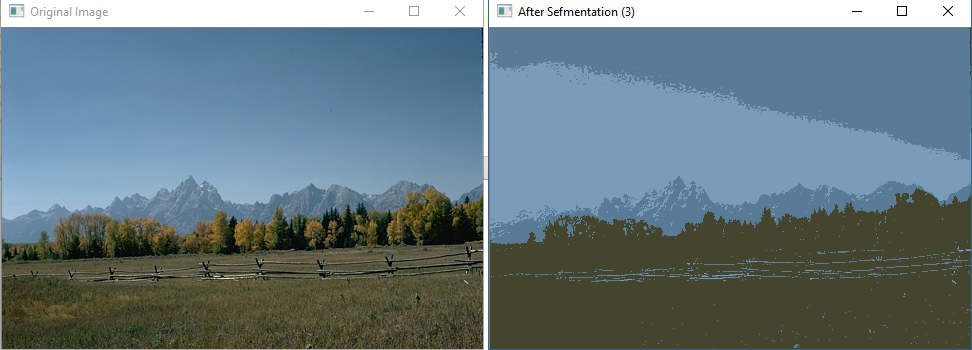
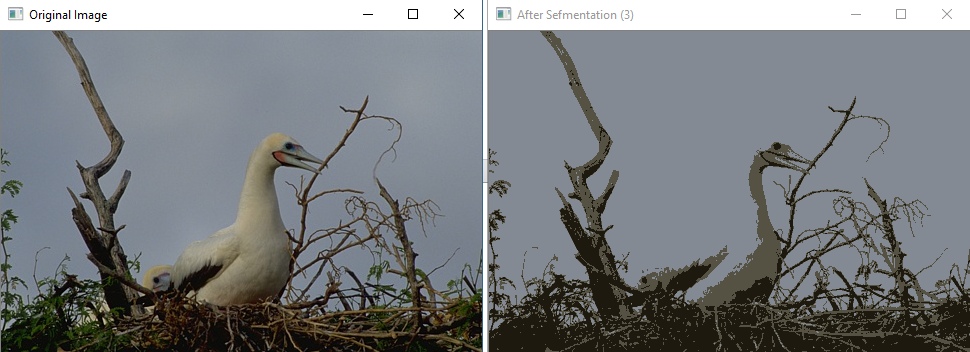
|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |

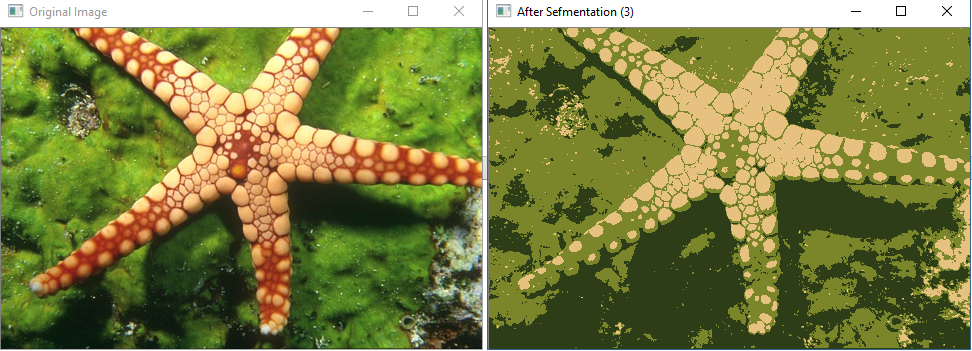
2 Color 3 Color 4 Color

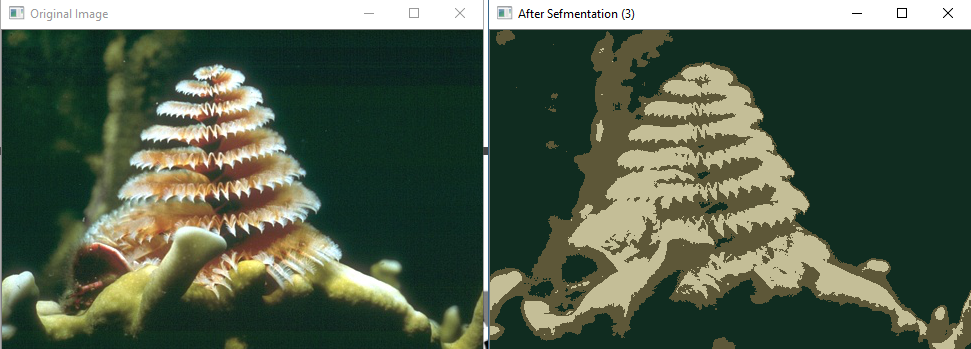
5 Color 6 Color 7 Color

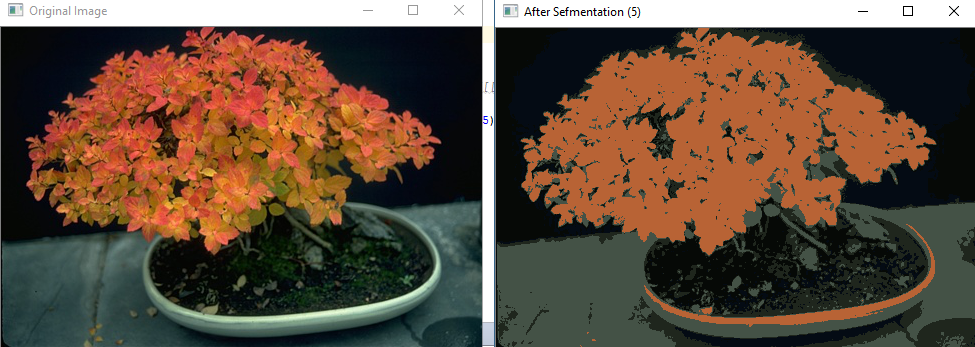
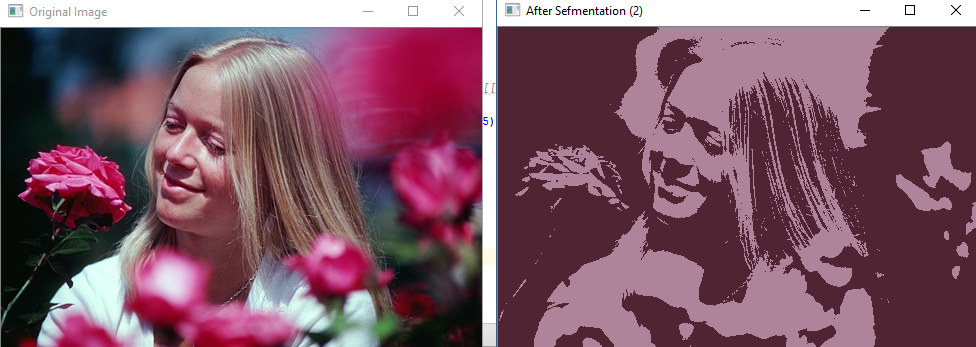
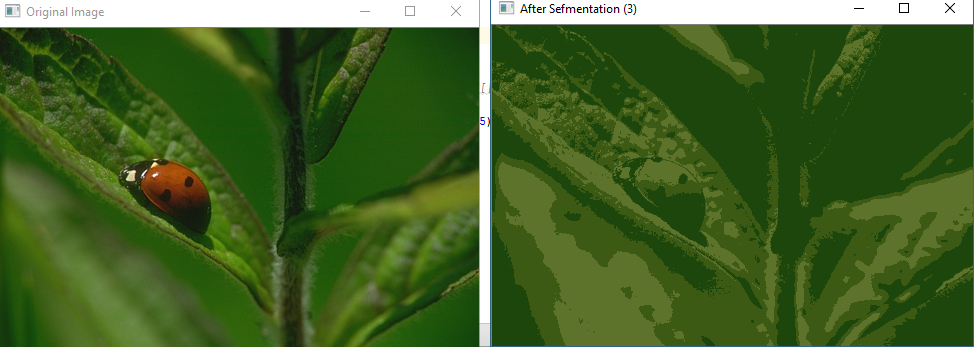


