Homework 1

1. (a) not a metric, first it violates that $d(x, y) \geq 0$, *if x=0, y=1*,then $d(x, y)$ =$0 - 1$ = -1 <0; secondly, it doesn't satisfy symmetry since $d(x, y)$ =$x - y \neq y - x$ =$d(y, x), if\ x \neq y$.

(b) yes it is, it satisfies all four properties.

Firstly, it is more and equal to 0, since the # of difference is non-negative;

Secondly, $d(x, y)$ =0 if and only if x=y;

Thirdly, d(x, y) = d(y, x);

Lastly, it satisfies triangle inequality, $d(x, z)$ can be seen as the transfer from x to z; $d(x, z) + d(x, y)$ can be seen as the transfer from x to y then to z, so it must be larger or equal to direct transform.

(c) No, it's not

It doesn't satisfy the triangle inequality.

When m=1, let x=4, y=2, z=0

$$d(x, z) = (x - z)^{2} = 16$$

$$> d(x, y) + d(y, z) = (x - y)^{2} + (y - z)^{2} = 4 + 4 = 8$$

Similarly, when m>1, let each entry of x be 4, each entry of x be 2, each entry of z be 0, then

$$d(x, z) = \sum_{i=1}^{m} (xi\ -\ zi)^{2} = 16m$$

$$d(x,y) = \sum_{i=1}^{m} (xi - yi)^{2} = 4m$$

$$d(y,z) = \sum_{i=1}^{m} (yi - zi)^{2} = 4m$$

Clearly, it doesn't satisfy the triangle inequality.

2. (a) clearly, according to triangle inequality, we have

$$f(\theta x + (1-\theta)y) \le f(\theta x) + f((1-\theta)y)$$

then we consider its homogeneous property,

$$f(\theta x) + f\big((1-\theta)y\big) = |\theta| f(x) + |(1-\theta)| f(y)$$

We know that θ is between 0 and 1, so

$$|\theta| f(x) + |(1-\theta)| f(y) = \theta f(x) + (1-\theta) f(y)$$

Therefore we have

$$f(\theta x + (1-\theta)y) \le \theta f(x) + (1-\theta) f(y)$$

So every norm is a convex function.

(b) no it's not,

For every t≠0, $f\big(t(1,0,2,0,0)\big) = f\big((t,0,2t,0,0)\big) = 2 \neq 2\mathrm{t} = |\mathrm{t}| f\big((1,0,2,0,0)\big)$, hence it doesn't fit the third property.

(c) Yes

firstly, according to the first two property of norm, it satisfies the first two property of metric (non-negativity and $d(x,y) = x - y \neq y -$

$x = d(y,x), if\ x \neq y$).

Secondly,

$$d(x,y) = \left||x-y|\right| = \left||-1(y-x)|\right|$$
$$= |-1|\left||(y-x)|\right| = \left||y-x|\right| = d(y,x),$$

hence it fits symmetry.

Lastly,

$$d(x,y) = \left||x-y|\right| = \left||x-z+z-y|\right|$$
$$\leq \left||x-z|\right| + \left||z-y|\right| = d(x,z) + d(z,y),$$

Hence it satisfies triangle inequality.

Above all, it satisfies all four property of metric, hence it's a metric.

3. (a) $TVD(p,q) = \frac{3}{8}$;

The set S could be {1},{1,3},{2,4},{2,3,4}, all of them will give the answer 3/8.

