8E and 8F: Finding the Probability P(Y==1|X)

8E: Implementing Decision Function of SVM RBF Kernel

After we train a kernel SVM model, we will be getting support vectors and their corresponsing coefficients α_i

Check the documentation for better understanding of these attributes:

https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html

As a part of this assignment you will be implementing the `decision_function()` of kernel SVM, here decision_function() means based on the value return by `decision_function()` model will classify the data point either as positive or negative

Ex 1: In logistic regression After traning the models with the optimal weights w we get, we will find the value $\frac{1}{1+\exp(-(wx+b))}$, if this value comes out to be < 0.5 we will mark it as negative class, else its positive class

Ex 2: In Linear SVM After traning the models with the optimal weights w we get, we will find the value of $sign(wx+b)$, if this value comes out to be -ve we will mark it as negative class, else its positive class.

Similarly in Kernel SVM After traning the models with the coefficients α_i we get, we will find the value of $sign(\sum_{i=1}^{n}(y_i\alpha_i K(x_i,x_q)) + intercept)$, here $K(x_i,x_q)$ is the RBF kernel. If this value comes out to be -ve we will mark x_q as negative class, else its positive class.

RBF kernel is defined as: $K(x_i,x_q) = exp(-\gamma ||x_i - x_q||^2)$

For better understanding check this link: https://scikit-learn.org/stable/modules/svm.html#svm-mathematical-formulation

Task E

> 1. Split the data into X_{train}(60), X_{cv}(20), X_{test}(20)
> 2. Train $SVC(gamma=0.001, C=100.)$ on the (X_{train}, y_{train})
> 3. Get the decision boundry values f_{cv} on the X_{cv} data i.e. f_{cv} `= decision_function(` X_{cv} `)` you need to implement this decision_function()

In [1]:
```
import numpy as np
import pandas as pd
from sklearn.datasets import make_classification
import numpy as np
from sklearn.svm import SVC
```

In [2]:
```
X, y = make_classification(n_samples=5000, n_features=5, n_redundant=2,
                           n_classes=2, weights=[0.7], class_sep=0.7, random_state=15)
```

In [3]:
```
#Splitting Data for tain_test, and CV
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split( X, y, test_size=0.4, random_state=42)#splitting in the ratio of 60:40
X_CV, X_test, y_CV, y_test = train_test_split( X_test, y_test, test_size=0.5, random_state=42)#splitting in the ration 60,20,20
```

Pseudo code

clf = SVC(gamma=0.001, C=100.)
clf.fit(Xtrain, ytrain)

def decision_function(Xcv, ...): #use appropriate parameters
  for a data point x_q in Xcv:
    #write code to implement $(\sum_{i=1}^{\text{all the support vectors}}(y_i\alpha_i K(x_i,x_q)) + intercept)$, here the values y_i, α_i, and $intercept$ can be obtained from the trained model
return *# the decision_function output for all the data points in the Xcv*

fcv = decision_function(Xcv, ...) # *based on your requirement you can pass any other parameters*

Note: Make sure the values you get as fcv, should be equal to outputs of clf.decision_function(Xcv)

In [4]:
```
# you can write your code here
```

In [5]:
```
clf=SVC(gamma=0.001,C=100,kernel='rbf')
clf.fit(X_train,y_train)#Trainig the SVC model
```

Out[5]: `SVC(C=100, gamma=0.001)`

In [6]:
```
def k(xi,xq,gamma):#kernel function that
    temp=0
    from math import exp
    x=xq-xi
    x=x**2
    x=x*gamma
    x=sum(x)
    return exp(-x)#return a value
```

In [7]:
```
def decision_function(X_CV,model,SV,gamma):
    fcr=[]
    for xq in X_CV:# for every point in X_CV
        temp=0#storing in a variable to sum up with all the SV vectors
        for a,xi in zip(model.dual_coef_.flatten(),SV):
            temp=temp+(a*(k(xi,xq,gamma)))
        fcr.append(temp+model.intercept_[0])
    return fcr
#SV----> support vectors of size 5 dimensions
#a----> yi*alphai we can get this value from model.dual_coef_
```

In [8]:
```
SV=clf.support_vectors_
temp2=decision_function(X_CV,clf,SV,0.001)
```

In [9]:
```
temp=clf.decision_function(X_CV)
```

In [ ]:

In [10]:
```
print(f'Number of points equal in both the sets = {(temp==temp2).sum()}')
fcv=temp2
```

```
Number of points equal in both the sets = 979
```

8F: Implementing Platt Scaling to find P(Y==1|X)