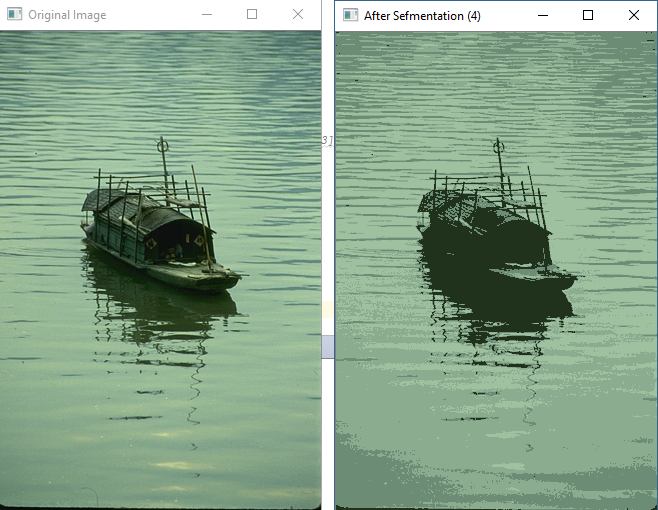


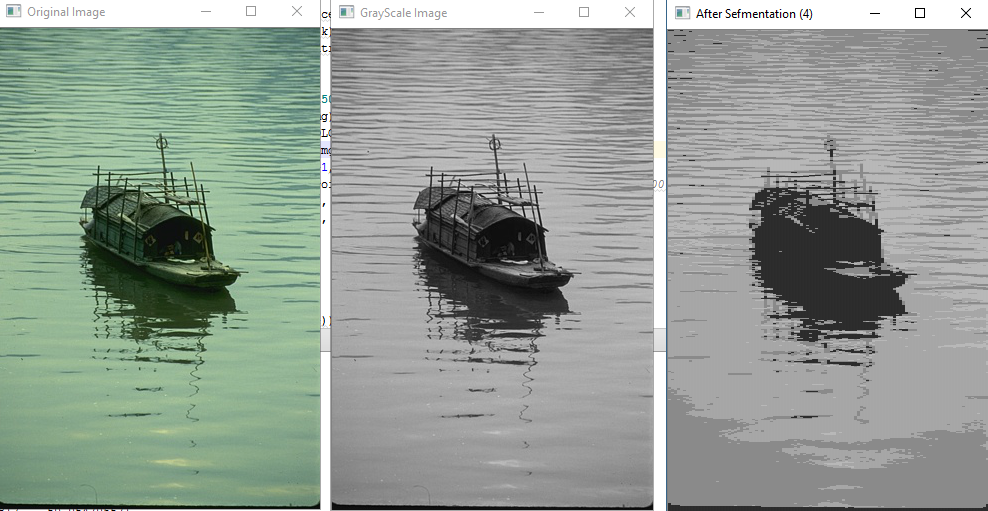


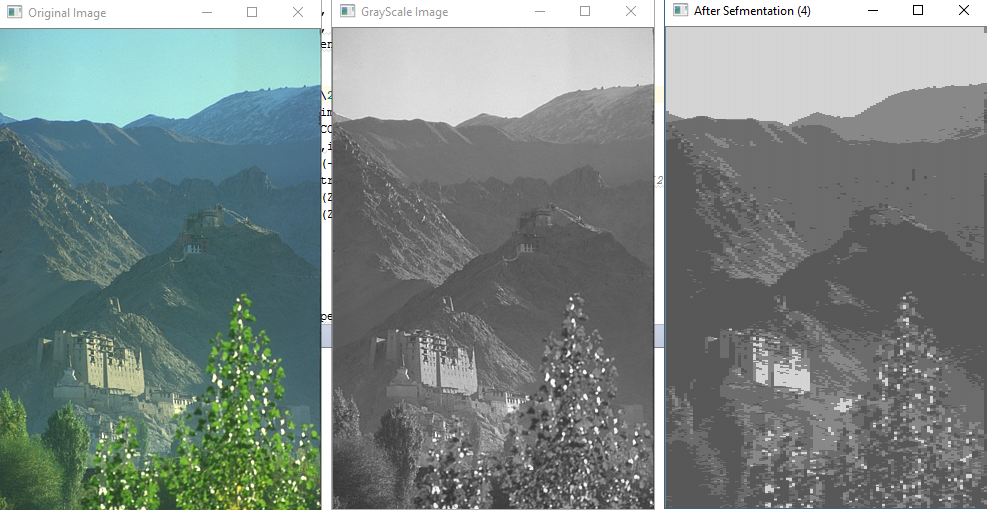