(b) $\left||p-q|\right|_1 = \left|\frac{1}{2}-\frac{1}{8}\right| + \left|\frac{1}{4}-\frac{1}{2}\right| + \left|\frac{1}{8}-\frac{1}{8}\right| + \left|\frac{1}{8}-\frac{1}{4}\right| = \frac{3}{4}$

(c) based on (a), we can conclude that the S could be $\mathrm{S1} = \{\mathrm{x}|\ \mathrm{p(x)} \geq \mathrm{q(x)}\}$ or $\mathrm{S2} = \{\mathrm{x}|\ \mathrm{p(x)} < \mathrm{q(x)}\}$; and we can say that

$$\Omega = \mathrm{S1} \cup S2\ \mathrm{s.t.}\ \mathrm{S1} \cap \mathrm{S2} = \varnothing$$

Therefore,

$$\left||p-q|\right|_1 = \sum_{x\in\Omega} |p(x) - q(x)|$$

$$= \sum_{x \in S1} |p(x) - q(x)| + \sum_{x \in S2} |p(x) - q(x)|$$

$$= \sum_{x \in S1} \left(p(x) - q(x)\right) + \sum_{x \in S2} \left(q(x) - p(x)\right)$$

$$= TVD(p,q) + TVD(p,q) = 2TVD(p,q)$$

4. (a)

X	*Prob*
1	1/2
2	1/4
3	1/8
4	1/8

Y	*Prob*
1	1/4
2	1/8
3	1/2
4	1/8

$$Pr(X \neq Y) = \frac{3}{8}$$

(b) $TVD(p,q) = \frac{3}{8}$ according to original definition, and the set can be {1,2} or {3} or {1,2,4} or {3,4}.

5. The best strategy for moving mass should be q move 1/8 from (0.5,0) to (0,0) and 1/8 to (0,1); and q move 1/8 from (0.5,1) to (1,1) and 1/8 to (0,1).

Such distribution is as follow :

(X, Y)	*Prob*
(0,0), (0,0)	1/8
(0,0), (0.5,0)	1/8
(1,0), (0.5,0)	1/8
(1,0), (1,0)	1/8
(0,1), (0,1)	1/8
(0,1), (0.5,1)	1/8
(1,1), (0.5,1)	1/8
(1,1), (1,1)	1/8

Then $W_1(p,q) = \frac{1}{4}$

6.

In [8]:
```python
import numpy as np
import collections
import math
```

In [31]:
```python
import matplotlib.pyplot as plt
```

Q6

(a)

In [33]:
```python
#get points with #=n, dimension=d
def getNPoints(d,n):
    aList=[]
    x=np.random.normal(0,1,size=(n,d))
    for xi in x:
        xi/=np.linalg.norm(xi)
        aList.append(xi)
    return aList
```

In [57]:
```python
#the function for displaying the histOgram
def showHist(x_value,n,d):
    plt.hist(x_value,bins=20,range=[0,2])

    plt.title("the Hist of d={},n={}".format(n,d))
    plt.xlabel("inter points distance")
    plt.ylabel("number of points")

    plt.show()
```

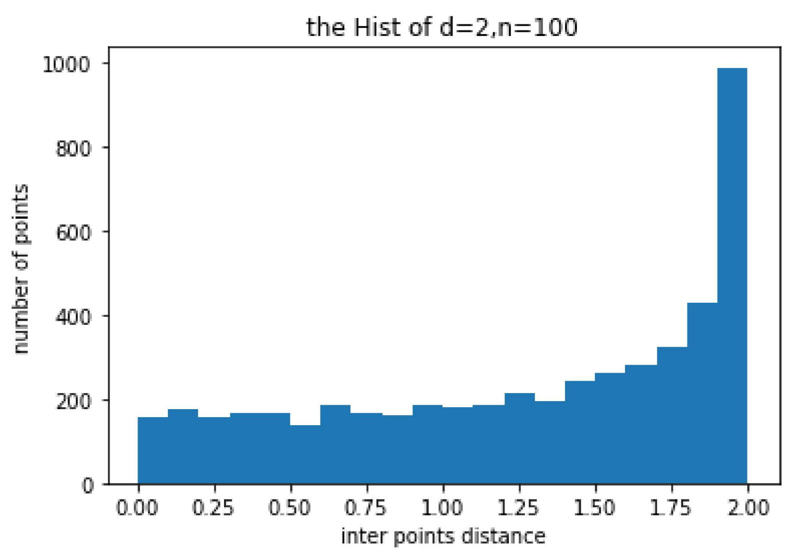
In [58]:
```python
#computer interpoint distance
def interdistance(pointList):
    n=len(pointList) #it should be 100 based on question but use len() to be general
    distanceList=[]
    for i in range(n-1):
        for j in range(i+1,n):
            distanceList.append(np.linalg.norm(pointList[i]-pointList[j]))

    return distanceList
```

experiment with d = 2, 5, 10, 20, 100 and n = 100

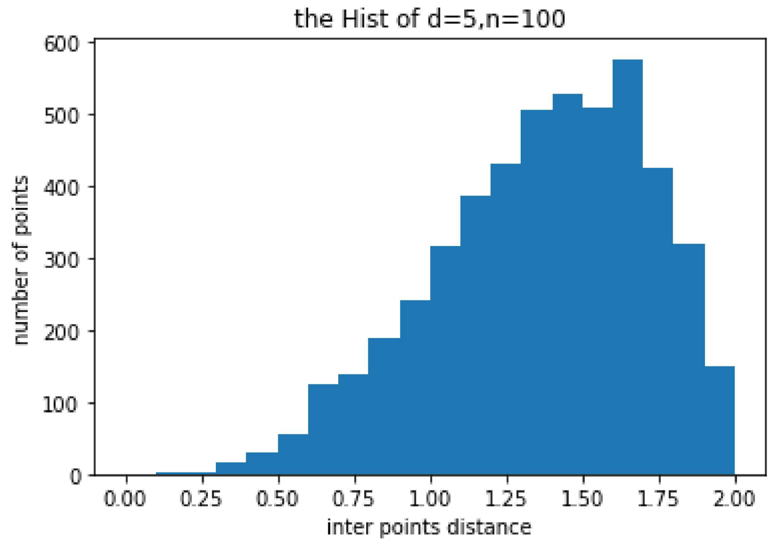
d = 2, n = 100

In [59]:
```python
pointList=getNPoints(2,100)
distanceList=interdistance(pointList)
showHist(distanceList,2,100)
```



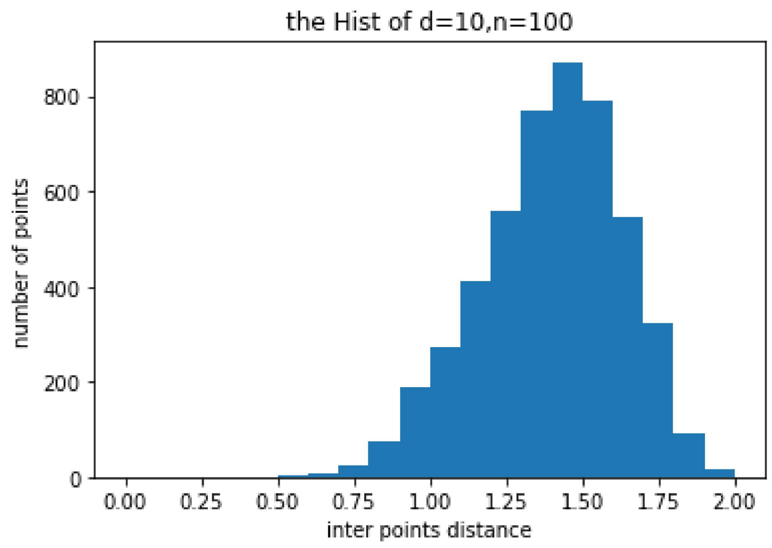
d = 5, n = 100

In [60]:
```
pointList=getNPoints(5,100)
distanceList=interdistance(pointList)
showHist(distanceList,5,100)
```



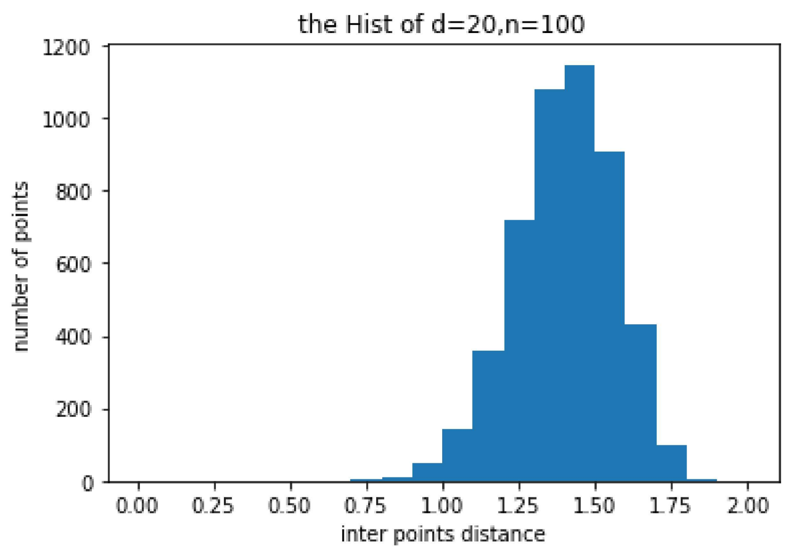
d=10,n=100

In [61]:
```
pointList=getNPoints(10,100)
distanceList=interdistance(pointList)
showHist(distanceList,10,100)
```



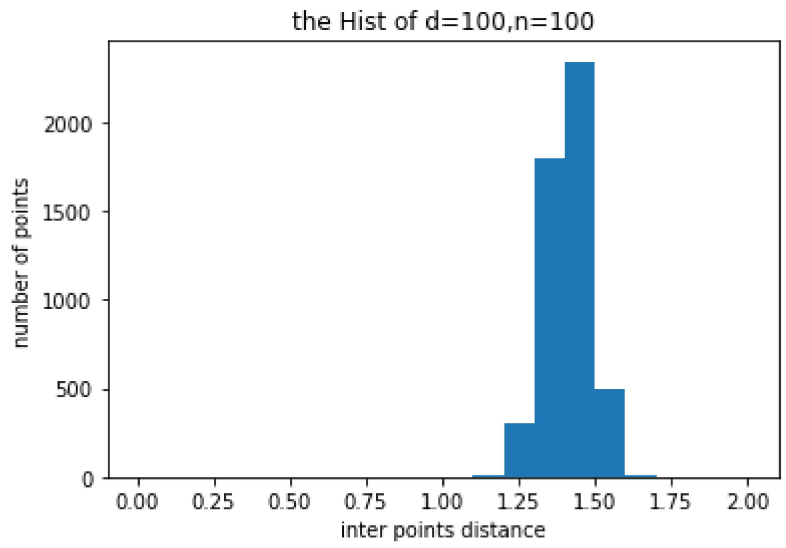
d=20,n=100

In [62]:
```
pointList=getNPoints(20,100)
distanceList=interdistance(pointList)
showHist(distanceList,20,100)
```



d=100,n=100

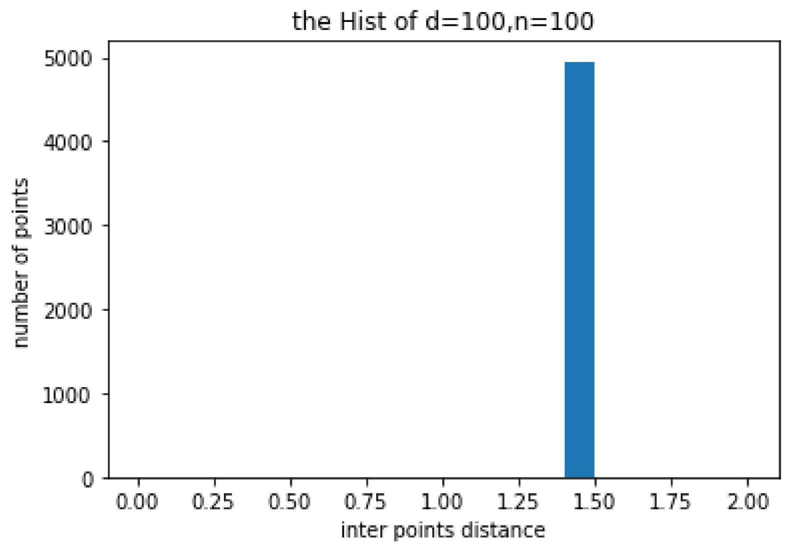
In [63]:
```python
pointList=getNPoints(100,100)
distanceList=interdistance(pointList)
showHist(distanceList,100,100)
```