Check this PDF

TASK F

> 1. Apply SGD algorithm with (f_{cv}, y_{cv}) and find the weight W intercept b `Note: here our data is of one dimensional so we will have a one dimensional weight vector i.e W.shape (1,)`
>
> Note1: Don't forget to change the values of y_{cv} as mentioned in the above image. you will calculate y+, y- based on data points in train data
>
> Note2: the Sklearn's SGD algorithm doesn't support the real valued outputs, you need to use the code that was done in the `'Logistic Regression with SGD and L2'` Assignment after modifying loss function, and use same parameters that used in that assignment. if Y[i] is 1, it will be replaced with y+ value else it will replaced with y- value
>
> 1. For a given data point from X_{test}, $P(Y=1|X) = \frac{1}{1+exp(-(W*f_{test}+b))}$ where f_{test} `= decision_function(` X_{test} `)`, W and b will be learned as metioned in the above step

Note: in the above algorithm, the steps 2, 4 might need hyper parameter tuning, To reduce the complexity of the assignment we are excluding the hyerparameter tuning part, but intrested students can try that

If any one wants to try other calibration algorithm istonic regression also please check these tutorials

1. http://fa.bianp.net/blog/tag/scikit-learn.html#fn:1
2. https://drive.google.com/open?id=1MzmA7QaP58RDzocB0RBmRiWfl7Co_VJ7
3. https://drive.google.com/open?id=133odBinMOIVb_rh_GQxxsyMRyW-Zts7a
4. https://stat.fandom.com/wiki/Isotonic_regression#Pool_Adjacent_Violators_Algorithm

In [11]:
```
Y_plus=(y_train[y_train==1].sum()+1)/(y_train[y_train==1].sum()+2)
Y_minus=1/(y_train[y_train==0].sum()+2)
```

In [12]:
```
t = lambda x,y_plus,Y_minus: Y_plus if x==1 else Y_minus
```

In [13]:
```
f_cv_Y=np.array([t(x,Y_plus,Y_minus) for x in y_CV])
```

In [14]:
```
f_cv_X=decision_function(X_CV,clf,SV,0.001)
```

In [15]:
```
b=0
w=0
```

In [16]:
```
print(b)
print(w)
```

```
0
0
```

In [17]:
```
def pred(w,b, X):
    N = len(X)
    predict = []
    for i in range(N):
        z=np.dot(w,X[i])+b
        predict.append(sigmoid(z))
    return np.array(predict)
```

In [18]:
```
def initialize_weights(row_vector):
    ''' In this function, we will initialize our weights and bias'''
    #initialize the weights as 1d array consisting of all zeros similar to the dimensions of row_vector
    #you use zeros_li`ke function to initialize zero, check this link https://docs.scipy.org/doc/numpy/reference/generated/numpy.zeros_like.html
    #initialize bias to zero
    w=np.zeros_like(row_vector)
    b=0
    return w,b
```

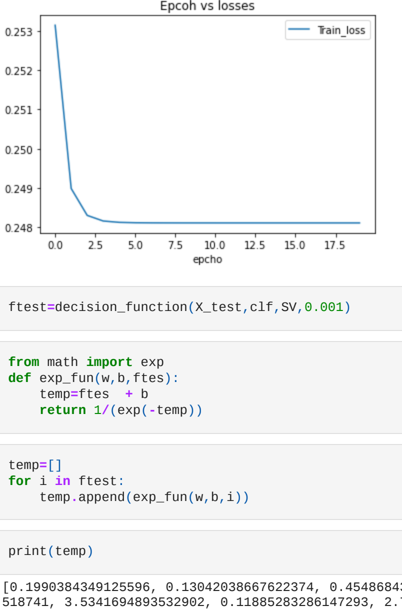
In [19]:
```
#make sure that the sigmoid function returns a scalar value, you can use dot function operation
def gradient_dw(x,y,w,b,alpha,N,eta):
    '''In this function, we will compute the gardient w.r.to w '''
    #temp1=np.sum(np.dot(w.T,x),b)
    temp=np.dot(w.T,x) + b
    si=sigmoid(temp)
    temp2=np.subtract(y,si)
    temp3=np.multiply(x,temp2)
    learning_rate=eta
    temp4=learning_rate/N  * w
    dw=np.subtract(temp3,temp4)
    return dw
```