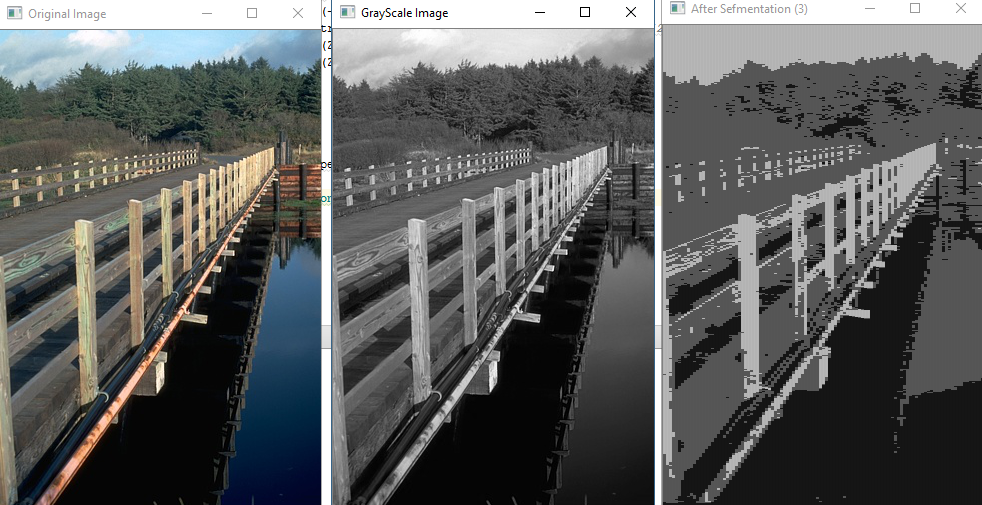


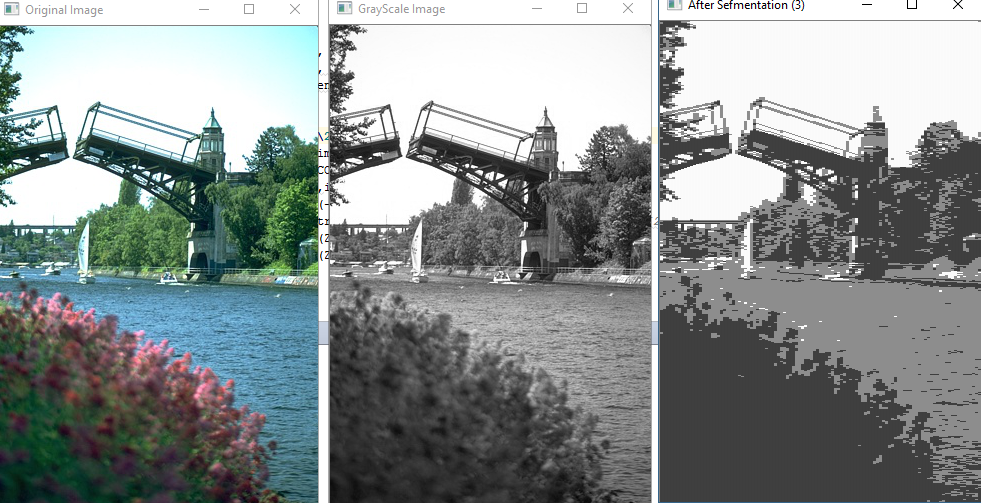


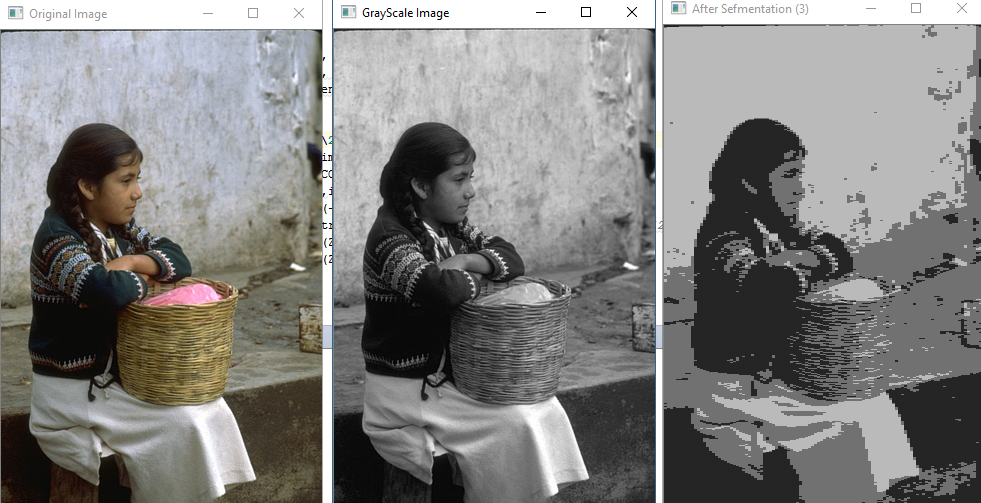


