
fore = select_foreground(img,1)
plt.figure(figsize=(10,10))
plt.imshow(img, cmap = 'gray')
plt.title("Threshold")
plt.show()

print("Otsu TTS Threshold : ",thres)
img2 = cv2.imread("iiitd1.png", 2)
img = cv2.imread("iiitd1.png", 2)
if(fore == 0):
    img[img <= thres]= 1
    img[img > thres]= 0
else :
    img[img <= thres]= 0
    img[img > thres]= 1

# m,n = img2.shape
# for i in range(m):
#     for j in range(n):
#         if(img[i,j] == 255):
#             img2[i,j] = 255
img2 = cv2.bitwise_and(img2, img2, mask = img)
plt.figure(figsize=(10,10))
plt.imshow(img2, cmap = 'gray')
plt.title("Otsu TTS Image")
plt.show()
```


Real Image Grayscale
0
100
200
300
400
500
0
100
200
300
400
500
600
700


Threshold
0
100
200
300
400
500
0
100
200
300
400
500
600
700

```
Otsu TTS Threshold :  138
```



0.0.3 Question 2

Silhouette Score and How does it work ? Silhouette Score is used to evaluate the quality of clusters created using the clustering algorithms such as KMeans. It measure how well the samples are clustered with the other similar samples clustered together. The Silhouette score (**s(i)**) is calculated for each sample. Higher the silhuotte score better is the clustering.

$$\mathrm{s}(i) = \begin{cases} 1 - \frac{a(i)}{b(i)} & \text{if } a(i) < b(i) \\ \frac{b(i)}{a(i)} - 1 & \text{if } a(i) > b(i) \\ 0 & \text{if } a(i) = b(i) \\ 0 & \text{if } |C(i)| == 1 \end{cases}$$

We can write this in one formula as :

$$s(i) = \frac{b(i) - a(i)}{max(a(i), b(i))}$$

a(i) denotes mean intra-cluster distance, which is the mean distance between the observation and all other data points in the same cluster.

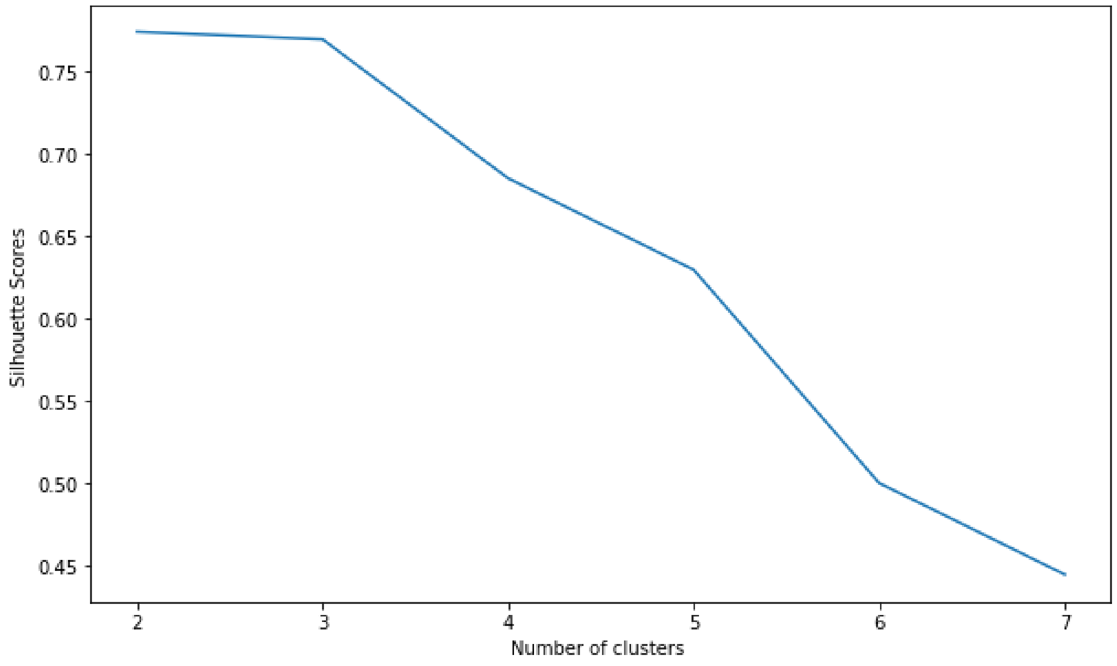
b(i) denotes mean nearest-cluster distance, which is the mean distance between the observation and all other data points of the next nearest cluster.

[158]:
```
def silhouette_score(clusters, labels):

    dict_ = {}
    m,n = clusters.shape
    epsilon = 10**-7
    for i in range(m):
        for j in range(n):
            if(labels[i,j] not in dict_):
                dict_[labels[i,j]] = []
                dict_[labels[i,j]].append(clusters[i,j])
            else:
                dict_[labels[i,j]].append(clusters[i,j])

    silhuoette_scores = np.zeros((m,n))

    for i in range(m):
        for j in range(n):
            cluster_points = dict_[labels[i,j]]
            if(len(cluster_points) > 1):
                curr_point = clusters[i,j]
                a = np.sum(((cluster_points - curr_point)**2)**0.5)/
 ↪(len(cluster_points) - 1)
                b = float('inf')
                for key in list(dict_.keys()):
                    if(key!=labels[i,j]):
                        cluster_points = dict_[key]
                        b = min(b,np.mean(((cluster_points - curr_point)**2)**0.
 ↪5))
                silhuoette_scores[i,j] = (b-a)/(max(a,b))
            else:
                silhuoette_scores[i,j] = 0

    return np.mean(silhuoette_scores)
```

[159]:
```
n_distint = len(np.unique(matrix.reshape((-1,1))))
silhouette_scores = []
for n in range(2,n_distint+1):
    X = matrix.reshape((-1,1))
    km = KMeans(n_clusters = n)
    km.fit(X)
    labels = km.labels_.reshape((-1,1))
    ss = silhouette_score(X,labels)
    silhouette_scores.append(ss)
    print("For n = ", n, ", Silhouette Scores : ", ss)
```

```
plt.figure(figsize = (10,6))
plt.plot([n for n in range(2,n_distint+1)],silhouette_scores )
plt.xlabel("Number of clusters ")
plt.ylabel("Silhouette Scores ")

plt.show()
```

```
For n =  2 , Silhouette Scores :  0.7742275847777773
For n =  3 , Silhouette Scores :  0.7696504132156419
For n =  4 , Silhouette Scores :  0.6851731620717241
For n =  5 , Silhouette Scores :  0.6296296296296295
For n =  6 , Silhouette Scores :  0.5
For n =  7 , Silhouette Scores :  0.4444444444444444
```



We get the best silhuotte score as 0.77422 for n = 2.

0.0.4 Question 4

Write a program to compute the LBP feature map of the given grayscale image. For comparing concerned pixels with their eight neighboring pixels while generating binary patterns, use min/max ratio. Round off the ratios to generate binary numbers, and use clockwise direction throughout. Display the feature map as another grayscale image. [min/max ratio of a&b is min(a,b)/max(a,b)]

[160]:
```
def calculateLbpPixel(image, x,y):

    center = image[x,y]
    arr = []
    epsilon = 10**-7

    if(min(image[x-1,y-1],center)/(max(image[x-1,y-1],center) + epsilon) >= 0.
↪5):
        arr.append(1)
    else:
        arr.append(0)

    if(min(image[x-1,y],center)/(max(image[x-1,y],center) + epsilon) >= 0.5):
        arr.append(1)
    else:
        arr.append(0)

    if(min(image[x-1,y+1],center)/(max(image[x-1,y+1],center) + epsilon) >= 0.
↪5):
        arr.append(1)
    else:
        arr.append(0)

    if(min(image[x,y+1],center)/(max(image[x,y+1],center)+ epsilon) >= 0.5):
        arr.append(1)
    else:
        arr.append(0)

    if(min(image[x+1,y+1],center)/(max(image[x+1,y+1],center) + epsilon) >= 0.
↪5):
        arr.append(1)
    else:
        arr.append(0)

    if(min(image[x+1,y],center)/(max(image[x+1,y],center) + epsilon) >= 0.5):
        arr.append(1)
    else:
        arr.append(0)

    if(min(image[x+1,y-1],center)/(max(image[x+1,y-1],center)+epsilon) >= 0.5):
        arr.append(1)
    else:
        arr.append(0)


    if(min(image[x,y-1],center)/(max(image[x,y-1],center) + epsilon) >= 0.5):
        arr.append(1)
    else:
        arr.append(0)
    bits = [128,64,32,16,8,4,2,1]

    val = 0

    for i in range(len(arr)):
        val+=arr[i]*bits[i]
    return val
```

[161]:
```
def generate_LBP_feature(image):
    m,n = image.shape
    pad_image = np.zeros((m+2,n+2))
    pad_image[1:m+1, 1:n+1] = image
    result = np.zeros((m,n))


    for i in range(1, m+1):
        for j in range(1,n+1):
            result[i-1,j-1] = calculateLbpPixel(pad_image, i,j)
    plt.figure(figsize = (10,10))
    plt.imshow(result, cmap = 'gray')
    plt.title("Image level LBP feature")
```

[162]:
```
img = cv2.imread("iiitd1.png",2)
generate_LBP_feature(img)
```

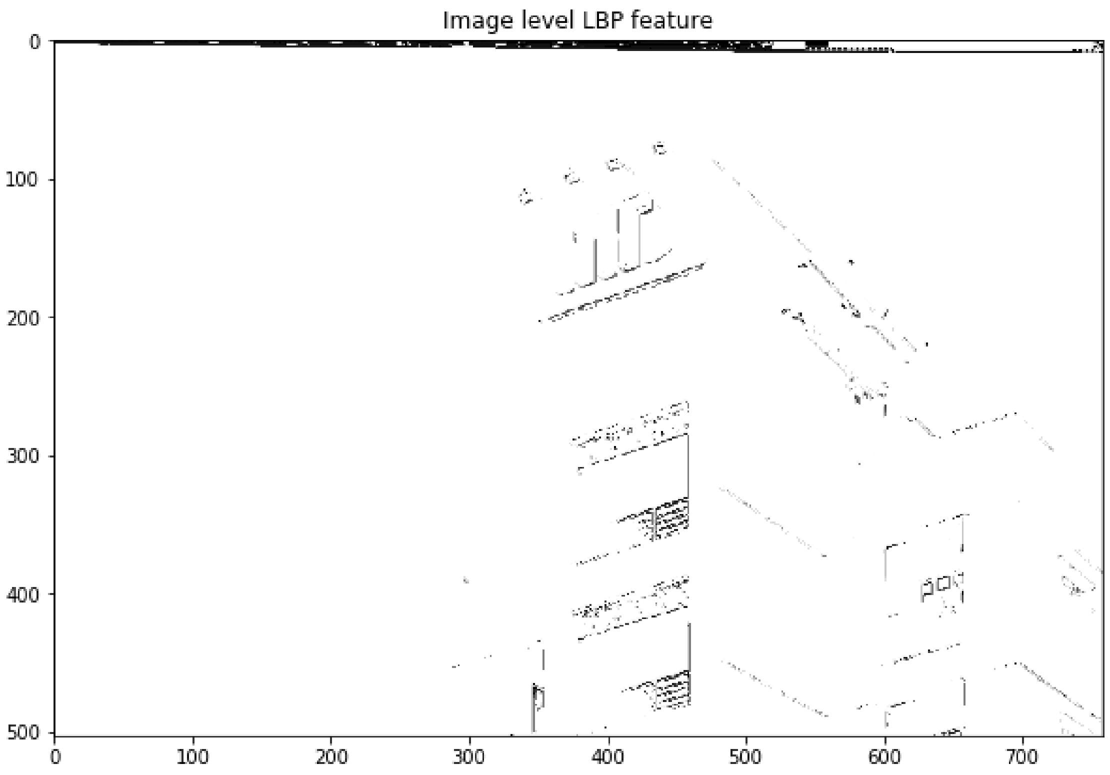
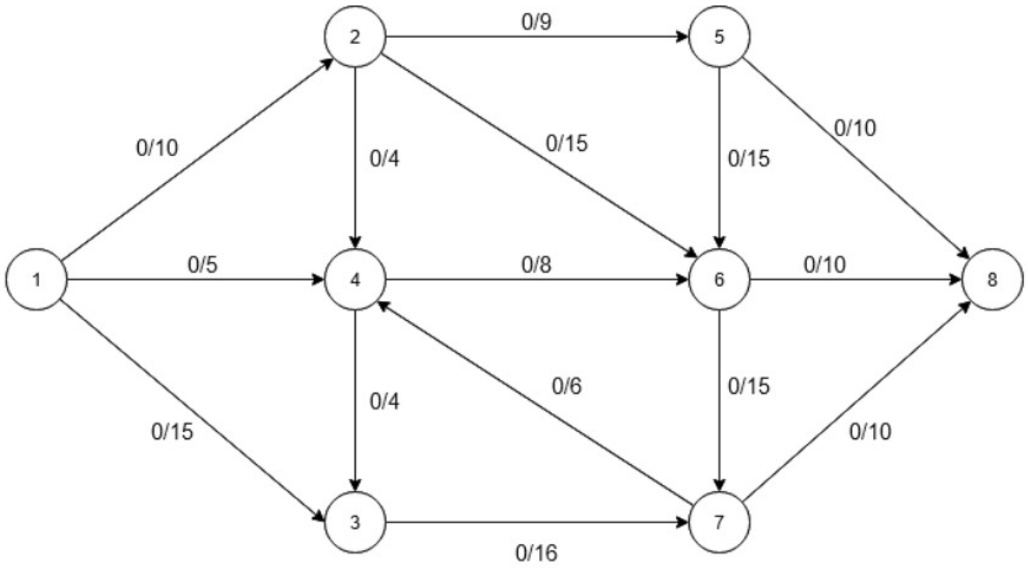
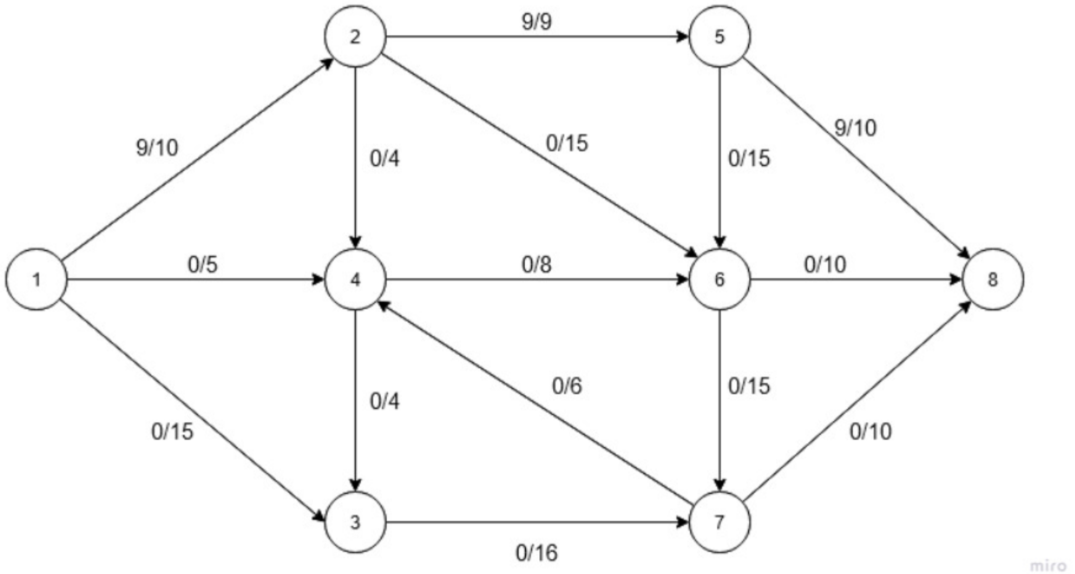


Image level LBP feature
0
100
200
300
400
500
0
100
200
300
400
500
600
700

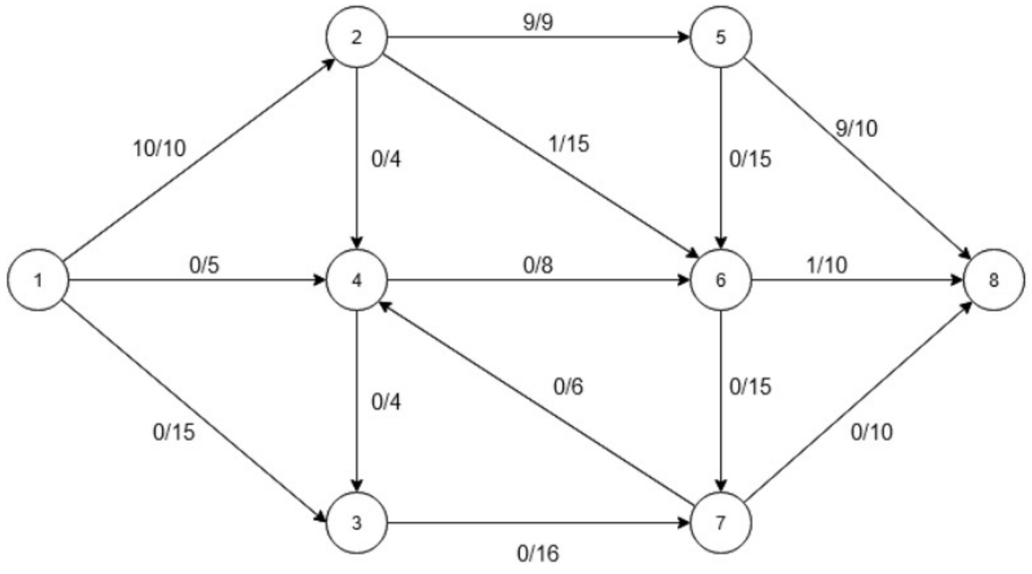
0.0.5 Question 5



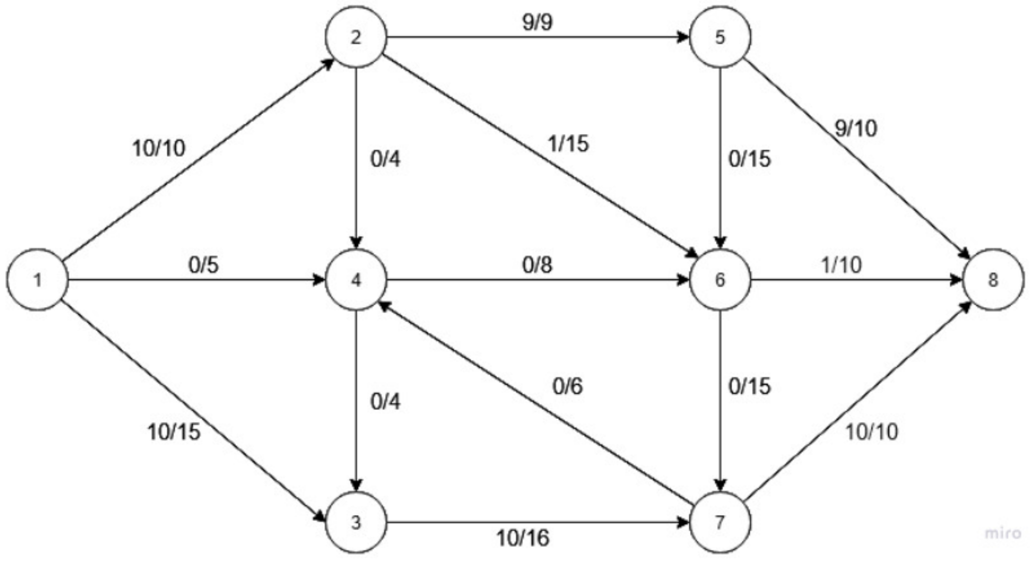
Augmenting Path-(1-> 2 -> 5 -> 8), Bottleneck = 9



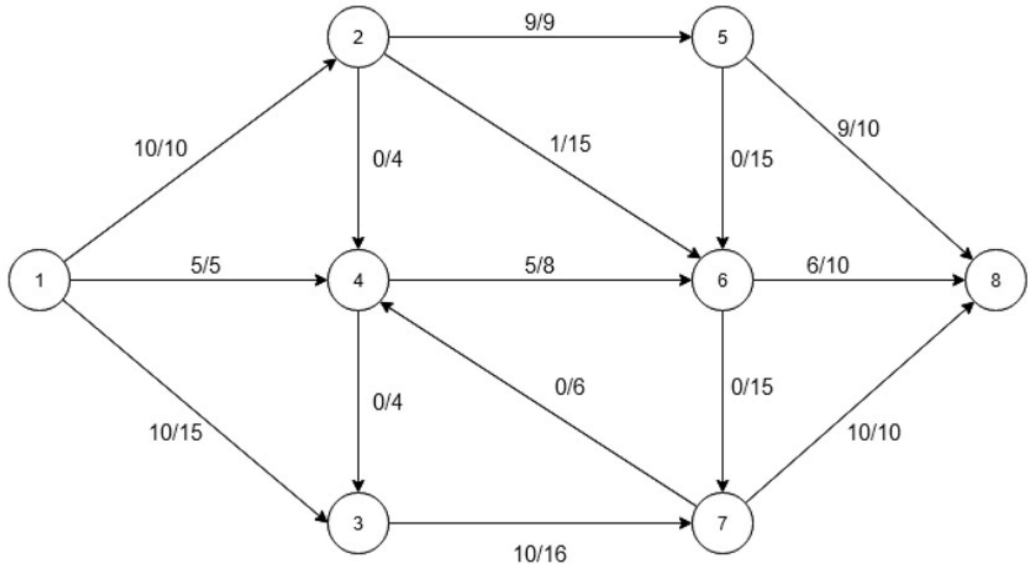
Augmenting Path-(1-> 2 -> 6 -> 8), Bottleneck = 1



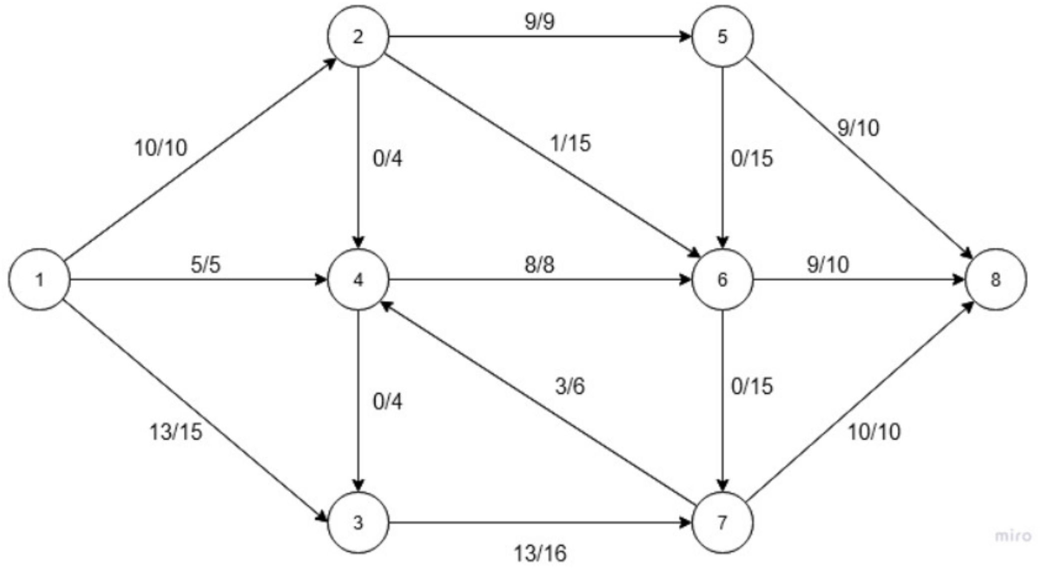
Augmenting Path-(1-> 3 -> 7 -> 8), Bottleneck = 10

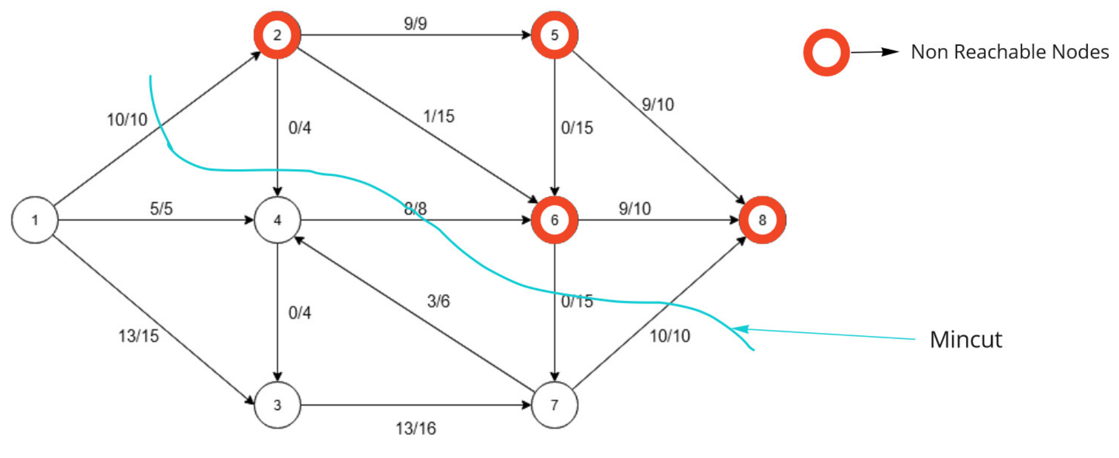


Augmenting Path-(1-> 4-> 6 -> 8), Bottleneck = 5



Augmenting Path-(1-> 3-> 7 -> 4 -> 6 -> 8), Bottleneck = 3





Mincut Edges- (1,2) , (4,6), (7,8), (2,4), (6,7)

0.0.6 Value of min cut = 10 + 8 + 10 + 0 + 0 = 28