In [20]:
```
from math import exp
def sigmoid(z):
    ''' In this function, we will return sigmoid of z'''
    # compute sigmoid(z) and return
    return 1/(1+exp(-z))
#sb should be a scalar value
def gradient_db(x,y,w,b):
     '''In this function, we will compute gradient w.r.to b '''
     temp=np.dot(w.T,x) + b
     si=sigmoid(temp)
     db=np.subtract(y,si)
     return db
import numpy as np
def logloss(y_true,y_pred):
    # you have been given two arrays y_true and y_pred and you have to calculate the logloss
    #while dealing with numpy arrays you can use vectorized operations for quicker calculations as compared to using loops
    #https://www.pythonlikeyoumeanit.com/Module3_IntroducingNumpy/VectorizedOperations.html
    #https://www.geeksforgeeks.org/vectorized-operations-in-numpy/
    #write your code here
    y_true2=np.ones_like(y_true)
    y_pred2=np.ones_like(y_pred)
    y_true3=np.subtract(y_true2,y_true)
    y_pred3=np.subtract(y_pred2,y_pred)
    loss= np.sum(np.add(np.multiply(y_true, np.log10(y_pred)), np.multiply(y_true3 , np.log10(y_pred3))))
    return -1*loss/len(y_true)
```

In [21]:
```
def train(X_train,y_train,epochs,alpha,eta0):
    ''' In this function, we will implement logistic regression'''
    train_loss = []
    w,b = initialize_weights(X_train[0])# Initialize the weights
    tempw,tempb=0,0
    #write your code to perform SGD
    N=len(f_cv_X)
    for i in range(epochs):
        for x,y in zip(X_train,y_train):# for every data point(X_train,y_train)
            dw=gradient_dw(x,y,w,b,alpha,N,eta0)          #compute gradient w.r.to w (call the gradient_dw() function)
            db=gradient_db(x,y,w,b)      #compute gradient w.r.to b (call the gradient_db() function)
            w=w+(alpha*dw)  #update w, b
            b=b+(alpha*db)
        predect=pred(w,b,X_train)
        losst=logloss(y_train,predect)
        train_loss.append(losst)
    return train_loss
```

In [22]:
```
train_loss=train(f_cv_X,f_cv_Y,20,0.01,0.001)
```

In [23]:
```
X=[i for i in range(20)]
Y=train_loss
import matplotlib.pyplot as plt
plt.plot(X,Y)
plt.xlabel('epcho')
plt.legend(['Train_loss','Test_loss'])
plt.title('Epcoh vs losses')
plt.show()
```



In [24]:
```
ftest=decision_function(X_test,clf,SV,0.001)
```

In [25]:
```
from math import exp
def exp_fun(w,b,ftes):
    temp=ftes  + b
    return 1/(exp(-temp))
```

