|  |  |  |
| --- | --- | --- |
| 学号： | 20009200527 |  |
| 姓名： | 杨玉甫 |

**《大数据安全与隐私》课程**

**实验报告**

|  |  |
| --- | --- |
| **学号：** | 20009200527 |
| **姓名：** | 杨玉甫 |
| **指导教师：** | 彭延国 |
| **日期：** |  |

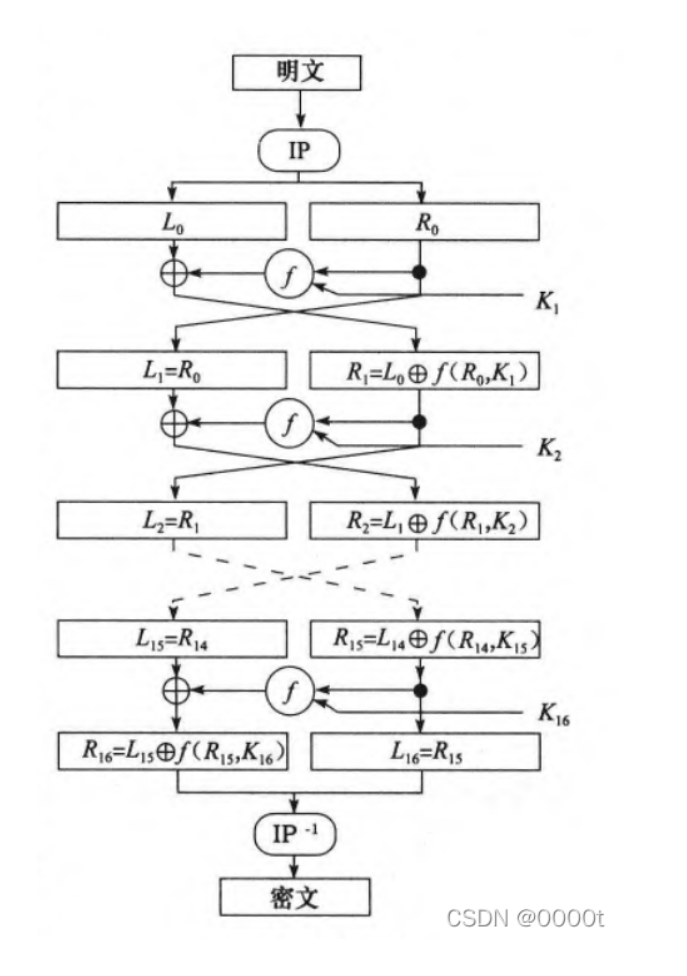
**实验报告撰写要求**

1. 不得抄袭、复制，一经发现取消课程实验成绩；
2. 正文小四字号，固定行距20磅，中文宋体，英文Times New Roman；
3. 每次实验起新页；
4. 实验报告电子档命名规则：“《大数据安全与隐私》实验报告-学号-姓名.doc/.docx/.pdf”；
5. A4纸单面打印，一份；
6. 实验报告作为实验验收的补充，仅提交实验报告未进行线上/线下验收的没有实验部分成绩。
7. 提交时间：2023年6月1日后。具体时间待全部验收后，通过学在西电和微信群通知。