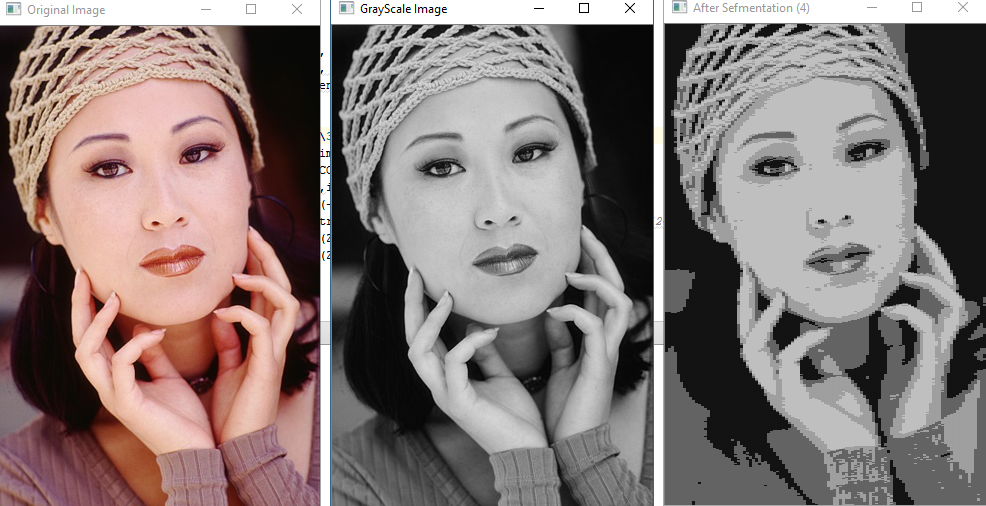


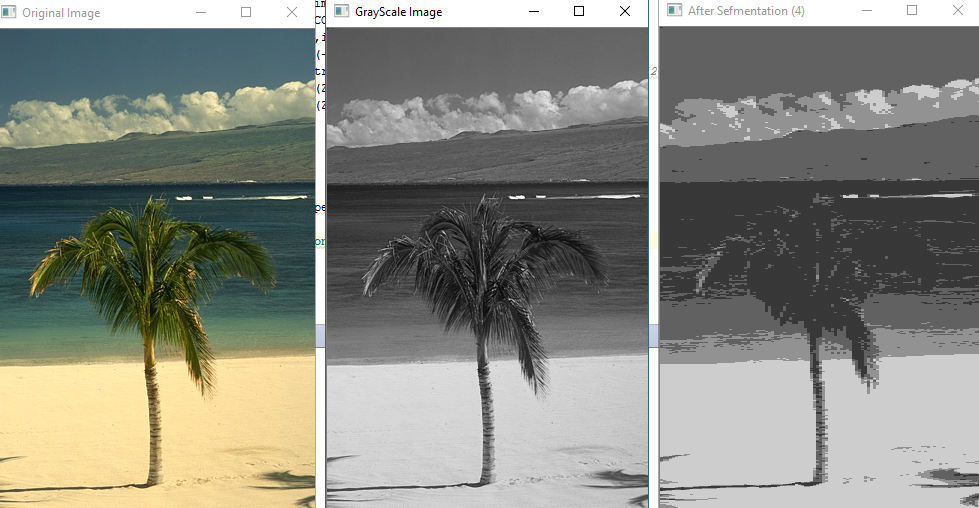


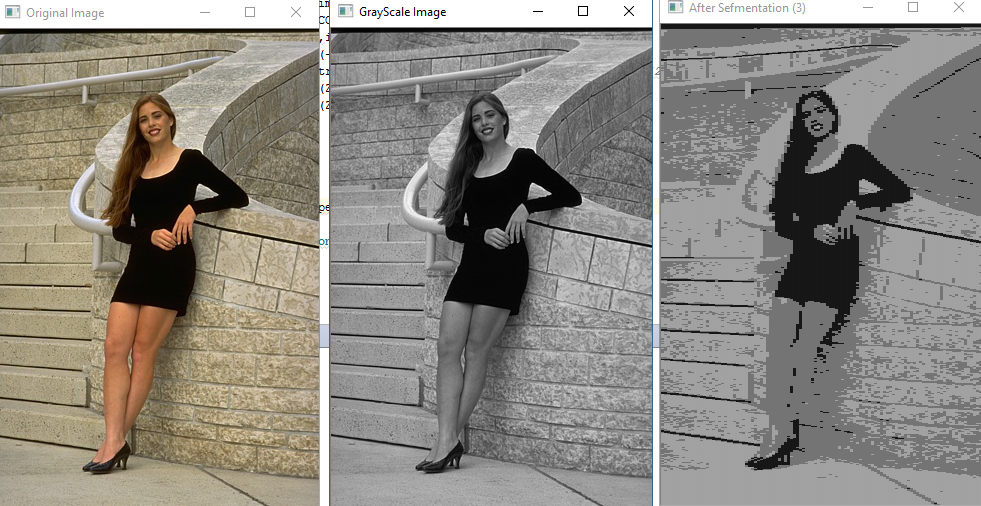