In [26]:
```
temp=[]
for i in ftest:
    temp.append(exp_fun(w,b,i))
```

In [27]:
```
print(temp)
```

```
[0.1990384349125596, 0.13042038667622374, 0.4548684318170676, 0.11002794978555795, 0.022129427427553586, 0.04689348272639252, 0.08789884098149284, 18.576051063634427, 0.03406817979
518741, 3.5341694893532902, 0.11885283286147293, 2.7118709911361925, 0.612858925288496, 0.13255236828954559, 0.4736916150793697, 28.19836823904677, 0.03771499011667371, 0.051060772
424962826, 0.29212513701967496, 0.05696777146597392, 6.892176620433089, 0.1218779788855356, 5.188133390254035, 0.2938303205563454, 0.1148574019159291, 2.0408286084771925, 5.6535444
31621272, 0.4119203343204511, 0.060557436281172924, 2.6226400676346726, 0.11316094242444578, 10.013053940665682, 0.172796554667986, 0.030208614196050393, 0.05200957785305591, 0.256
03054949564397, 0.28408325796502804, 0.09715455445164181, 7.302077405625844, 0.03418550011148117, 0.014629537901782603, 1.458793145778746, 0.10317810228291979, 0.04771389003594695,
0.048337353496522004, 0.16768508116257577, 0.05391680737421425, 0.08238662952488662, 4.2500523420658824, 0.16253228960732133, 0.060306513985778076, 0.09855959305726927, 0.087404284
56946633, 1.486812295109756, 4.908632760046323, 8.372042561855183, 0.02897683091765853, 11.91382393619438, 2.2023823066704833, 0.045491576997814975, 0.1010893696844716, 6.878285863
779243, 0.11198181978330152, 0.0645123269948899, 0.15300878675141286, 0.43575230986362035, 0.03870979327034872, 0.03175311758854435, 0.41256743699311177, 0.037854602593494824, 0.07
88547974994019, 0.7640808135400393, 4.7415126488558945, 0.16423196813854657, 0.04802289779160676, 0.1982325582002952, 0.4259499471349618, 0.21717097286833845, 0.03186058879682192,
0.05647152328000247, 1.7482968459398942, 0.027487320937159714, 0.16035473178523155, 2.2434852577591005, 10.892603735351308, 1.936931114827374, 0.211664518282626, 3.931655322553758,
0.15134608957298254, 1.4512817589325173, 2.267610253702226, 0.036166217874574226, 0.20399073125757908, 0.031228849331205177, 6.207034490628861, 0.6550609260559211, 0.06220204589099
68, 0.2312269680516894, 3.4331179729312864, 2.6395703233486048, 4.898554893586495, 6.942252501463559, 4.41707566240242, 2.2894458366662533, 3.322060826785597, 0.0881059207719933, 6.
583616536330761, 0.27178457882512863, 0.3412970802372931, 0.06207928212457218, 0.04866078752408683, 0.026562394686151774, 0.06108868837067765, 2.1973994596557773, 0.011566140497069
704, 0.05545198033160963, 0.09430691313610974, 0.08568006756483312, 5.123386801884291, 9.031304136721845, 4.688599647381936, 0.4619662511253166, 0.042563099820240397, 0.13297942600
741722, 2.351036061836681, 0.015795386970075067, 0.12030514342553833, 5.566538795242764, 3.681242813053486, 1.4773234188402695, 0.17728917118059273, 0.05742594603494959, 0.01679166
0210992904, 0.016739272274886845, 0.05785406208819822, 1.7804790272277713, 0.048865176435323045, 0.2032599370404507, 0.0902566229332576, 0.023942446304441815, 0.07243076893779221,
9.812273485626074, 0.15665615713324504, 0.04858831028630013, 0.06464872498594147, 0.010715145793508964, 0.04747031647848512, 0.0445266147857559, 3.2611327188006247, 0.0903359099194
136, 5.434781541815629, 8.38706838358231, 0.9757680608124204, 0.2557533448422831, 0.04226508474331299, 0.05963047863595911, 0.04697523225436661, 0.05622234702413362, 0.049030016907
19441, 3.8470469982813222, 0.048416643533521604, 0.9731374937597679, 5.060154245383749, 0.008062183930442796, 0.16533425417653091, 0.04575092982746565, 0.04953702714632729, 0.24863
407990315856, 1.888116624929422, 0.11977417591475954, 0.4422655341105047, 10.358754640063632, 5.25060286644809, 0.022020145346042012, 0.09504355432819438, 0.10767924719152874, 0.06
623696434606338, 0.08615844431563138, 0.2761745595095393, 0.2017007667764851, 0.010324992778535888, 5.593969912198338, 4.156065099587662, 0.18690136498413326, 7.167930299427721, 0.
22763278036723222, 5.15802379497378, 0.013053347840325776, 11.438435311149036, 0.08951609363743215, 0.07907971286732075, 3.2018062288524773, 1.348231105232967, 0.03437735757512829