**实验一：DES算法实现不定长度字符串加密解密**

1. **实验原理：**

**Des算法流程图：**



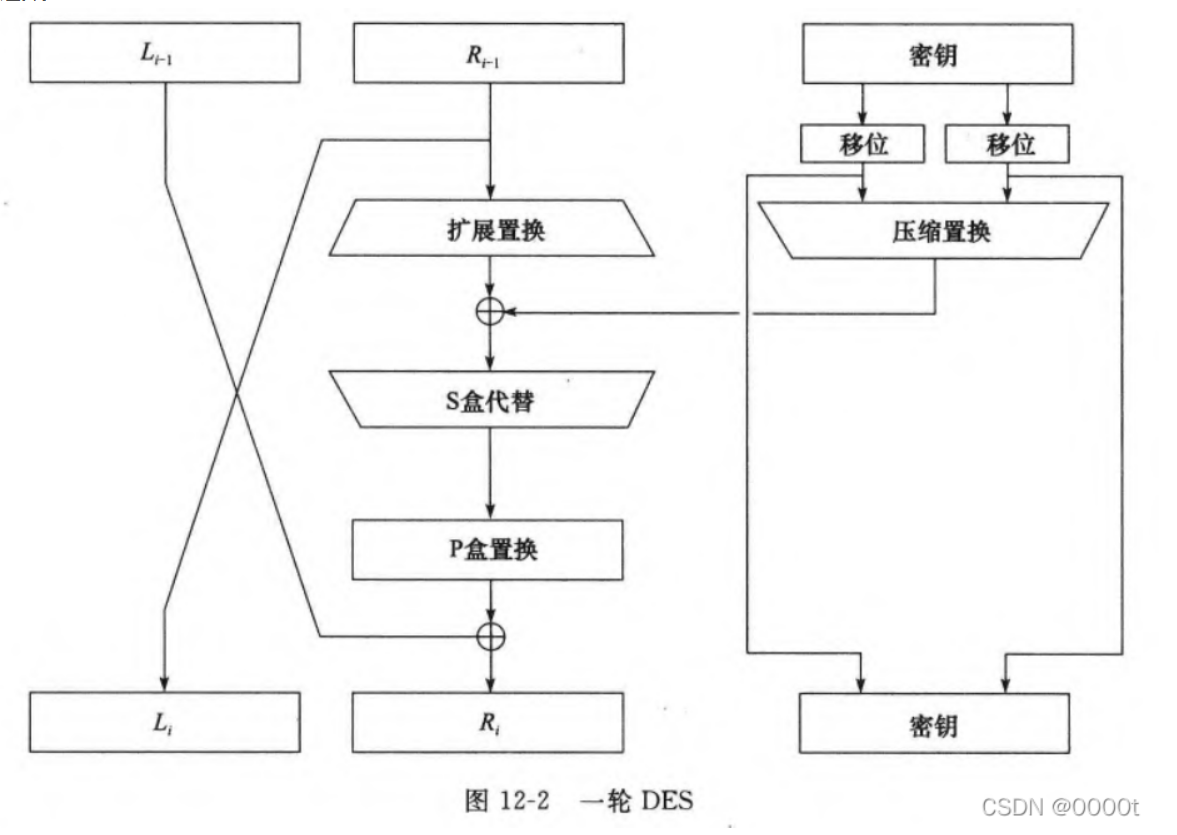
通过初始置换，将明文分成左半部分和右半部分，各32位长

进行16轮完全相同的运算，这些运算称为函数f，在运算过程中数据与密钥结合。

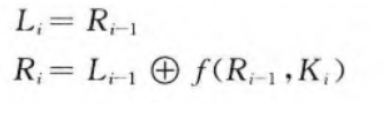
经过十六轮后，左、右半部分合在一起经过一个末置换（初始置换的逆置换）

**一轮DES：**

在每一轮中，密钥位移位，然后再从密钥的56位中选出48位，通过一个扩展置换将数据的右半部位扩展成48位，并通过一个异或运算与48位密钥结合，通过8个S盒将这个48位替代成新的32位数据，再将其置换一次。这4步运算构成了函数f。然后通过一个异或运算，函数f的输出与左半部分结合，其结果即称为新的有右半部分，原来的右半部分成为新的左半部分。将该运算重复16次，便实现了DES的16轮运算。



假设Bi 是第i次迭代的结果，Li和Ri是Bi的左半部分和右半部分，Ki是第i轮的48位密钥，且f是实现代替、置换及密钥异或等运算的函数，那么每一轮就是：



**初始置换：**

该表从左向右，从上向下读。例如，初始置换将明文的第58位换到第一位的位置，把第50位换到第二位的位置，等等。

**密钥置换**

由于不考虑每个字节的第8位，第八位可以作为奇偶校验以确保密钥不发生错误。所以DES的密钥由64位减至56位。密钥置换如下图所示：

在DES的每一轮中，从56位密钥中产生不同的48位子密钥，这些子密钥的Ki由下面的方式确定

1.首先，56位密钥被分成两部分，每部分28位。然后，根据轮数，这两部分分别循环左移1位或2位。

2.移动后，就从56位中选出48位。因为这个运算不仅置换了每位的顺序，同时也选择了子密钥，因而成为压缩置换。该运算提供了一组48位的集，如图所示，处在第33位的那一位在输出时移到了第35位的位置，而处在第18位的那一位被略去了。

3.因为有移位运算，所以在每一个子密钥中使用了不同的密钥子集的位。虽然不是所有的位在子密钥中使用的次数均相同，但是在16个子密钥中，每一位大约使用了其中14个子密钥（，，，啥意思，，，）。

**扩展置换**

将数据的右半部分Ri从32位扩展到48位。

这个运算改变了位的次序并且重复了某些位，成为扩展变换。

该运算实现了两个目的：

产生了与密钥同长度的数据以进行异或运算；

产生了更长的结果使得能在替代运算时进行压缩

在密码学上主要是为了尽可能快的使密文的每一位依赖明文和密钥的每一位。尽管输出分组大于输入分组，但是每一个输入分组产生唯一的输出分组。

显示了扩展置换，有时它也叫做E盒。对每个4位输入分组，第1位和第4位分别表示输出分组中的两位，而第2位和第3位分别表示输出分组中的一位。例如，处在输入分组中的第3位的位置位移到输出分组中第4位的位置，而输入分组中的第21位的位置移到了输出分组的第30位和第32位的位置。

**S盒代替**

压缩后的密钥与扩展分组异或以后，将48位的结果送入进行代替运算。代替由8个代替盒子或S盒完成。每一个S盒都有6位输入，4位输出，且这8个S盒是不同的。48位的输入被分为8个6位的分组，每个分组对应一个S盒代替操作：分组1由S盒1操作，分组2由S盒2操作等。

每个S盒是一个4行，16列的表。盒中的每一项都是一个4位的数。S盒的6个位输入确定了其对应的输出是在哪一行哪一列。

输入位以一种非常特殊的方式确定了S盒中的项。假定将S盒的6位输入标记为b1, b2, b3, b4, b5, b6。b1和b6组合构成了一个2位的数，从0~3，它对应于表中的一行。从b2 ~ b5,构成了一个4位的数，从0 ~ 15，对应于表中的一列。

例如：假设第6个S盒的输入是110011.第一位和最后一位组合形成了11，它对应着第六个S盒的第三行。中间的4位组合在一起形成了1001，它对应着同一个S盒的第九列。S盒6的第三行第九列的数是14，则值1110就代替成了110011.