(b)

Since the n grows the concentration will be more obvious So to guess a approximate answer, I did a experiment with huge dimension first

In [68]:
```python
pointList=getNPoints(100000,100)
distanceList=interdistance(pointList)
showHist(distanceList,100000,100)
app=sum(distanceList)/len(distanceList)
app
```



Out[68]: 1.4142186080637633

According to this approximate value 1.4142186080637633, I hence guess the value is sqrt(2).

(c)

In [78]:
```python
#to use the function we alreay built
#instead of generate one sample at each step, generate them (10000) at once
pointList=getNPoints(1000,10000)
#then the xi are ith entry(pointList[i-1]) of pointList
```

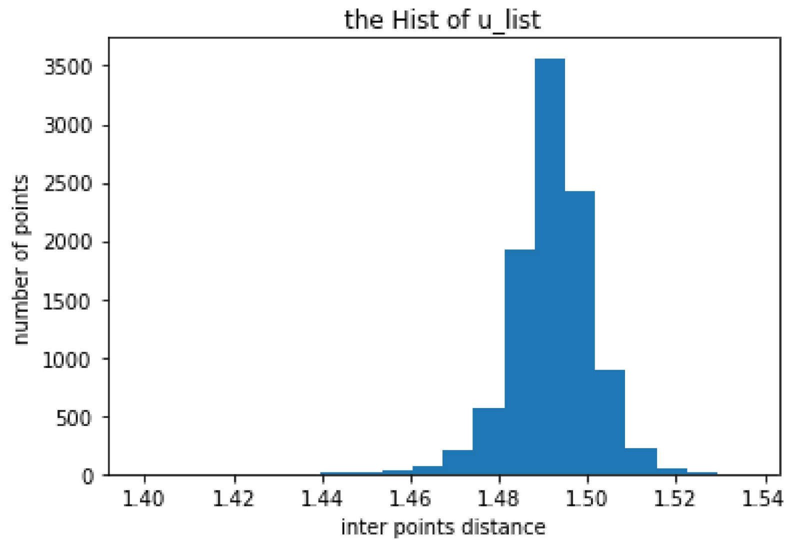
In [79]:
```python
u_list,s_list=[],[]
for i in range (1,10000):
    u,s=0,2
    for j in range(0,i):
        tmp=np.linalg.norm(pointList[i]-pointList[j])
        u=max(u,tmp)
        s=min(s,tmp)
    u_list.append(u)
    s_list.append(s)
```

histgram for two lists each

In [102]:
```
plt.hist(u_list,bins=20)

plt.title("the Hist of u_list")
plt.xlabel("inter points distance")
plt.ylabel("number of points")
plt.show
```

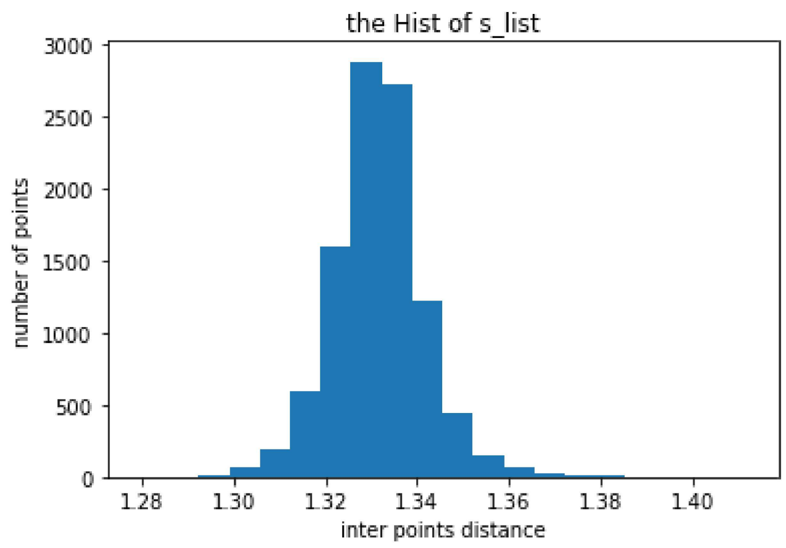
Out[102]: <function matplotlib.pyplot.show(close=None, block=None)>



In [103]:
```
plt.hist(s_list,bins=20)

plt.title("the Hist of s_list")
plt.xlabel("inter points distance")
plt.ylabel("number of points")
plt.show
```

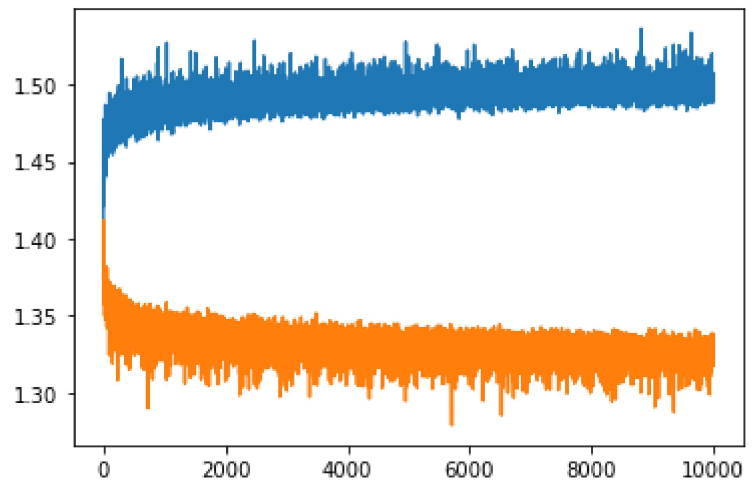
Out[103]: <function matplotlib.pyplot.show(close=None, block=None)>



show them in one plot

In [111]:
```
x=[i for i in range(2,10001)]
plt.plot(x,u_list)
plt.plot(x,s_list)
plt.show
```

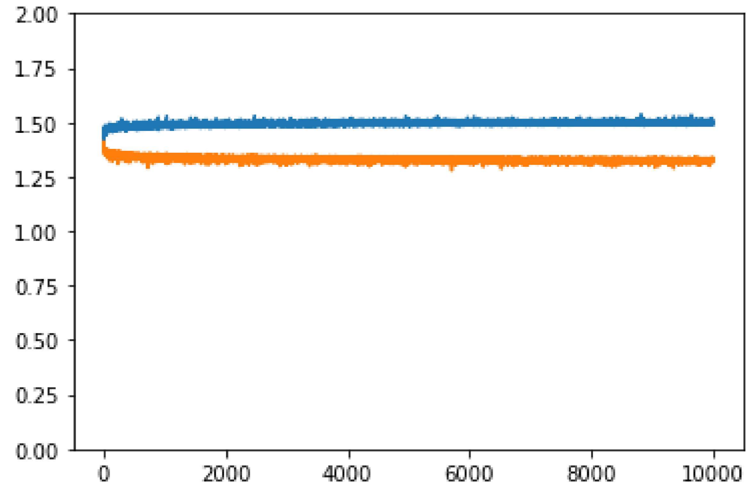
Out[111]: <function matplotlib.pyplot.show(close=None, block=None)>



set y-axis range to [0,2]

In [114]:
```
#set the range
x=[i for i in range(2,10001)]
plt.plot(x,u_list)
plt.plot(x,s_list)
plt.ylim(0, 2)
plt.show
```

Out[114]:
```
<function matplotlib.pyplot.show(close=None, block=None)>
```



Q7

In [115]:
```
#import dataset as the way Prof. mention in piazza
import torchvision.datasets as datasets
train_set=datasets.MNIST(root='./data',train=True,download=True,transform=None)
test_set=datasets.MNIST(root='./data',train=False,download=True,transform=None)
```

```
Downloading http://yann.lecun.com/exdb/mnist/train-images-idx3-ubyte.gz (http://yann.lecun.com/exdb/mnist/train-images-idx3-ubyte.g
z)
Downloading http://yann.lecun.com/exdb/mnist/train-images-idx3-ubyte.gz (http://yann.lecun.com/exdb/mnist/train-images-idx3-ubyte.g
z) to ./data\MNIST\raw\train-images-idx3-ubyte.gz

100%|██████████| 9912422/9912422 [00:01<00:00, 9158965.02it/s]

Extracting ./data\MNIST\raw\train-images-idx3-ubyte.gz to ./data\MNIST\raw

100%|██████████| 28881/28881 [00:00<?, ?it/s]
  0%|          | 0/1648877 [00:00<?, ?it/s]


Downloading http://yann.lecun.com/exdb/mnist/train-labels-idx1-ubyte.gz (http://yann.lecun.com/exdb/mnist/train-labels-idx1-ubyte.g
z)
Downloading http://yann.lecun.com/exdb/mnist/train-labels-idx1-ubyte.gz (http://yann.lecun.com/exdb/mnist/train-labels-idx1-ubyte.g
z) to ./data\MNIST\raw\train-labels-idx1-ubyte.gz
Extracting ./data\MNIST\raw\train-labels-idx1-ubyte.gz to ./data\MNIST\raw

Downloading http://yann.lecun.com/exdb/mnist/t10k-images-idx3-ubyte.gz (http://yann.lecun.com/exdb/mnist/t10k-images-idx3-ubyte.gz)
Downloading http://yann.lecun.com/exdb/mnist/t10k-images-idx3-ubyte.gz (http://yann.lecun.com/exdb/mnist/t10k-images-idx3-ubyte.gz)
to ./data\MNIST\raw\t10k-images-idx3-ubyte.gz

100%|██████████| 1648877/1648877 [00:00<00:00, 9846731.70it/s]

Extracting ./data\MNIST\raw\t10k-images-idx3-ubyte.gz to ./data\MNIST\raw

Downloading http://yann.lecun.com/exdb/mnist/t10k-labels-idx1-ubyte.gz (http://yann.lecun.com/exdb/mnist/t10k-labels-idx1-ubyte.gz)
Downloading http://yann.lecun.com/exdb/mnist/t10k-labels-idx1-ubyte.gz (http://yann.lecun.com/exdb/mnist/t10k-labels-idx1-ubyte.gz)
to ./data\MNIST\raw\t10k-labels-idx1-ubyte.gz

100%|██████████| 4542/4542 [00:00<?, ?it/s]

Extracting ./data\MNIST\raw\t10k-labels-idx1-ubyte.gz to ./data\MNIST\raw
```

In [117]:
```
train_set[0]
```

Out[117]:
```
(<PIL.Image.Image image mode=L size=28x28 at 0x193111A9040>, 5)
```

import neccessary package for the question

In [119]:
```
from sklearn.neighbors import KDTree
```

(a)

For dimensions d = 5, 10, 15, 20, 25, 30, 35, 40, 45, 50

In [140]:

```
aveList=[]
for d in range(5,55,5):
    #d is for demension
    trainList=getNPoints(d,60000)
    testList=getNPoints(d,100)
    tree=KDTree(trainList,leaf_size=2)
    tree.query(testList)
    aveList.append(tree.get_n_calls()/100)
    tree.reset_n_calls()
aveList
```

Out[140]:

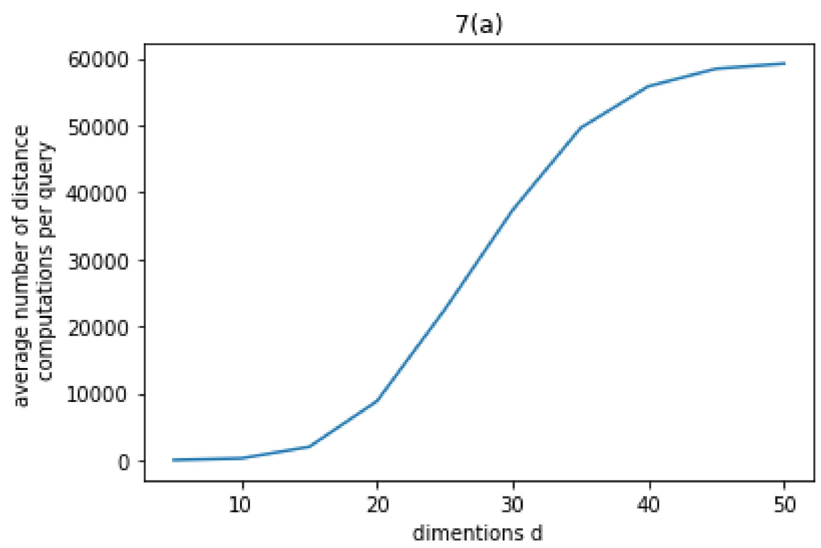
```
[16.58,
 274.48,
 1991.23,
 8833.11,
 22506.62,
 37335.77,
 49577.88,
 55796.16,
 58423.39,
 59179.83]
```

In [144]:

```
x=[i for i in range(5,55,5)]
plt.plot(x,aveList)
plt.xlabel("dimentions d")
plt.ylabel("average number of distance\n computations per query")
plt.title("7(a)")
plt.show
```

Out[144]:

```
<function matplotlib.pyplot.show(close=None, block=None)>
```



(b)

In [145]:

```
print(len(train_set))
print(len(test_set))
```

```
60000
10000
```

In [151]:

```
#the dataset has already downloaded
#select 60000 for training set and 100 for testset
#as the len is shown above, only need to select 100 for testset
test_Set=[]
for i in range(100):
    test_Set.append(test_set[i])
```

convert the dataset, transefer the image to vector

In [168]:

```
trainSet=[]
testSet=[]
for sample in train_set:
    trainSet.append((np.reshape(sample[0], (1, -1)))[0])
for sample in test_Set:
    testSet.append((np.reshape(sample[0], (1, -1)))[0])
```

In [170]:

```
tree=KDTree(trainSet,leaf_size=2)
tree.query(testSet)
ave=(tree.get_n_calls()/100)
tree.reset_n_calls()
```

In [171]:
```
ave
```
Out[171]: 29542.31

(3)

to make sure the above anwer is correct, I tried two more test set with different selection

In [173]:
```
test_set2=[]
for i in range(0,1010,10):
    test_set2.append((np.reshape(test_set[i][0], (1, -1)))[0])
```
In [175]:
```
tree.query(test_set2)
ave2=(tree.get_n_calls()/100)
tree.reset_n_calls()
ave2
```
Out[175]: 32464.55

In [176]:
```
test_set3=[]
for i in range(6,1016,10):
    test_set3.append((np.reshape(test_set[i][0], (1, -1)))[0])

tree.query(test_set3)
ave3=(tree.get_n_calls()/100)
tree.reset_n_calls()
ave3
```
Out[176]: 29690.38

it seems close to answer in question(b) that is to say the answer seem to be around 30000

now we print the answer in (a)

In [177]:
```
print(aveList)
```
```
[16.58, 274.48, 1991.23, 8833.11, 22506.62, 37335.77, 49577.88, 55796.16, 58423.39, 59179.83]
```

the answer in question (b) is between 22506.62 and 37335.77 which are given by dimension 25 and 30

therefore, my rough answer is 27 (to be honest, I think 26,27,28,29 are reasonable).