6, 6.349025926328691, 0.05752123579349908, 0.09219596434710872, 0.09588013755771882, 0.07218901978588, 0.3448953875360029, 0.11646990764314046, 0.09320750104284554, 6.351400195674
592, 1.1933509373233269, 0.21013929194255937, 0.01585078071047978, 5.087377062932173, 4.434144608687535, 0.09608620246859123, 4.6041306519202765, 0.23571589769809437, 0.06493817893
43674, 0.03694427698955341, 19.417723771749216, 0.0630347092780666, 0.07634245944534294, 0.5953620348350686, 3.934208277489238, 0.09139430267886296, 0.1567709871734467, 0.059396252
67069701, 0.07921061680548327, 0.10282636741228332, 0.09429936785727738, 6.307996790688975, 0.4449784871602628, 0.06570525357599859, 0.03389504059698026, 1.5344716603801176, 0.0178
88243344007213, 0.03227381189432286, 0.05626562157448676, 0.2872875446347851, 0.1184379573911561, 0.027252955184095186, 0.10962468211866871, 0.021008227339371053, 0.176141292711513
9, 1.4601117693210242, 0.06578681776549115, 4.427363000504013, 0.14275030064942085, 0.09515460099901064, 4.668384362247437, 0.23132595902378045, 4.996557705450738, 0.37199163063599
094, 0.31989618402301156, 0.17295768520948282, 0.1664672480956766, 1.5777198292438686, 0.42762453981256954, 9.346472795812648, 1.3844197193736614, 0.09197900639234552, 1.0529152291
855155, 3.6415621832232876, 1.1941684072694385, 0.7023958466618504, 4.148070237288259, 0.3971584642305965, 0.15079756804051145, 0.020899508421094504, 0.11516574966405042, 0.1163428
1534272517, 0.17733004417234569, 0.4747480021537325, 0.8832684325282871, 0.1803230861044579, 1.083789966556583, 0.0262121482626064, 0.2258741244440689, 0.3414360830461436, 0.090119
11605959486, 0.102031506438154, 9.083379289879424, 9.623881270921794, 1.9343659809787992, 0.29185285664039784, 9.650727902725956, 0.03498780486661349, 2.175492654603881, 4.53775681
0616766, 0.1638474357682973, 0.07667001403417929, 1.839376623448945, 0.07502140390143362, 0.1976194082631581, 0.03429314996891683, 0.41796794849754554, 0.4899905813906166, 2.266554
992004731, 4.3992485677307345, 0.05814025058267568, 0.0809786882194019, 0.39577265028216974, 6.001631402668052, 0.6322193878229195, 1.394519669343754, 7.78207703629869, 3.338261480
6377284, 0.7262250188025823, 5.6327485426383115, 0.22391535010111746, 5.660521679111207, 0.04725683117788995, 2.962873906800802, 0.10268210430190798, 0.8612129747478611, 0.08698687
912201503, 0.1147780905298551, 0.04835109012626189, 0.12763894948099253, 0.3539040479870342, 0.06561882787676826, 0.005017430961301858, 2.949216300352139, 0.025761204444974282, 0.2
830915198105584, 0.05914662787394915, 0.0551710124839858, 5.759884770253636, 1.497792994698459, 0.0930841791615649, 0.07245695519391454, 0.08045323422074717, 0.04964445953919595,
0.15787955218303867, 0.255671086129585, 0.7885744770230934, 0.011152618868138552, 0.13489758790980363, 8.634452055670554, 0.6771552919299518, 0.3170330897772616, 1.083728556747409
8, 0.026608142699155013, 8.496150363838884, 3.0521989379066623, 0.2889050724666469, 0.1293973051275788, 0.09850457494603478, 6.345582229311573, 0.061713800341293146, 0.315746257082
30986, 0.05614869475061503, 0.023342763894045822, 3.515543615552259, 0.15141846276754656, 0.16035681468400403, 0.8493798532521031, 0.5289787394987598, 0.03915718109492194, 0.037564
783823028046, 0.157493645981296, 0.5431882412744924, 0.3863290256699462, 1.3291745167962756, 3.5861681700705414, 0.35736460795492037, 1.9944738616399642, 0.07274256114412266, 2.567
6866547630337, 0.06203310865076874, 0.05679198650973252, 1.5442913867586403, 0.2704792054330784, 0.15924741008487964, 0.08728207748593761, 2.7000264784864947, 0.11467601341809264,
0.2543848033385362, 5.134599435827591, 1.6371103100175568, 3.7716439288045716, 0.06524273838399276, 0.1563303233553265, 7.758562693046634, 6.531721662368638, 4.298200523759193, 5.7
961931080979525, 4.856735870004874, 0.04766577130330387, 0.10799373196967786, 0.7829316011034736, 0.86237879290336, 0.04605823626528087, 0.1086315541352202, 0.05328176265761017, 0.
15036476825300665, 0.7033613121771664, 0.02080540978115870, 0.04560497101311653, 0.03289722246159091, 0.21791787712055738, 0.3668824233268173, 1.7383070331069979, 0.0454269846729114