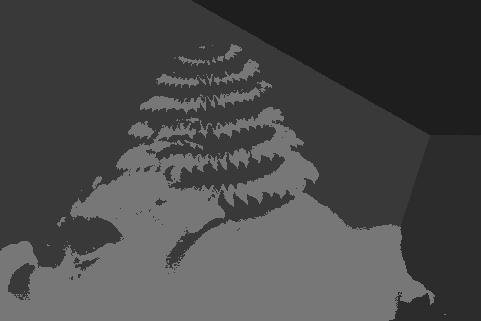




4means(R,G,B,Row,Col)

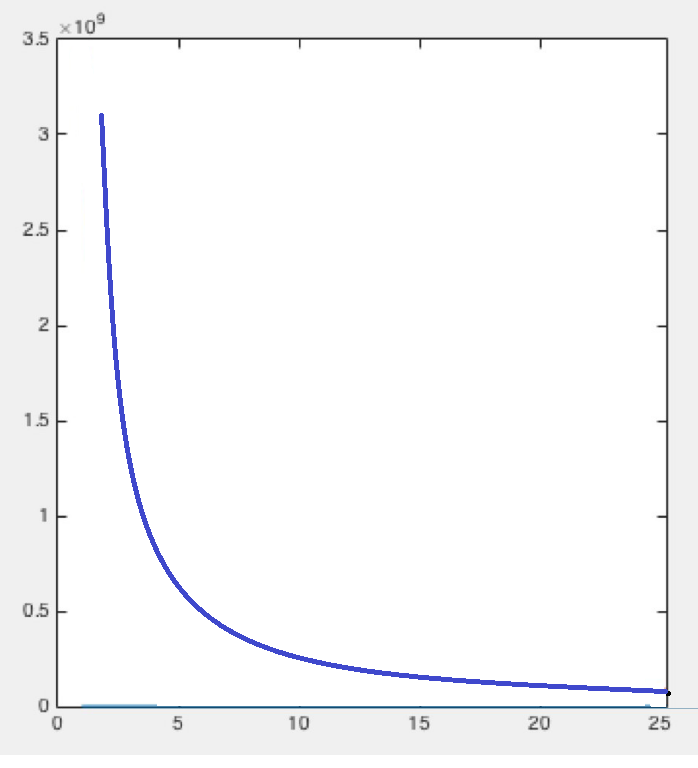


4means(R,G,B,Row,Col)



آ) انتخاب تصادفی مراکز بی تاثیر نیست ولی زمانی که ما مقدار آستانه را برای شرط توقف Kmean به اندازه کافی کوچک بگیریم این تغییرات ناچیز می شوند. در کد زده شده توسط بنده مرکز خوشه ها به صورت شانسی تولید میشود

ب) با اجرای الگوریتم با k های مختلف برنامه را اجرا

می کنیم و از یه نقطه ای به بعد اختلاف ها آن چنان محسوس

نیست و قابل چشم پوشی است در این الگوریتم و مثال

بهترین k عدد 5 انتخاب شده چرا که پیش از آن بسیار اختلاف ها

زیاد است و بیشتر از 5 اختلاف ها چندان محسوس نیست و فقط

بار محاسباتی و زمان اجرا افزایش می یابد

پ) به طبع هر چه تعداد k های ما بیشتر باشد فاصله نقاط از مرکز

به طور کلی کم می شود لیکن تابع زیاد کردن K منطقی نیست و باید

یک آستانه ای تعریف شود برای انتخاب K پس در الگوریتم انتخاب

K ما یک حد آستانه در نظر میگیریم که ضریبی از میزان تغییرات و

هزینه باش و یا فواصل باشد تا n عضو به n خوشه تقسیم نشوند.

ت( برای ارزیابی الگورتیم های بدون ناظر غالبا دو رویکرد پیش روست : 1\_ فاصله نقاط از مرکز : به این صورت که مجموع فاصله ی هر نقطه نسبت به مرکز خوشه ای که در آن است را به حداقل برسانیم

2\_ فاصله خوشه ها از یکدیگر : دوم آن که تا آن جا که امکان دارد فاصله خوشه ها از یکدیگر ماکسیمم باشد برای این کار در نظر گرفتن فاصله مراکز خوشه ها به نمایندگی نیز صحیح است

سوال 2 ) با توجه به این مسئله که یک کارت اگر در دو خود پرداز استفاده شود احتمالا این دو خود پرداز بهم نزدیک اند پس دو خودپرداز اگه تعداد کارت های مشترک بیشتری داشته باشند احتمال اینکه در یک شهر باشند بیشتر است حال میتوان زمان تراکنش را نیز در این امر لحاظ نمود به این صورت اگه زمان تراکنش یک کارت در دو خود پرداز از یه آستانه ای کمتر باشد با احتمال بسیار قوی این دو خود پرداز نزدیک بهم هستند یک آستانه برای شهر و یک آستانه برای استان در نظر میگیریم

پس در ابتدا ما شماره کارت هایی که در حداقل دو خود پرداز استقاده شده اند را استخراج میکنیم و زمان و شماره خود پرداز را می سنجیم دو خود پردازی که تراکنشی در کمترین فاصله زمانی داشته اند و یا بیشترین تراکنش مشترک رو داشته باشند با هم ترکیب میکنیم مثلا خود پرداز A و خود پرداز B انتخاب میشوند که در مرحله اول با هم خوشه بندی شوند . پس جدول را آپدیت میکنیم و هر جا در جدول به جای A , B در جدول AB می گذاریم . طبیعی است که انتخاب بین دو خود پرداز که یک تراکنش با فاصله زمانی کم دارند و دو خودپرداز که چندین تراکنش مشترک دارند نقطه چالش بر انگیز ماست . دو رویکرد متفاوت در خوشه بندی شهری و خوشه بندی استانی پیش روست . رویکرد اول اینکه ما خوشه بندی سلسله مراتبی را با همین روش ادامه دهیم و در آخر برای انتخاب خوشه شهر ها یک برش و خوشه استان ها یک برش بدهیم که ارتفاع برش شهر طبیعتا پایین تر است و رویکرد دوم اینکه اشتراک هایی که در فاصله زمانی کم رخ داده اند احتمال حضور دو خودپرداز در یک شهر و فاصله متوسط یا زیاد احتمال حضور در استان رو تقویب میکند .

ایده های بعدی میتوان خوشه بندی بر اساس مقدار موجودی باشد که احتمالا کلان شهر ها موجودی بیشتر و ... که نمیتواند معیار دقیقی باشد و یا ایده دیگر آن است که 3 رقم ابتدایی شماره کارت ها مربوط به منطقه خود پرداز تعلق میگیرد ..

**import** networkx **as** nx  
**import** matplotlib.pyplot **as** plt  
  
**def** importDS(path):   
 f1 = open(path,**'r'**)  
 ds=[]  
 **for** i,line **in** enumerate(f1):  
 **if** i>0 :  
 str1=line.replace(**'\n'**,**''**).split(**','**)  
 ds.append(str1)  
print(**'==============================================='**)  
 s2 =set()  
 d2 = {} *# a dictionary which key is number of card Id and value is ps id which servved that card* **for** i,line **in** enumerate(ds):  
 l2 = repr(line).replace(**"'"**,**''**).replace(**']'**,**''**).replace(**'['**,**''**).split(**','**)  
 s2.add(l2[1])  
 **for** line **in** ds:  
 l2 = repr(line).replace(**"'"**,**''**).replace(**']'**,**''**).replace(**'['**,**''**).split(**','**)  
 p = int(str(l2[1]).replace(**"'"**,**''**))  
 **if** p **in** d2.keys():  
 d2[p].add(int(str(l2[2]).replace(**"'"**, **''**)))  
 **else**:  
 d2[p] = set()  
 d2[p].add(int(str(l2[2]).replace(**"'"**,**''**)))  
 d3={}  
 **for** k **in** d2:  
 **if** len(d2[k]) > 1:  
 d3[k]=d2[k]  
 **return** d3  
  
   
  
 **def** HCluster(Dict1):  
 set1 = set()  
 faravaniDict = {}  
 **for** k, v **in** s2.items():  
 temp = str(v).replace(**'}'**,**''**).replace(**'{'**,**''**).split(**','**)  
 **for** row **in** temp:  
 set1.add(row)  
 print(str(k) + **' : '** + str(v))  
 **for** a **in** set1:  
 **for** b **in** set1:  
 **if** str(a).strip()!=str(b).strip() :  
 **for** k,v **in** s2.items():  
 **if** a **in** str(v) **and** b **in** str(v) :  
 **if** str(a).strip()+**'|'**+str(b).strip() **in** faravaniDict.keys() : faravaniDict[str(a).strip()+**'|'**+str(b).strip()]+=1  
 **elif** str(b).strip()+**'|'**+str(a).strip() **in** faravaniDict.keys() : faravaniDict[str(b).strip()+**'|'**+str(a).strip()]+=1  
 **else**: faravaniDict[str(a).strip()+**'|'**+str(b).strip()]=1  
 print(**'===================='**)  
 max = 0  
 maxIndice = **''  
 for** k,v **in** faravaniDict.items():  
 print(str(k)+ **' = '**+str(v))  
 **if** v>max :  
 max = v  
 maxIndice = k  
  
 print(maxIndice)  
 i = maxIndice.index(**'|'**)  
 print(i)  
 **for** k,v **in** faravaniDict.items():  
 j= k.index(**'|'**)  
 print(j)  
 *#while(len(faravaniDict.keys())!=1):*s2 = importDS(**'transactiondata.csv'**)  
HCluster(s2)

متد importDS مسیر دیتاست را میگیرد و جدول شماره کارت هایی را می دهد که از بیش از 2 خودپرداز استفاده کرده اند و متد Hcluster بزرگترین مشترک ها رو پیدا میکند و با هم یکی میکند و کار را انقدر تکرار میکند که یک خوشه بیشتر نمانده باشد. و یک جدول بر میگرداند که نشان دهنده آن است که در هر مرحله کدام دو دسته با هم merg شده اند.