S盒是非线性的，它比DES的其它任何一步都提供了更好的安全性。

这个代替过程的结果是8个4位的分组，它们重新组合在一起形成了一个32位的分组。这个分组将进行下一步：P盒置换。

**P盒置换**

S盒代替运算后的３２位输出依照P盒进行置换。该置换把每个输入位映射到输出位，任一位不能映射两次，也不能被略去，这个置换叫做直接置换。

例如，第21位移到了第４位，同时第４位移到了第31位。

最后，将P盒置换的结果与最初的64位分组的左半部分异或，然后左，右半部分交换，接着开始另一轮。

末置换

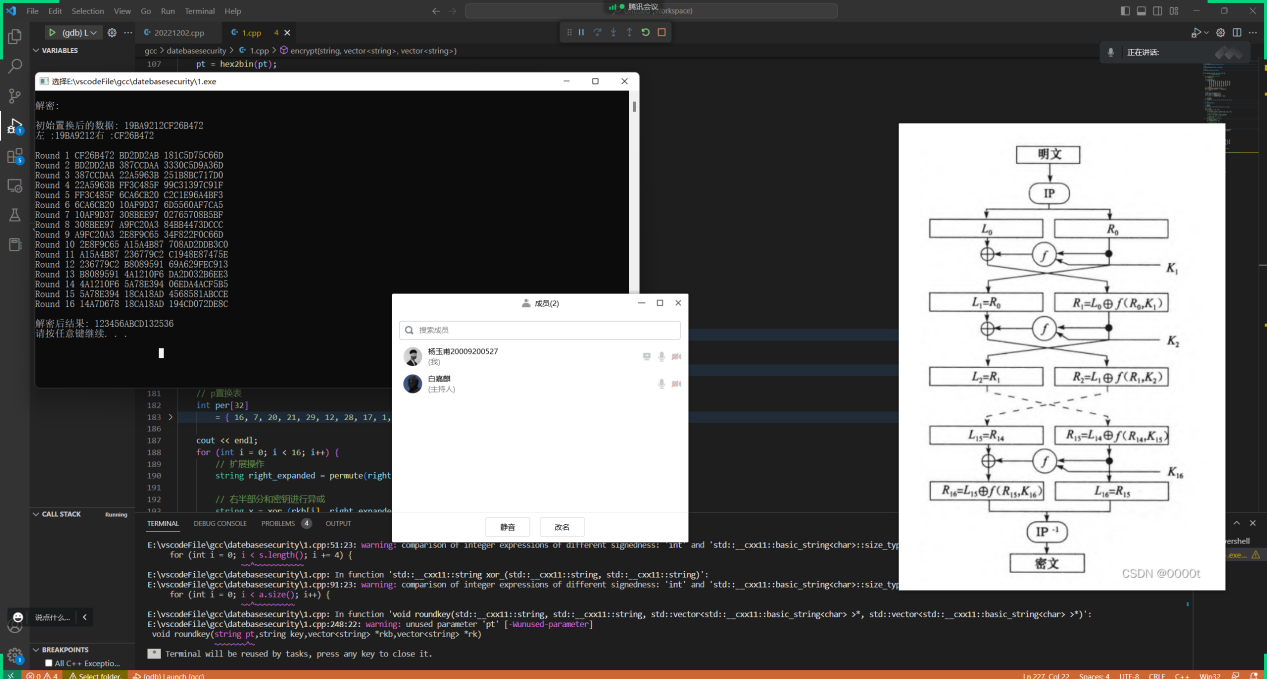
是初始置换的逆过程。注意DES在最后一轮后，左半部分和右半部分并未交换，而是将R16与L16并在一起形成一个分组作为末置换的输入。

**DES的解密**

在经过所有的代替，置换，异或和循环移动之后，你或许认为解密算法和加密算法完全不同，并且也像加密算法一样有很强的混乱效果。恰恰相反，经过精心选择各种运算，获得了这样一个非常有用的性质：加密和解密可以使用相同的算法。

DES使得用相同的函数来加密或解密每个分组成为可能。两者位移不同的是密钥的次序相反。这就是说各轮的加密密钥如果是K１，K２，　K３，　K４，，，K１６，那么解密密钥就是K16, K 15, K14,…,K1。为各轮产生密钥的算法也是循环的，密钥向右移动，每次移动的个数为0，1，2，2，2，2，2，2，1，2，2，2，2，2，2，1。

1. **实验结果：**



1. **实验总结与心得：**

通过本此实验，我更加清楚的了解了des加密和解密的全过程，也锻炼了我的代码编写能力，受益匪浅。

**实验二：空间数据安全（近似）近邻检索**

1. **实验原理：**

**（1）实验要求**

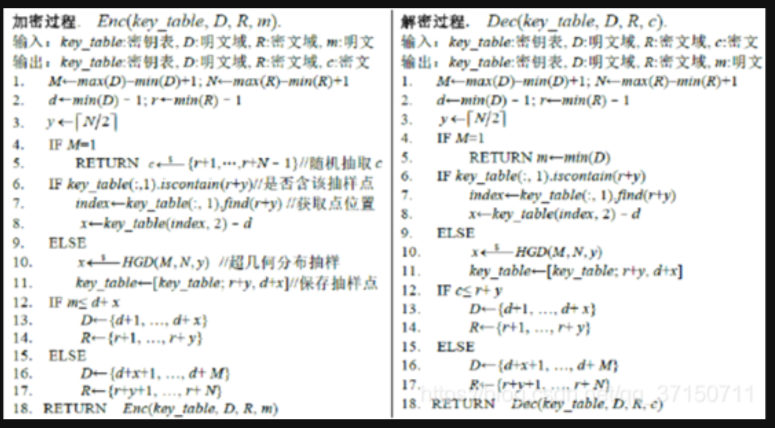
1.二维乱序数据，NE数据集；

2.采用任一种保序加密方法加密；

3.构建恰当的索引结构，如KD树等；

4.可在二维数据上进行安全（近似）近邻查询。

**（2）poe保序加密算法**

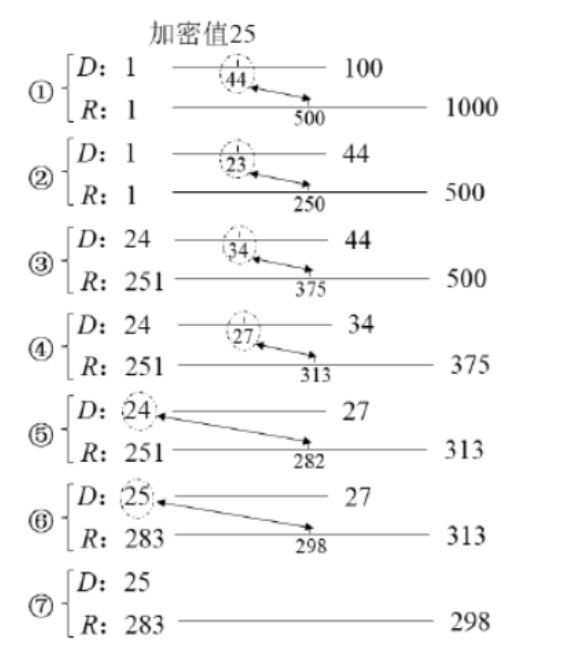


举个栗子~我要把[1, 100]这个区间的整数映射到[1, 1000]中，我们采用上述方法映射一个数字，我们先选择25进行映射。

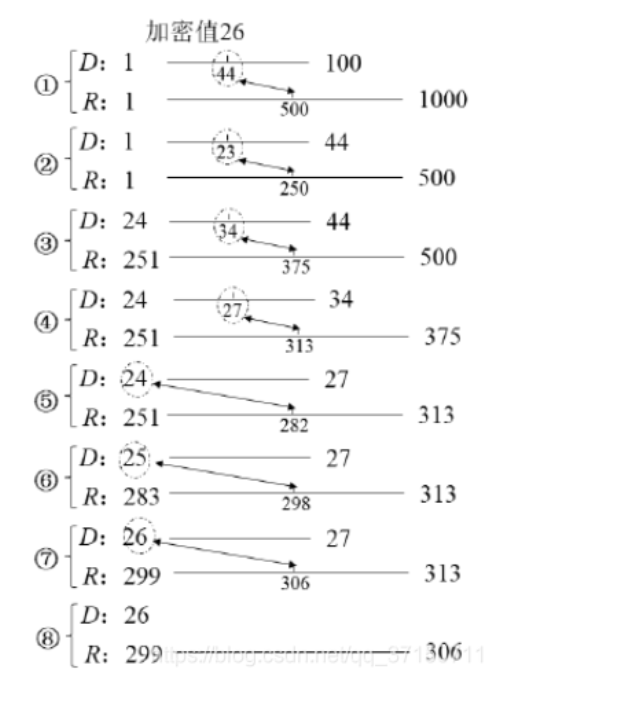
我们直接看第十行：x = HGD（M，N，y），它的意思是：我们从一个框里面抽球，里面黑球有M个（就是明文域的大小100），白球有N个（就是密文域的大小1000），我抽了y个球（就是密文域大小的一半500），这个函数可以给我返回我抽了多少个黑球，也就是数值x（这个值必然是0-100之间的数，因为我从1100个球里面抽，可能抽不到黑球，也可能抽到所有的黑球）。

接下来，阐述具体的超几何采样的步骤：

第一步：假设我采样到了44这个数，因为大于25，所以重新调整明文域和密文域分别为[1, 44]，[1, 500]；反之，如果我抽到的数小于25，假设是11，那么分别调整为[11, 100]，[500，1000]。



第二步：调整后，继续进行上述抽球活动。就这样反复抽几轮，一定能抽到明文域里的25，密文域也在一半一半的切，至于切左区间还是右区间，要看每次抽样抽到的x与25的大小关系，最后从区间[283, 298]的区间中随机取一个数作为25的密文。注意！不仅要抽样，还要把每次抽样结果记录在key\_table这张表里面！

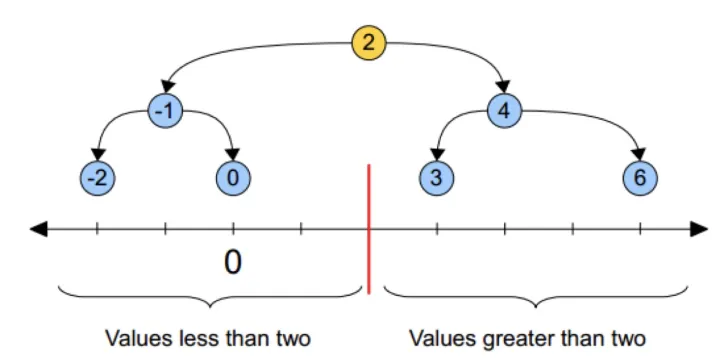
第三步：下面我们加密明文域里面的26，我们查看key\_table这张表发现，一直到原来的第⑥步，抽样的过程都可以省略，直接把之前抽到的x拿来用完全合理，但是第⑥步由于26>25，所以我要取第⑥步密文域的右半部分才合理即[299, 313]，然后剩下26与27，26就是值在左边，所以再取左区间[299, 306]。再从最后这个左区间随机取一个数作为26的密文。

代码：

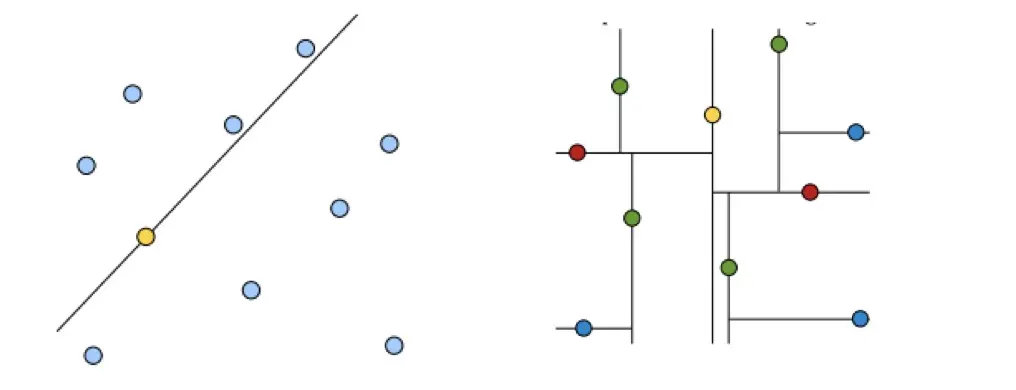
from pyope.ope import OPE,ValueRange  
cipher = OPE(b'long key' \* 2, in\_range=ValueRange(9999, 1000000),  
 out\_range=ValueRange(1, 1000000))  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 data=[]  
 k=0  
 with open("NE.txt", "r") as f: # 打开文件  
 file\_data = f.readlines() # 读取文件  
 for row in file\_data:  
 tmp\_list = row.split(' ')  
 tmp\_list[-1] = tmp\_list[-1].replace('\n','') #去掉换行符  
 tmp\_list[0]=cipher.encrypt(int(float(tmp\_list[0])\*1000000))  
 tmp\_list[1]=cipher.encrypt(int(float(tmp\_list[1])\*1000000))  
 print(k)  
 k+=1  
 data.append(tmp\_list)  
 with open("NE\_encrypto.txt","w") as f:  
 for i in data:  
 for j in i:  
 f.write(str(j))  
 f.write(' ')  
 f.write('\n')

1. kd树建立

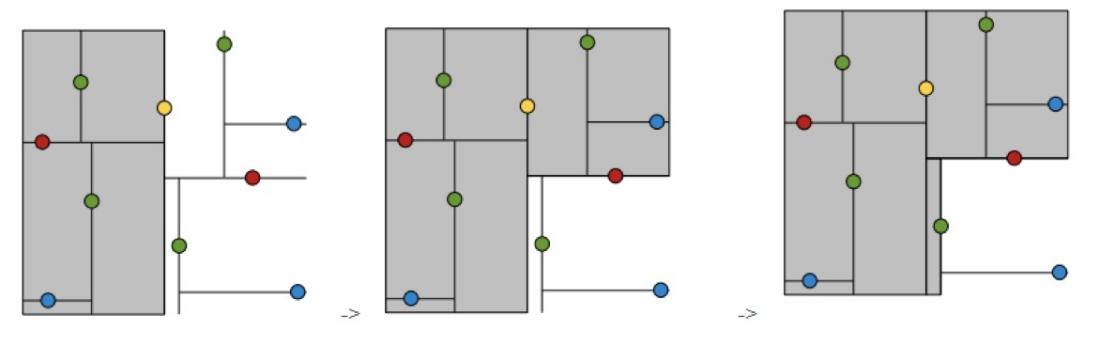
kd树：为了避免每次都重新计算一遍距离，算法会把距离信息保存在一棵树里，这样在计算之前从树里查询距离信息，尽量避免重新计算。其基本原理是，如果A和B距离很远，B和C距离很近，那么A和C的距离也很远。有了这个信息，就可以在合适的时候跳过距离远的点。 原理



黄色的点作为根节点，上面的点归左子树，下面的点归右子树，接下来再不断地划分，分割的那条线叫做分割超平面（splitting hyperplane），在一维中是一个点，二维中是线，三维的是面。



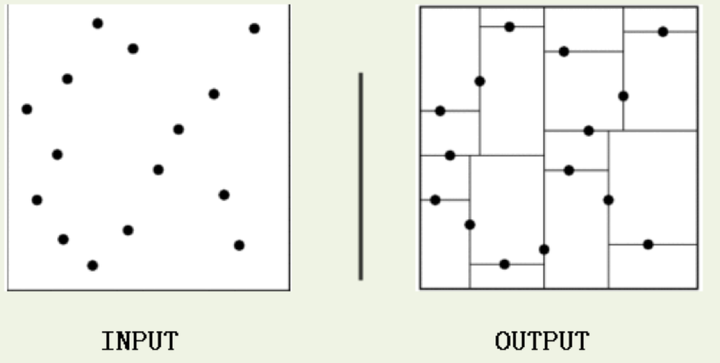
黄色节点就是Root节点，下一层是红色，再下一层是绿色，再下一层是蓝色。



1.树的建立；

2.最近邻域搜索（Nearest-Neighbor Lookup）

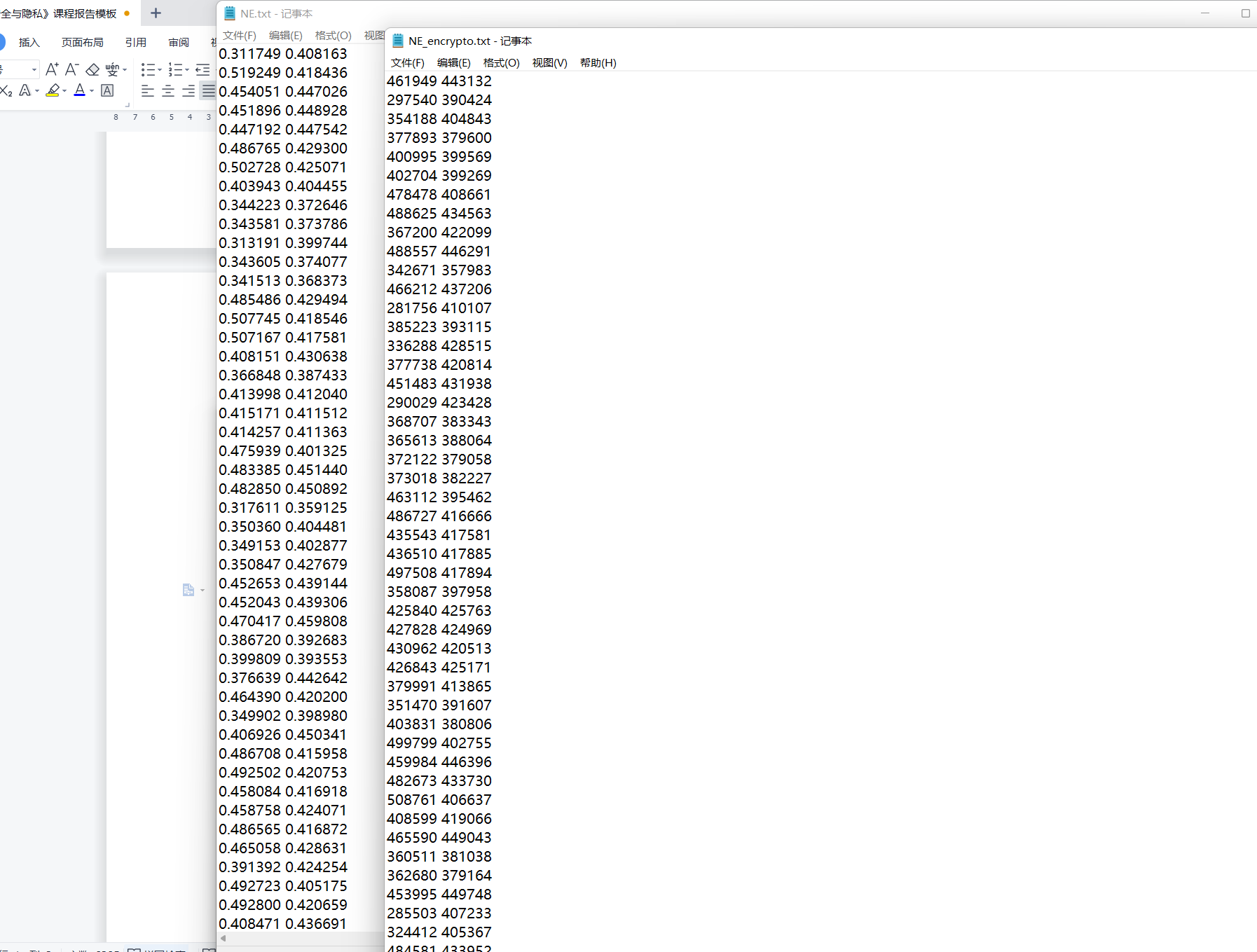
kd树(K-dimension tree)是一种对k维空间中的实例点进行存储以便对其进行快速检索的树形数据结构。kd树是一种二叉树，表示对k维空间的一个划分，构造kd树相当于不断地用垂直于坐标轴的超平面将K维空间切分，构成一系列的K维超矩形区域。kd树的每个结点对应于一个k维超矩形区域。利用kd树可以省去对大部分数据点的搜索，从而减少搜索的计算量。

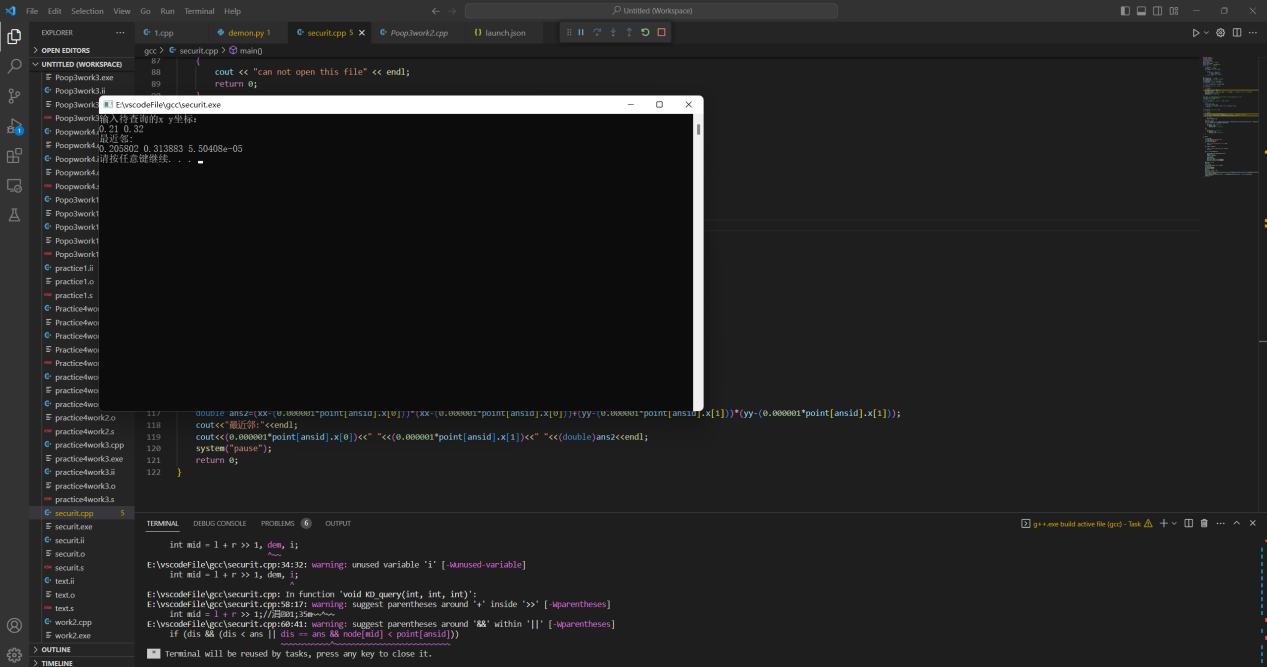


类比“二分查找”：给出一组数据：[9 1 4 7 2 5 0 3 8]，要查找8。如果挨个查找（线性扫描），那么将会把数据集都遍历一遍。而如果排一下序那数据集就变成了：[0 1 2 3 4 5 6 7 8 9]，按前一种方式我们进行了很多没有必要的查找，现在如果我们以5为分界点，那么数据集就被划分为了左右两个“簇” [0 1 2 3 4]和[6 7 8 9]。

因此，根本就没有必要进入第一个簇，可以直接进入第二个簇进行查找。把二分查找中的数据点换成k维数据点，这样的划分就变成了用超平面对k维空间的划分。空间划分就是对数据点进行分类，“挨得近”的数据点就在一个空间里面。

1. **实验结果：**





1. **实验总结与心得：**

从实验中学到了什么，以及有哪些收获和感悟。

**实验三：k匿名**

1. **实验原理：**

1．数据集: Adult Data Set ;

2. 编程自动化程序，根据输入参数k，自动生成k匿名结果;

3．能够形象展示出k匿名数据发布结果。

在大数据时代，随着数据的价值被人们更多地挖掘和利用，各种隐私保护问题也越来越多地引起了大家的注意。给予这一点，相继提出了多种隐私保护算法。今天我们来讲解一种匿名隐私保护模型——k-匿名模型

在讲解k-匿名模型之前，我们首先对用户数据类型进行一个分类：

①显式标识符（ID，能够唯一地确定一条用户记录）

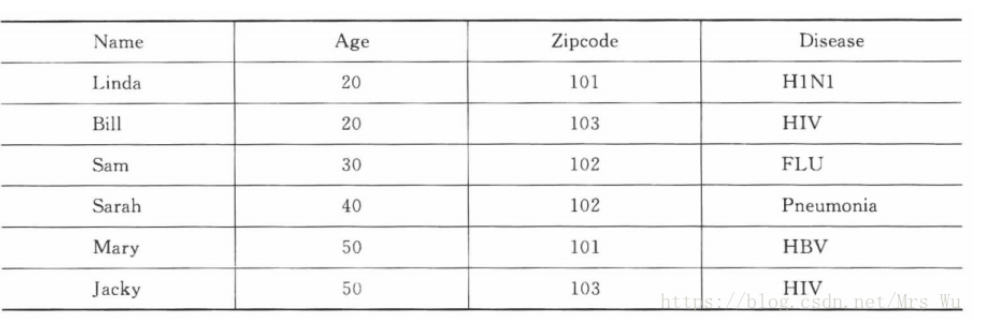
②准标识符（QI，能够以较高的概率结合一定的外部信息确定一条用户记录）

③敏感属性（需要保护的信息）

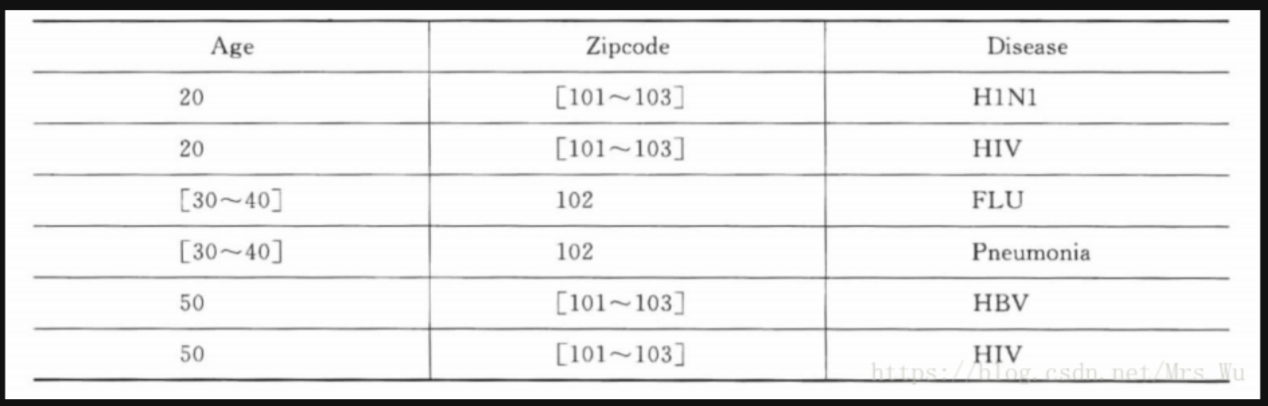
④非敏感属性（一般可以直接发布的信息）

接下来我们来正式介绍k-匿名模型，用T(Q1,Q2,...,Qd,S1,S2,...,Sm)来表达一张待发布的数据表，简称为T(d)，其中d是准标识符的个数，而m是敏感属性的个数。k-匿名机制要求表中的每一条记录都至少和表中的k-1条记录的准标识符QI一致。令是表T(d)在属性集合QI上的投影。表T(d)在属性集QI下满足k-匿名，当且仅当中的任意一条记录都至少重复出现k次。在II运算符下，有相同QI值的所有记录组成一个匿名组。对于一个给定的k，每一个这样的匿名组都是一个k-匿名组或者叫做一个等价类。

匿名化技术可以实现上述的k-匿名，概化和抑制是最常见的两种方法。所谓的概化是指对数据进行更加概括、抽象的描述，而抑制则是指不发布某些数据项。对下表中的数据进行概化：

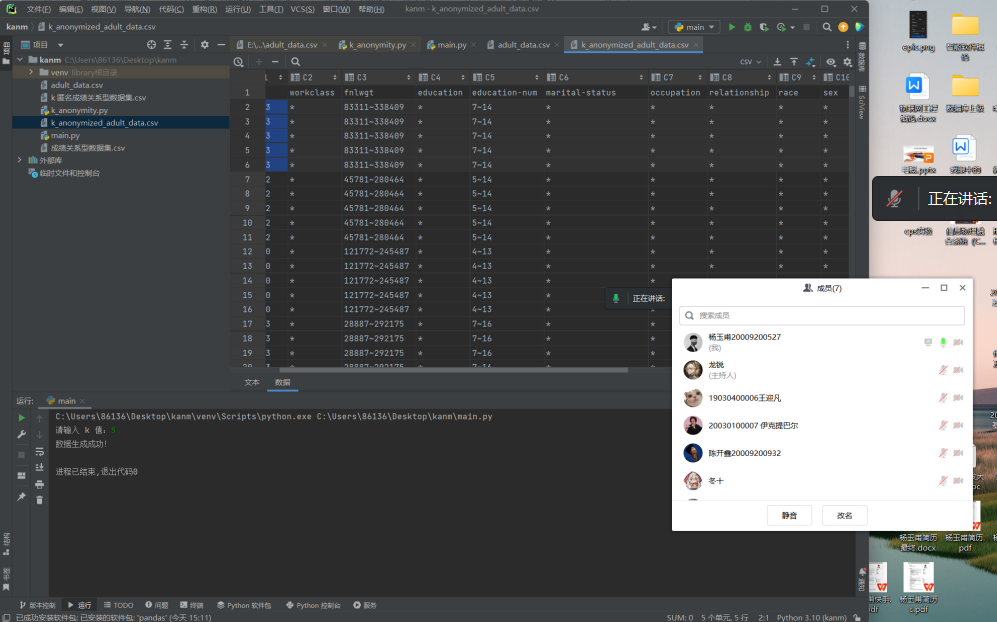


得到：



k-匿名攻击存在着同质化攻击和背景知识攻击两种缺陷。所谓同质化攻击是指某个k-匿名组内对应的敏感属性的值也完全相同，这使得攻击者可以轻易获取想要的信息。而背景知识攻击是指即使k-匿名组内的敏感属性值并不相同，攻击者也有可能依据其已有的背景知识以高概率获取到其隐私信息

1. **实验结果：**



**验收截图(线下验收免附)**