04, 0.05412662540186483, 0.02922751455616745, 0.7226280720634205, 0.6198828289838875, 0.11722744247822459, 9.08310847529101, 0.05087959263138582, 0.05782526131944655, 0.10913642119
340393, 0.04823555873884944, 0.6189195301419951, 0.13006747826541254, 3.523018181118128, 0.09899621434290218, 0.13730645785243997, 0.7193005034430731, 0.03241059440066973, 3.497600
1861982775, 0.09707007532379391, 0.21176948957468583, 0.7881066714643303, 0.20497779637996638, 0.1617600555432714, 0.04712187124877776, 0.04746553154061125, 3.634700325590975, 0.07
866931933529031, 4.696447157580087, 0.046450873859351496, 2.3970181143293656, 0.05279310145250239, 0.08209352243356462, 0.13523514162132225, 0.1875970905484572, 3.770457228665049,
0.19343574347067347, 0.0784627627865068, 7.334592179137668, 0.07059124662637101, 0.03292749529835976, 0.11770318496334048, 0.15079808814440707, 0.12708636966039813, 0.0384009334023
7266, 6.435282506101102, 0.19192729024059318, 0.03276531435018142, 0.046369427563768856, 0.06903144950095244, 0.030978978540217004, 0.46727074954355863, 0.015881589535217826, 5.741
648716880407, 0.144093123163907, 0.5295329177929544, 0.18870777616533152, 0.5761288480160721, 0.378152952642545, 2.3374526361735968, 0.6208542899280339, 0.025843032650003858, 0.090
69151343006575, 6.880444903177291, 0.19872668617846703, 2.620229166701349, 0.023738421011331383, 3.7274709228529237, 0.027705857819896804, 0.11661185698910209, 0.05851586286832724,
0.02851324545473148, 0.05665597767893867, 0.0643438719286669, 5.868564586153901, 5.2146935147821445, 0.3313865750678202, 10.564790840895446, 4.744053062771775, 0.02336775565203628
7, 7.318299146675176, 2.5536056144763215, 0.39369588371998593, 8.812968378836063, 1.1554349436547013, 0.8863920308030683, 0.49768852505679556, 58.989193285224744, 0.04488444751985
15, 3.443501992179223, 0.2713217431767199, 6.511066876187797, 3.7079358808243, 0.081915089952, 1.089954556168743, 0.30004521468905304, 0.06383287744725831, 0.10854600132491395, 0.0
6696463385543869, 1.3890912820673533, 0.03772573407755818, 6.828031122525413, 0.2265579265764297, 0.010295652244576577, 0.3724120306531106, 5.950883737452462, 4.461937539084825, 0.
29811178985547426, 0.5162504299490922, 0.16702546690014208, 0.5265091672680223, 0.07738536795746086, 3.5901365479691907, 0.9997995717541159, 14.39063643619138, 0.1614187727216364,
5.789423084730187, 5.564883368247802, 0.07815329907316948, 2.557670560640231, 0.035564604233573605, 0.03235444114206368, 4.9959187426448075, 0.03567548253878319, 2.321913148981933,
0.10702405315138004, 1.4201797987250284, 5.830462694288921, 0.03247162279152282, 5.57375508140797, 0.019595849120630268, 0.0683593280634158, 0.5522549651861299, 0.054981222939947 57
4, 0.1784402690058183, 5.05540640037539, 6.449917313815492, 0.17834410968457823, 0.00807708722290036, 0.7313544776514775, 1.6433835978840894, 0.05621884711290424, 0.046136335598279
97, 0.07192044175098145, 0.14210733839538403, 0.019809613331320402, 4.935559085403752, 18.673313213750546, 8.0316719723952, 7.035889399486422, 0.41152978014155256, 0.06380113265547
076, 0.0979956401916792, 2.826095761244704, 1.0260723100645315, 0.997897300051498, 0.30454819411879225, 3.4148557658187952, 0.06489222365676546, 0.7377488565190753, 0.039019627834 7
12606, 0.08667087579517448, 0.03450946019418353 6, 0.4995820830539336, 1.1312829917820828, 6.575011456064863, 0.1535052261994721, 0.2731774102345919, 3.2112286937149888, 14.62520683
1812578, 0.11770343043556916, 0.05196848311271436, 0.09291575141814247, 0.20580579994831258, 0.2656112639422 7325, 0.09010038887272583, 0.04954720418607011, 0.11605534890068522, 0.0
5501107870219379, 0.05736762742696685, 0.027606970941538533, 0.046723683552731266, 0.0867907268741896, 0.047092542970521484, 0.8770977328325411, 0.013124927212092914, 8.44689176191
0148, 0.11656578706406708, 4.2681592619792035, 0.4018733405735616, 0.06832358115311192, 1.7360610442390239, 0.060538291688257 24, 3.875532605839003, 5.030055013298509, 8.50209912648
541, 0.0981197170984329, 0.016411792030535204, 0.49120615358737246, 0.2937309792464206, 0.13232967529744 04, 0.575670936358185, 0.25318133788406916, 0.014530462495202496, 0.03415647
572005542, 0.034137601520364685, 0.09850859324279011, 7.239792457587727, 0.142790851330063, 0.547374998338233, 0.08004143814412062, 0.07062298407195548, 6.527015267743565, 5.788131
016554897, 6.1708870446115975, 0.6985702020992868, 0.007426972751448979, 0.07078214988525282, 5.416115229233355, 0.08684476532646847, 0.09322492159719088, 0.09568433425503686, 6.09
5945631193126, 0.03202965715982226, 0.05237460041574893, 2.7470515903445754, 0.186687292138801, 0.21185894646335185, 0.038927384549545835, 0.05354199945624711, 0.16164621845356983,
0.07076125983881648, 6.574272568044507, 0.019118857069590312, 0.1305622941938147, 0.07792835089678436, 0.7242478666413343, 3.2326116445083497, 0.03290391015554786, 0.49529474679111
45, 0.12492680645711883, 4.242694474768931, 0.03348237712773211, 2.1494381801231928, 0.11594864650480392, 0.0678776088412228, 0.06526103092171137, 0.2828245149907476, 0.01624939106
294507, 0.04742740637566373, 0.7608311038568986, 0.08966193616927343, 0.07202234164439826, 0.08498833881378824, 0.058113261500442576, 0.11190573651444888, 0.20653436339758724, 6.61
5414482803819, 0.23599550371924796, 1.1122339641584973, 0.05806132516314035, 0.08160840854346599, 0.07419216377156099, 4.882105102591767, 0.10447113358772377, 0.07640318168602699,
0.1430311027032067, 0.016858275730395923, 0.10929002055797468, 2.889200606071526, 0.10324088805868048, 7.943301537374262, 4.690555687814552, 0.19254408926196928, 0.087937904788557
2, 0.11080506677243192, 2.1391086045632726, 2.8975431774275355, 0.0907372184698532, 0.048045094736262194, 0.11690800017093365, 0.06969244606021117, 0.12202737942518538, 0.509355144
5485046, 2.135188781008948, 5.820273241818069, 0.041963332521066835, 0.03867657888761077, 0.5868352194556509, 2.75644967992792, 0.017491360751731366, 0.4311635299977446, 5.12513178
402141, 0.05441944381628596, 0.009812909576716712, 0.05447954202653005 5, 0.18652072302262762, 0.05101742049222104, 4.612190137047505, 0.5722636917752163, 1.42576059132097, 0.092594
05714503594, 0.5678089074328758, 9.950150686617047, 0.8536802770558989, 0.7151867208778695, 0.5030157109930028, 0.06858520546617353, 0.08436054948832823, 9.542122785163382, 0.22586
918604086773, 0.050133317326475, 0.02915049500759649, 3.0373450103376007, 0.05551980354225402, 0.018590381463047114, 0.08597756579503406, 0.10340858923033976, 1.541527063901695, 0.
07943705288848055, 0.07170612727058899, 4.96009372255247, 7.416091850963037, 0.03352574152325435, 0.2380393770492294, 0.43504624072187636, 0.04967852534601726, 1.0130108592701306,
0.21445168472450965, 0.03583874006587704, 0.41219188972257415, 0.5858378167516538, 0.4652386209799323, 0.13259113462807437, 0.38343648448693307, 0.06092767610733608, 3.941044103520
4355, 0.15547709870118662, 0.09745340740130477, 0.08256018431307127, 0.11042412827194313, 0.19523889008989936, 1.0181487834514866, 1.406319350960598, 0.11884622220339112, 3.7016644
647483283, 0.14244360919868482, 2.98908525148879, 0.15056010965080727, 0.030740749848866676, 0.17212693240795476, 0.03189887506992662, 0.8529191595330834, 5.762553011785445, 0.0325
4306391567291, 0.8690849522186105, 0.3329832993256995, 0.0522551916631854, 0.7117072731554286, 0.19456216020327227, 0.17574488272176708, 0.1293730454242682, 1.9719122010957426, 5.3
91232465318214, 1.1318881030530301, 0.5126330495571179, 5.619393941853867, 0.40456159765113076, 0.09801130031071795, 0.19443880688516182, 8.18062675569478, 8.214261452108222, 4.369
815254756828, 2.207219651475896, 0.321930383444948, 0.8905836966931755, 0.07660718031085927, 3.0109793985831734, 0.1865750608500367, 10.089300686868837, 6.227781282078232, 0.373522
4036857591, 0.19991648230826128, 0.010364870545604674, 0.23435954990410018, 3.4142122956403735, 4.052383145851897, 0.23192660996143588, 0.16697999506617928, 0.07928824098172312, 0.
22659490054577996, 0.060521302023278406, 0.12238535406828188, 8.225588667119359, 4.257706338804405, 0.021409472482364834, 5.162507701898714, 4.949818542754839, 0.0759522084825641,
0.47065904070247494, 0.20693138781883944, 0.06084759890331014, 0.6447326974512375, 3.997785748605986, 7.129450262014452, 6.6837382825114116, 0.04174266284177258, 0.0511611593216618
7, 0.0689357831920709, 17.56888970835694, 0.2039463882412837, 0.26236393918292683, 5.316240445068045, 0.12633988799207027, 0.20046847802510623, 0.07827002410650678, 0.2524367003130
988, 0.0400511715992636, 3.7432342206033336, 0.27730822969665997, 0.05063004846330218, 0.13775312950825813, 0.032406021713872225, 8.276175931799942, 5.913639110403415, 0.0437227946
5371328, 0.8761520380720909, 3.594209957497673, 9.235370387075754, 0.0854994658115079, 1.1608307252427916, 0.08673793437221763, 0.052354287245743876, 5.835163010374287, 9.334436508
91878, 0.15962592302015183, 0.4664431705331748, 0.6539906137276831, 0.48460963459505146, 2.9079060181776963, 0.09633909771710139, 0.3352873468053147, 0.008365120905182313, 0.012924
945302854712, 0.1969976764699557, 2.804085557386776, 3.0415178139458376, 0.187720286540408, 0.040661728417409566, 0.09053135119587365, 0.022355715624519473, 0.02421067785792369, 0.
1463356682854465, 0.03696206743062447, 0.21946988063065037, 0.02280179963645574, 1.954384971551982, 0.29711621121769893, 0.08046723071691952, 0.1632875622559195, 1.781974216465093
8, 8.145362840433643, 0.2616670045556029, 0.07863611948903079, 4.855791212431799, 0.18618203890360804, 0.04064853783366766, 6.746303069358536, 1.3493821555535144, 0.019540669078935
51, 1.5478265464918157, 0.07351506620352706, 0.12945968966389135, 0.08024053606123659, 4.350404666021362, 0.5024214191092242, 4.425925743100413, 3.77959942920096, 0.111395050075677
24, 0.0729400523194947, 0.2372393425042456, 0.05135401861776242, 0.034241852779541934, 0.1884207101918736, 0.2221621094245538, 0.03292619614952058, 0.09734791407821064, 0.233366328
87429637, 0.1104314452587805, 0.4089815222647348, 4.52697675431054, 0.9788991067135387, 0.14779045821718928, 6.32813781487662, 0.05017759507576518, 0.09990642743222962, 3.268379128
3328167, 0.06909175251107366, 0.6397716936996393, 0.020925588068341466, 0.058284941286327996, 1.4752016371712242, 0.051604225100332725, 0.028255579589957388, 0.06642516963160164,
0.02349431907474709, 0.01459830113644496, 1.0962439655014908, 0.1609458611878162, 0.5661883407947339, 5.606245577577955, 0.26122718095260045, 6.110766834935329, 0.0306770537139288
7, 0.08809152812483087, 1.98373823738326, 0.11216620728472798, 0.18943372852854373, 0.24345856136477936, 0.1506906454215694, 0.04689517798973253, 1.8327417808062771, 0.027635295159
51555, 0.5146889113935058, 0.12703444256419666, 5.44776598498044, 0.05464438229687879, 0.8766906226203449, 0.6865528200543768, 5.751323636583924, 0.031368658626617925, 0.1349830526
4163246, 4.349815877385736, 0.36553935934097326, 0.3350012037714977, 11.056558357454671, 0.6499637366669242, 5.879295770033022, 0.23546882899358473, 0.24072030987759832, 2.92703414
3789611, 0.09545051571089098, 0.7564216954241324, 0.39781851702407006, 0.067899800756327, 0.1274502398744705, 0.2078872069804697, 0.048670905738357084, 0.17158015725345077, 5.44500
8563626455, 0.10562380035733726, 0.24562566045269868, 0.177211912775613, 0.05327383150857194, 0.45965658628945694, 1.212638433844639, 0.09767804536091579, 0.03025506523374657, 0.11
450955474657619, 0.04993628171446166, 0.02035253936555691, 0.5038777113979475, 0.03549153464379482, 0.08686267046273999, 0.14426466027532406, 0.01894473357977008, 0.029454417317893
822, 0.3252882885648389, 0.10880670660163282, 0.050072432810283854, 6.447841925212524, 0.04919532732701011, 34.421144198468475, 3.9668710423442968, 0.2007857440199108, 0.0513388509
6876533, 0.4854890697393565, 0.47207125261239996, 0.1720128363010237, 0.015782610201209184, 0.022903966319322004, 0.14135976060254257, 0.16808408808535558, 0.017985168338778414, 0.
04762942914182645, 0.13398650961127304, 0.8832176607120565, 0.05868054962844335, 0.061135176776227186, 0.016936743147238 79, 0.0644852070928685, 0.5399799901233356, 0.72170200322653
04, 0.012000023906225512, 1.582966717716, 3.647624626056127, 0.05080130161083468, 0.21118233689831573, 0.053640892729491615, 0.11427386612372831, 0.399487606335277, 3.8477060183198
404, 0.7385606551456064]
```

In [ ]: