数据隐私 Lab1

范翔宇 PB18000006

实验环境

windows 10

vscode

C语言

问题描述

k-anonymity is a property possessed by certain anonymized data. The concept of *k*-anonymity was first introduced by Latanya Sweeney and Pierangela Samarati in a paper published in 1998 as an attempt to solve the problem: "Given person-specific field-structured data, produce a release of the data with scientific guarantees that the individuals who are the subjects of the data cannot be re-identified while the data remain practically useful." A release of data is said to have the *k*-anonymity property if the information for each person contained in the release cannot be distinguished from at least individuals whose information also appear in the release.

核心问题

如何正确输入并挑选我们想要的数据?

如何泛化到对应的阶层?

如何统计泛化好的数据?

如何判断是否满足条件?

必做部分

1.Samarati

(不想看实现过程的话，可用直接点击书签跳转到程序使用指南和结果分析)

输入部分

主要讲解从adult.data里读取正确数据，C语言遇到空格或换行符会停止输入，而且我们只需要部分有效列的数据，并不是全盘输入，所以要进行筛选。而Attributes包括'age', 'work_class', 'final_weight', 'education', 'education_num', 'marital_status', 'occupation', 'relationship', 'race', 'sex', 'capital_gain', 'capital_loss', 'hours_per_week', 'native_country', 'class'。依题意，我们只需要第一列、第六列、第七列、第九列、第十列即可。所以每次输入时用一个char数组暂存，如果到我们所需要的列，则strcpy给对应的数组。当然也不是所有行的数据都要输入。我们每次暂存时都判断存的数据是否为"?"如果存在，那么保持对应数组下标不变，直到下一行有效数据填入对应下标的时候，再进行加一操作。其中Age用int数组来存，方便后续处理，其余均用char数组。

泛化部分

泛化结果由另外char数组存；

Age

原始数据：即将int类型利用+'0'转换成char类型即可；

range 5：十位数部分即为原始数据的十位数(记为tens下同)，所以将int转换成char即可。而个位数部分，先判断让其除五，结果(记为fives)用int来存，那么就，如果结果是0，那么代表其对应[0,4]，即[fives * 5, fives * 5 + 4]，所以即将其泛化为[tens (fives * 5) - tens (fives * 5 + 4)]，如果结果是1，那么代表其对应[5,9]，同样可将其泛化为[tens (fives * 5) - tens (fives * 5 + 4)]，这里tens代表对应字符串第二位、第五位数据，(fives * 5)代表第三位数据，(fives * 5 + 4)代表第六位数据；

range 10：十位数部分同样为原始数据的十位数，统一泛化为[tens 0 - tens 9]；

range 20：个位数部分同样是0和9，而十位数部分要计算一下。先让十位数部分除以2，结果(同样用tens表示，下面的tens表示为该结果)用int来存，则可统一泛化为[(tens * 2)0 - (tens * 2 + 1)9]。比如26，十位数部分是2的1倍，即泛化为[20-29]；

*：所有数据，全部泛化为星号，strcpy即可。

Gender & Race

原始数据：同Age，将int转换成char；

*：所有数据，全部泛化为星号，strcpy即可。

Marital_Status

首先要辨识分辨，不同类型的数据。其实并不需要完全比较，可以靠数据的独特性判断某一位即可。对于"Never-married"，只有其第一位(在数组中对应第零位，后面不转换)为'N'；对于"Married-civ-spouse"，只有其第九位为'c'；对于"Married-AF-spouse"，只有其第九位为'A'；对于"Divorced"，只有其第一位为'D'；对于"Separated"，只有其第一位为'S'；对于"Widowed"，只有其第一位为'W'；对于"Married-spouse-absent"，只有其第九位为's。这样，我们只用比较一位就可以确定种类，进而泛化到对应部分。

统计部分

用一个四维的Count数组来存储统计结果。一个维度对应一种属性。我们把四个维度的下标称为"统计向量"。我们只需要保证统计向量独立唯一地对应一类泛化情况即可，即类似上述的唯一标识。

对于Age：原始数据下对应数值即可唯一标识Age；range 5下可用十位数乘2 + int(个位数 / 5)唯一标识；rang 10下即可用十位数唯一标识；rang 20下可用int(十位数 / 2)唯一标识；

对于Gender：Femal用1标识，Male用0标识；泛化为星号，统一为0；

对于Race：类似Marital_Status，只有"Other"的第一位为'O'，只有"Amer-Indian-Eskimo"的第二位为'm'，只有"Black"的第一位为'B'，只有"White"的第一位为'W'，只要"Asian-Pac-Islander"的第二位为's'。然后对应的赋一个与众不同的值即可，泛化为星号时，统一赋0；

对于Marital：标识同泛化部分。同样对应类型赋一个与众不同的值即可。

这样对应每一类泛化情况，我们的统计向量，都是独一无二的。在对每一行数据，泛化的同时，也在对应统计向量的Count ++。泛化完毕，也统计完毕。

判断部分

判断泛化结果是否是我们想要的，即遍历Count，统计小于10的，如果大于MaxSup，则直接return 0表明不满足条件，否则，大于等于10的则允许输出，然后return 1表面符合条件。

整体

大体按照PPT算法，还有每次找向量都是遍历，如果有等于try的，且之前未被使用即可代入。每次都是先泛化，泛化同时统计，然后判断是否满足条件。

Find_vector

INPUT: Table $T_i = \mathsf{PT}[QI]$ to be generalized, anonymity requirement k, suppression threshold MaxSup, lattice VL_{DT} of the distance vectors corresponding to the domain generalization hierarchy DGH_{DT}, where DT is the tuples of the domains of the quasi-identifier attributes.

OUTPUT: The distance vector *sol* of a generalized table GT_{sol} that is a k-minimal generalization of $\mathsf{PT}[QI]$ according to Definition 4.3.

METHOD: Executes a binary search on VL_{DT} based on height of vectors in VL_{DT}.

1. $low := 0$; $high := height(\top, \mathsf{VL}_{DT})$; $sol := \top$
2. **while** $low < high$
   - 2.1 $try := \lfloor \frac{low+high}{2} \rfloor$
   - 2.2 $Vectors := \{vec \mid height(vec, \mathsf{VL}_{DT}) = try\}$
   - 2.3 $reach\_k :=$ **false**
   - 2.4 **while** $Vectors \neq \emptyset \wedge reach\_k \neq$ **true do**
     - Select and remove a vector vec from $Vectors$
     - **if** satisfies(vec,k,T_i,MaxSup) **then** $sol := vec$; $reach\_k :=$ **true**
   - 2.5 **if** $reach\_k =$ **true then** $high := try$ **else** $low := try + 1$
3. **Return** *sol*

程序使用指南

修改第31行和第324行的文件路径即可，然后运行输入K和MAXSUP，静待几秒后会运行完毕。

结果分析

```
PS C:\Users\Lenovo> cd "g:\360MoveData\Users\Lenovo\Desktop\" ; if ($?) { gcc Samarati.c -o Samarati } ; if ($?) { .\Samarati }
--------------------please input your K and MAXSUP--------------------
10 20
--------------------please wait--------------------
----------------------finish and output----------------------
cost 6386ms time
the best LM is 2.066455
Age:1 Race:1 Gender:0 Marital:2
PS G:\360MoveData\Users\Lenovo\Desktop>
```

result.txt - 记事本

```
[35-39]  *  Male,  *  Adm-clerical,
[50-54]  *  Male,  *  Exec-managerial,
[35-39]  *  Male,  *  Handlers-cleaners,
[50-54]  *  Male,  *  Handlers-cleaners,
[25-29]  *  Female,  *  Prof-specialty,
[35-39]  *  Female,  *  Exec-managerial,
[45-49]  *  Female,  *  Other-service,
[50-54]  *  Male,  *  Exec-managerial,
[30-34]  *  Female,  *  Prof-specialty,
[40-44]  *  Male,  *  Exec-managerial,
[35-39]  *  Male,  *  Exec-managerial,
[30-34]  *  Male,  *  Prof-specialty,
[20-24]  *  Female,  *  Adm-clerical,
[30-34]  *  Male,  *  Sales,
[30-34]  *  Male,  *  Transport-moving,
[25-29]  *  Male,  *  Farming-fishing,
[30-34]  *  Male,  *  Machine-op-inspct,
[35-39]  *  Male,  *  Sales,
[40-44]  *  Female,  *  Exec-managerial,
[40-44]  *  Male,  *  Prof-specialty,
[50-54]  *  Female,  *  Other-service,
[35-39]  *  Male,  *  Farming-fishing,
[40-44]  *  Male,  *  Transport-moving,
[55-59]  *  Female,  *  Tech-support,
[55-59]  *  Male,  *  Tech-support,
[15-19]  *  Male,  *  Craft-repair,
[35-39]  *  Male,  *  Exec-managerial,
[45-49]  *  Male,  *  Craft-repair,
[20-24]  *  Male,  *  Protective-serv,
[20-24]  *  Male,  *  Sales,
[45-49]  *  Male,  *  Exec-managerial,
[30-34]  *  Male,  *  Adm-clerical,
[20-24]  *  Male,  *  Other-service,
[45-49]  *  Male,  *  Machine-op-inspct,
```

adult.data - 记事本

```
39, State-gov, 77516, Bachelors, 13, Never-married, Adm-clerical, Not-in-family, White, Male, 2174, 0, 40, United-States, <=50K
50, Self-emp-not-inc, 83311, Bachelors, 13, Married-civ-spouse, Exec-managerial, Husband, White, Male, 0, 0, 13, United-States, <=50K
38, Private, 215646, HS-grad, 9, Divorced, Handlers-cleaners, Not-in-family, White, Male, 0, 0, 40, United-States, <=50K
53, Private, 234721, 11th, 7, Married-civ-spouse, Handlers-cleaners, Husband, Black, Male, 0, 0, 40, United-States, <=50K
28, Private, 338409, Bachelors, 13, Married-civ-spouse, Prof-specialty, Wife, Black, Female, 0, 0, 40, Cuba, <=50K
37, Private, 284582, Masters, 14, Married-civ-spouse, Exec-managerial, Wife, White, Female, 0, 0, 40, United-States, <=50K
49, Private, 160187, 9th, 5, Married-spouse-absent, Other-service, Not-in-family, Black, Female, 0, 0, 16, Jamaica, <=50K
52, Self-emp-not-inc, 209642, HS-grad, 9, Married-civ-spouse, Exec-managerial, Husband, White, Male, 0, 0, 45, United-States, >50K
31, Private, 45781, Masters, 14, Never-married, Prof-specialty, Not-in-family, White, Female, 14084, 0, 50, United-States, >50K
42, Private, 159449, Bachelors, 13, Married-civ-spouse, Exec-managerial, Husband, White, Male, 5178, 0, 40, United-States, >50K
37, Private, 280464, Some-college, 10, Married-civ-spouse, Exec-managerial, Husband, Black, Male, 0, 0, 80, United-States, >50K
30, State-gov, 141297, Bachelors, 13, Married-civ-spouse, Prof-specialty, Husband, Asian-Pac-Islander, Male, 0, 0, 40, India, >50K
23, Private, 122272, Bachelors, 13, Never-married, Adm-clerical, Own-child, White, Female, 0, 0, 30, United-States, <=50K
32, Private, 205019, Assoc-acdm, 12, Never-married, Sales, Not-in-family, Black, Male, 0, 0, 50, United-States, <=50K
40, Private, 121772, Assoc-voc, 11, Married-civ-spouse, Craft-repair, Husband, Asian-Pac-Islander, Male, 0, 0, 40, ?, >50K
34, Private, 245487, 7th-8th, 4, Married-civ-spouse, Transport-moving, Husband, Amer-Indian-Eskimo, Male, 0, 0, 45, Mexico, <=50K
25, Self-emp-not-inc, 176756, HS-grad, 9, Never-married, Farming-fishing, Own-child, White, Male, 0, 0, 35, United-States, <=50K
32, Private, 186824, HS-grad, 9, Never-married, Machine-op-inspct, Unmarried, White, Male, 0, 0, 40, United-States, <=50K
38, Private, 28887, 11th, 7, Married-civ-spouse, Sales, Husband, White, Male, 0, 0, 50, United-States, <=50K
43, Self-emp-not-inc, 292175, Masters, 14, Divorced, Exec-managerial, Unmarried, White, Female, 0, 0, 45, United-States, >50K
40, Private, 193524, Doctorate, 16, Married-civ-spouse, Prof-specialty, Husband, White, Male, 0, 0, 60, United-States, >50K
54, Private, 302146, HS-grad, 9, Separated, Other-service, Unmarried, Black, Female, 0, 0, 20, United-States, <=50K
35, Federal-gov, 76845, 9th, 5, Married-civ-spouse, Farming-fishing, Husband, Black, Male, 0, 0, 40, United-States, <=50K
43, Private, 117037, 11th, 7, Married-civ-spouse, Transport-moving, Husband, White, Male, 0, 2042, 40, United-States, <=50K
59, Private, 109015, HS-grad, 9, Divorced, Tech-support, Unmarried, White, Female, 0, 0, 40, United-States, <=50K
56, Local-gov, 216851, Bachelors, 13, Married-civ-spouse, Tech-support, Husband, White, Male, 0, 0, 40, United-States, >50K
19, Private, 168294, HS-grad, 9, Never-married, Craft-repair, Own-child, White, Male, 0, 0, 40, United-States, <=50K
54, ?, 180211, Some-college, 10, Married-civ-spouse, ?, Husband, Asian-Pac-Islander, Male, 0, 0, 60, South, >50K
39, Private, 367260, HS-grad, 9, Divorced, Exec-managerial, Not-in-family, White, Male, 0, 0, 80, United-States, <=50K
49, Private, 193366, HS-grad, 9, Married-civ-spouse, Craft-repair, Husband, White, Male, 0, 0, 40, United-States, <=50K
23, Local-gov, 190709, Assoc-acdm, 12, Never-married, Protective-serv, Not-in-family, White, Male, 0, 0, 52, United-States, <=50K
20, Private, 266015, Some-college, 10, Never-married, Sales, Own-child, Black, Male, 0, 0, 44, United-States, <=50K
45, Private, 386940, Bachelors, 13, Divorced, Exec-managerial, Own-child, White, Male, 0, 1408, 40, United-States, <=50K
30, Federal-gov, 59951, Some-college, 10, Married-civ-spouse, Adm-clerical, Own-child, White, Male, 0, 0, 40, United-States, <=50K
```

第 10 行，第 115 列　100%　Windows (CRLF)　UTF-8

去除了存在"?"的行，而且泛化结果遵循原始数据和泛化阶层。

下面测试不同K和MAXSUP对实验结果的影响

K = 10 MAXSUP = 5

```
PS C:\Users\Lenovo> cd "g:\360MoveData\Users\Lenovo\Desktop\" ; if ($?) { gcc Samarati.c -o Samarati } ; if ($?) { .\Samarati }
----------------------please input your K and MAXSUP----------------------
10 5
----------------------please wait----------------------
----------------------finish and output----------------------
cost 6192ms time
the best LM is 2.000000
Age:4 Race:0 Gender:0 Marital:2
PS G:\360MoveData\Users\Lenovo\Desktop>
```

K = 10 MAXSUP = 50

```
PS G:\360MoveData\Users\Lenovo\Desktop> cd "g:\360MoveData\Users\Lenovo\Desktop\" ; if ($?) { gcc Samarati.c -o Samarati } ; if ($?) { .\Samarati }
----------------------please input your K and MAXSUP----------------------
10 50
----------------------please wait----------------------
----------------------finish and output----------------------
cost 6525ms time
the best LM is 1.625000
Age:2 Race:1 Gender:0 Marital:1
PS G:\360MoveData\Users\Lenovo\Desktop>
```

K = 10 MAXSUP = 90

```
PS C:\Users\Lenovo> cd "g:\360MoveData\Users\Lenovo\Desktop\" ; if ($?) { gcc Samarati.c -o Samarati } ; if ($?) { .\Samarati }
----------------------please input your K and MAXSUP----------------------
10 90
----------------------please wait----------------------
----------------------finish and output----------------------
cost 6228ms time
the best LM is 1.000000
Age:4 Race:0 Gender:0 Marital:0
PS G:\360MoveData\Users\Lenovo\Desktop>
```

K = 10 MAXSUP = 300

```
PS G:\360MoveData\Users\Lenovo\Desktop> cd "g:\360MoveData\Users\Lenovo\Desktop\" ; if ($?) { gcc Samarati.c -o Samarati } ; if ($?) { .\Samarati }
----------------------please input your K and MAXSUP----------------------
10 300
----------------------please wait----------------------
----------------------finish and output----------------------
cost 4887ms time
the best LM is 0.722475
Age:3 Race:0 Gender:0 Marital:1
PS G:\360MoveData\Users\Lenovo\Desktop>
```

K = 10 MAXSUP = 600

```
PS G:\360MoveData\Users\Lenovo\Desktop> cd "g:\360MoveData\Users\Lenovo\Desktop\" ; if ($?) { gcc Samarati.c -o Samarati } ; if ($?) { .\Samarati }
----------------------please input your K and MAXSUP----------------------
10 600
----------------------please wait----------------------
----------------------finish and output----------------------
cost 4900ms time
the best LM is 0.125000
Age:2 Race:0 Gender:0 Marital:0
PS G:\360MoveData\Users\Lenovo\Desktop>
```

K = 10 MAXSUP = 1000

```
PS C:\Users\Lenovo> cd "g:\360MoveData\Users\Lenovo\Desktop\" ; if ($?) { gcc Samarati.c -o Samarati } ; if ($?) { .\Samarati }
----------------------please input your K and MAXSUP----------------------
10 1000
----------------------please wait----------------------
----------------------finish and output----------------------
cost 5092ms time
the best LM is 0.066656
Age:1 Race:0 Gender:0 Marital:0
PS G:\360MoveData\Users\Lenovo\Desktop>
```

可知，当K固定时，MAXSUP越大，满足条件的泛化阶层总和越低，对应的LM也越小，花费的时间也略微减少。

K = 20 MAXSUP = 20

```
PS C:\Users\Lenovo> cd "g:\360MoveData\Users\Lenovo\Desktop\" ; if ($?) { gcc Samarati.c -o Samarati } ; if ($?) { .\Samarati }
--------------------please input your K and MAXSUP--------------------
20 20
----------------------please wait----------------------
----------------------finish and output----------------------
cost 6568ms time
the best LM is 2.000000
Age:4 Race:0 Gender:0 Marital:2
PS G:\360MoveData\Users\Lenovo\Desktop>
```

K = 100 MAXSUP = 20

```
PS G:\360MoveData\Users\Lenovo\Desktop> cd "g:\360MoveData\Users\Lenovo\Desktop\" ; if ($?) { gcc Samarati.c -o Samarati } ; if ($?) { .\Samarati }
--------------------please input your K and MAXSUP--------------------
100 20
----------------------please wait----------------------
----------------------finish and output----------------------
cost 6311ms time
the best LM is 2.500000
Age:4 Race:1 Gender:0 Marital:1
PS G:\360MoveData\Users\Lenovo\Desktop>
```

K = 500 MAXSUP = 20

```
PS G:\360MoveData\Users\Lenovo\Desktop> cd "g:\360MoveData\Users\Lenovo\Desktop\" ; if ($?) { gcc Samarati.c -o Samarati } ; if ($?) { .\Samarati }
--------------------please input your K and MAXSUP--------------------
500 20
----------------------please wait----------------------
----------------------finish and output----------------------
cost 4924ms time
the best LM is 3.000000
Age:4 Race:1 Gender:0 Marital:2
PS G:\360MoveData\Users\Lenovo\Desktop>
```

K = 36000 MAXSUP = 20

```
PS G:\360MoveData\Users\Lenovo\Desktop> cd "g:\360MoveData\Users\Lenovo\Desktop\" ; if ($?) { gcc Samarati.c -o Samarati } ; if ($?) { .\Samarati }
--------------------please input your K and MAXSUP--------------------
36000 20
----------------------please wait----------------------
----------------------finish and output----------------------
cost 4867ms time
the best LM is 100000.000000
Age:0 Race:0 Gender:0 Marital:0
PS G:\360MoveData\Users\Lenovo\Desktop>
```

这里需要额外说明，全部为0 0 0 0，代表没有找到合适的泛化阶层，唯一的解决办法是全部数据泛化到星号。这里不是4 2 2 1的原因是我的是int类型的try = (low + height)/2;所以始终到不了泛化阶层8。这里的LM是我设的初值。

可知，当MAXSUP固定时，K越大，满足条件的泛化阶层总和越高，对应的LM也越大，花费的时间同样减小。

所以，综合考虑，我们设计K匿名时，可用设计K略小，MAXSUP略大的K匿名，来达到很好的LM。

2.Mondrain

(同样，不想看实现过程的话，可用直接点击书签跳转到程序使用指南和结果分析)

输入部分

参考第一部分，不赘述。需要额外注意的时，EducationNum可能只有一位数，所以要多加判断即可。而且Age和EducationNum均由int数组来存。occupation存到char数组。

泛化部分

无论是EducationNum还是Age，在某一部分泛化都是**[该部分内最小数据-该部分内最大数据]**，当然泛化后是字符串，所以要最小值、最大值除以10，来确定十位数，然后将数据 - 10 * 十位数即得个位数。另外找最小值找最大值也不算麻烦。

排序部分

普通的快排，虽然只对一个数组排序，但有另外两个数组随着其改变。这样可以确保其数据仍是一一对应的。

递归部分

每次递归都是先选择一个维度(即Age或EducationNum)，先对该维度排序，然后分为两部分，下界到 (strict - 1) 和 strict 到 上界 继续递归。这里的strict即PPT中的含义。

判断部分

首先找strict，然后判断递归用到的维度下，下界和(strict-1)、strict和上界之间是否均满足K匿名(即个数均大于等于K)，如果其中一个不满足，那么就代表这个维度不能继续递归，我们这时不能直接泛化，而是跳转到另一维度，对另一维度排序后，然后同样找该维度的strict是否和上界、下界满足条件。如果满足K匿名，那么久按照这个维度继续递归，如果不满足，那么就标明已经不能分割了，就开始泛化部分。

整体

遵循PPT算法，大概就是判断→选择维度→排序→按选择维度递归或者判断→判断另一维度→排序→按另一维度递归、判断→判断另一维度→排序→泛化

Anonymize($partition$)
 if (no allowable multidimensional cut for $partition$)
  return $\phi : partition \rightarrow summary$
 else
  $dim \leftarrow$ choose_dimension()
  $fs \leftarrow$ frequency_set($partition, dim$)
  $splitVal \leftarrow$ find_median(fs)
  $lhs \leftarrow \{t \in partition : t.dim \leq splitVal\}$
  $rhs \leftarrow \{t \in partition : t.dim > splitVal\}$
  return Anonymize(rhs) $\cup$ Anonymize(lhs)

程序使用指南

你需要更改的是第24行和第291行输入输出的文件路径，另外如果想测试不同的K，需要更改第六行define中K的值。

结果分析

由于是打乱顺序输出，而且每次选择维度都是随机的，部分细节无从考证，但是和原始结果无异议。

result_mon.txt - 记事本

文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

```
[17-17]    [03-07]    Other-service,
[17-17]    [03-07]    Farming-fishing,
[17-17]    [03-07]    Sales,
[17-17]    [03-07]    Craft-repair,
[17-17]    [03-07]    Other-service,
[17-17]    [03-07]    Other-service,
[17-17]    [03-07]    Sales,
[17-17]    [03-07]    Other-service,
[17-17]    [03-07]    Craft-repair,
[17-17]    [03-07]    Handlers-cleaners,
[17-17]    [03-07]    Machine-op-inspct,
[17-17]    [03-07]    Other-service,
[17-17]    [03-07]    Other-service,
[17-17]    [03-07]    Other-service,
[17-17]    [03-07]    Other-service,
[17-17]    [03-07]    Sales,
[17-17]    [03-07]    Other-service,
[17-17]    [03-07]    Other-service,
[17-17]    [03-07]    Other-service,
[17-17]    [03-07]    Other-service,
[17-17]    [03-07]    Other-service,
[17-17]    [03-07]    Craft-repair,
[17-17]    [03-07]    Sales,
[17-17]    [03-07]    Other-service,
[17-17]    [03-07]    Other-service,
[17-17]    [03-07]    Other-service,
[17-17]    [03-07]    Transport-moving,
[17-17]    [03-07]    Other-service,
[17-17]    [03-07]    Other-service,
[17-17]    [03-07]    Sales,
[17-17]    [03-07]    Other-service,
[17-17]    [03-07]    Handlers-cleaners,
[17-17]    [03-07]    Other-service,
[17-17]    [03-07]    Craft-repair,
```

K = 5 运行10次

```
PS C:\Users\Lenovo> cd "g:\360MoveData\Users\Lenovo\Desktop\" ; if ($?) { gcc Mondrian.c -o Mondrian } ; if ($?) { .\Mondrian }
cost 235ms time
LM is 0.164079
PS G:\360MoveData\Users\Lenovo\Desktop> cd "g:\360MoveData\Users\Lenovo\Desktop\" ; if ($?) { gcc Mondrian.c -o Mondrian } ; if ($?) { .\Mondrian }
cost 388ms time
LM is 0.111828
PS G:\360MoveData\Users\Lenovo\Desktop> cd "g:\360MoveData\Users\Lenovo\Desktop\" ; if ($?) { gcc Mondrian.c -o Mondrian } ; if ($?) { .\Mondrian }
cost 277ms time
LM is 0.167403
PS G:\360MoveData\Users\Lenovo\Desktop> cd "g:\360MoveData\Users\Lenovo\Desktop\" ; if ($?) { gcc Mondrian.c -o Mondrian } ; if ($?) { .\Mondrian }
cost 416ms time
LM is 0.080244
PS G:\360MoveData\Users\Lenovo\Desktop> cd "g:\360MoveData\Users\Lenovo\Desktop\" ; if ($?) { gcc Mondrian.c -o Mondrian } ; if ($?) { .\Mondrian }
cost 385ms time
LM is 0.095698
PS G:\360MoveData\Users\Lenovo\Desktop> cd "g:\360MoveData\Users\Lenovo\Desktop\" ; if ($?) { gcc Mondrian.c -o Mondrian } ; if ($?) { .\Mondrian }
cost 201ms time
LM is 0.238819
PS G:\360MoveData\Users\Lenovo\Desktop> cd "g:\360MoveData\Users\Lenovo\Desktop\" ; if ($?) { gcc Mondrian.c -o Mondrian } ; if ($?) { .\Mondrian }
cost 704ms time
LM is 0.058614
PS G:\360MoveData\Users\Lenovo\Desktop> cd "g:\360MoveData\Users\Lenovo\Desktop\" ; if ($?) { gcc Mondrian.c -o Mondrian } ; if ($?) { .\Mondrian }
cost 575ms time
LM is 0.082970
PS G:\360MoveData\Users\Lenovo\Desktop> cd "g:\360MoveData\Users\Lenovo\Desktop\" ; if ($?) { gcc Mondrian.c -o Mondrian } ; if ($?) { .\Mondrian }
cost 472ms time
LM is 0.024477
PS G:\360MoveData\Users\Lenovo\Desktop> cd "g:\360MoveData\Users\Lenovo\Desktop\" ; if ($?) { gcc Mondrian.c -o Mondrian } ; if ($?) { .\Mondrian }
cost 254ms time
LM is 0.136484
PS G:\360MoveData\Users\Lenovo\Desktop>
```

平均LM为0.1167068，平均运行时间为390.7ms

K = 20 运行10次

```
PS C:\Users\Lenovo> cd "g:\360MoveData\Users\Lenovo\Desktop\" ; if ($?) { gcc Mondrian.c -o Mondrian } ; if ($?) { .\Mondrian }
cost 231ms time
LM is 0.126935
PS G:\360MoveData\Users\Lenovo\Desktop> cd "g:\360MoveData\Users\Lenovo\Desktop\" ; if ($?) { gcc Mondrian.c -o Mondrian } ; if ($?) { .\Mondrian }
cost 442ms time
LM is 0.111148
PS G:\360MoveData\Users\Lenovo\Desktop> cd "g:\360MoveData\Users\Lenovo\Desktop\" ; if ($?) { gcc Mondrian.c -o Mondrian } ; if ($?) { .\Mondrian }
cost 445ms time
LM is 0.111148
PS G:\360MoveData\Users\Lenovo\Desktop> cd "g:\360MoveData\Users\Lenovo\Desktop\" ; if ($?) { gcc Mondrian.c -o Mondrian } ; if ($?) { .\Mondrian }
cost 238ms time
LM is 0.160397
PS G:\360MoveData\Users\Lenovo\Desktop> cd "g:\360MoveData\Users\Lenovo\Desktop\" ; if ($?) { gcc Mondrian.c -o Mondrian } ; if ($?) { .\Mondrian }
cost 215ms time
LM is 0.172502
PS G:\360MoveData\Users\Lenovo\Desktop> cd "g:\360MoveData\Users\Lenovo\Desktop\" ; if ($?) { gcc Mondrian.c -o Mondrian } ; if ($?) { .\Mondrian }
cost 416ms time
LM is 0.096377
PS G:\360MoveData\Users\Lenovo\Desktop> cd "g:\360MoveData\Users\Lenovo\Desktop\" ; if ($?) { gcc Mondrian.c -o Mondrian } ; if ($?) { .\Mondrian }
cost 422ms time
LM is 0.096377
PS G:\360MoveData\Users\Lenovo\Desktop> cd "g:\360MoveData\Users\Lenovo\Desktop\" ; if ($?) { gcc Mondrian.c -o Mondrian } ; if ($?) { .\Mondrian }
cost 332ms time
LM is 0.104094
PS G:\360MoveData\Users\Lenovo\Desktop> cd "g:\360MoveData\Users\Lenovo\Desktop\" ; if ($?) { gcc Mondrian.c -o Mondrian } ; if ($?) { .\Mondrian }
cost 437ms time
LM is 0.051150
PS G:\360MoveData\Users\Lenovo\Desktop> cd "g:\360MoveData\Users\Lenovo\Desktop\" ; if ($?) { gcc Mondrian.c -o Mondrian } ; if ($?) { .\Mondrian }
cost 557ms time
LM is 0.067944
PS G:\360MoveData\Users\Lenovo\Desktop>
```

平均LM为0.1001695，平均运行时间为373.5ms

K = 50 运行10次

```
PS C:\Users\Lenovo> cd "g:\360MoveData\Users\Lenovo\Desktop\" ; if ($?) { gcc Mondrian.c -o Mondrian } ; if ($?) { .\Mondrian }
cost 236ms time
LM is 0.198609
PS G:\360MoveData\Users\Lenovo\Desktop> cd "g:\360MoveData\Users\Lenovo\Desktop\" ; if ($?) { gcc Mondrian.c -o Mondrian } ; if ($?) { .\Mondrian }
cost 226ms time
LM is 0.198609
PS G:\360MoveData\Users\Lenovo\Desktop> cd "g:\360MoveData\Users\Lenovo\Desktop\" ; if ($?) { gcc Mondrian.c -o Mondrian } ; if ($?) { .\Mondrian }
cost 229ms time
LM is 0.181032
PS G:\360MoveData\Users\Lenovo\Desktop> cd "g:\360MoveData\Users\Lenovo\Desktop\" ; if ($?) { gcc Mondrian.c -o Mondrian } ; if ($?) { .\Mondrian }
cost 193ms time
LM is 0.217769
PS G:\360MoveData\Users\Lenovo\Desktop> cd "g:\360MoveData\Users\Lenovo\Desktop\" ; if ($?) { gcc Mondrian.c -o Mondrian } ; if ($?) { .\Mondrian }
cost 472ms time
LM is 0.126556
PS G:\360MoveData\Users\Lenovo\Desktop> cd "g:\360MoveData\Users\Lenovo\Desktop\" ; if ($?) { gcc Mondrian.c -o Mondrian } ; if ($?) { .\Mondrian }
cost 384ms time
LM is 0.103574
PS G:\360MoveData\Users\Lenovo\Desktop> cd "g:\360MoveData\Users\Lenovo\Desktop\" ; if ($?) { gcc Mondrian.c -o Mondrian } ; if ($?) { .\Mondrian }
cost 441ms time
LM is 0.098215
PS G:\360MoveData\Users\Lenovo\Desktop> cd "g:\360MoveData\Users\Lenovo\Desktop\" ; if ($?) { gcc Mondrian.c -o Mondrian } ; if ($?) { .\Mondrian }
cost 228ms time
LM is 0.189933
PS G:\360MoveData\Users\Lenovo\Desktop> cd "g:\360MoveData\Users\Lenovo\Desktop\" ; if ($?) { gcc Mondrian.c -o Mondrian } ; if ($?) { .\Mondrian }
cost 186ms time
LM is 0.208483
PS G:\360MoveData\Users\Lenovo\Desktop> cd "g:\360MoveData\Users\Lenovo\Desktop\" ; if ($?) { gcc Mondrian.c -o Mondrian } ; if ($?) { .\Mondrian }
cost 491ms time
LM is 0.076591
PS G:\360MoveData\Users\Lenovo\Desktop>
```

平均LM为0.1599371，平均时间为308.6ms

K = 200 运行10次

```
PS C:\Users\Lenovo> cd "g:\360MoveData\Users\Lenovo\Desktop\" ; if ($?) { gcc Mondrian.c -o Mondrian } ; if ($?) { .\Mondrian }
cost 217ms time
LM is 0.337433
PS G:\360MoveData\Users\Lenovo\Desktop> cd "g:\360MoveData\Users\Lenovo\Desktop\" ; if ($?) { gcc Mondrian.c -o Mondrian } ; if ($?) { .\Mondrian }
cost 187ms time
LM is 0.337433
PS G:\360MoveData\Users\Lenovo\Desktop> cd "g:\360MoveData\Users\Lenovo\Desktop\" ; if ($?) { gcc Mondrian.c -o Mondrian } ; if ($?) { .\Mondrian }
cost 415ms time
LM is 0.168251
PS G:\360MoveData\Users\Lenovo\Desktop> cd "g:\360MoveData\Users\Lenovo\Desktop\" ; if ($?) { gcc Mondrian.c -o Mondrian } ; if ($?) { .\Mondrian }
cost 430ms time
LM is 0.168251
PS G:\360MoveData\Users\Lenovo\Desktop> cd "g:\360MoveData\Users\Lenovo\Desktop\" ; if ($?) { gcc Mondrian.c -o Mondrian } ; if ($?) { .\Mondrian }
cost 193ms time
LM is 0.295084
PS G:\360MoveData\Users\Lenovo\Desktop> cd "g:\360MoveData\Users\Lenovo\Desktop\" ; if ($?) { gcc Mondrian.c -o Mondrian } ; if ($?) { .\Mondrian }
cost 377ms time
LM is 0.230982
PS G:\360MoveData\Users\Lenovo\Desktop> cd "g:\360MoveData\Users\Lenovo\Desktop\" ; if ($?) { gcc Mondrian.c -o Mondrian } ; if ($?) { .\Mondrian }
cost 463ms time
LM is 0.179349
PS G:\360MoveData\Users\Lenovo\Desktop> cd "g:\360MoveData\Users\Lenovo\Desktop\" ; if ($?) { gcc Mondrian.c -o Mondrian } ; if ($?) { .\Mondrian }
cost 267ms time
LM is 0.204399
PS G:\360MoveData\Users\Lenovo\Desktop> cd "g:\360MoveData\Users\Lenovo\Desktop\" ; if ($?) { gcc Mondrian.c -o Mondrian } ; if ($?) { .\Mondrian }
cost 472ms time
LM is 0.150319
PS G:\360MoveData\Users\Lenovo\Desktop> cd "g:\360MoveData\Users\Lenovo\Desktop\" ; if ($?) { gcc Mondrian.c -o Mondrian } ; if ($?) { .\Mondrian }
cost 250ms time
LM is 0.260093
PS G:\360MoveData\Users\Lenovo\Desktop>
```

平均LM为0.23316849，平均运行时间为332.5ms

K = 2000 运行10次

```
PS C:\Users\Lenovo> cd "g:\360MoveData\Users\Lenovo\Desktop\" ; if ($?) { gcc Mondrian.c -o Mondrian } ; if ($?) { .\Mondrian }
cost 174ms time
LM is 0.742979
PS G:\360MoveData\Users\Lenovo\Desktop> cd "g:\360MoveData\Users\Lenovo\Desktop\" ; if ($?) { gcc Mondrian.c -o Mondrian } ; if ($?) { .\Mondrian }
cost 382ms time
LM is 0.730627
PS G:\360MoveData\Users\Lenovo\Desktop> cd "g:\360MoveData\Users\Lenovo\Desktop\" ; if ($?) { gcc Mondrian.c -o Mondrian } ; if ($?) { .\Mondrian }
cost 343ms time
LM is 0.700542
PS G:\360MoveData\Users\Lenovo\Desktop> cd "g:\360MoveData\Users\Lenovo\Desktop\" ; if ($?) { gcc Mondrian.c -o Mondrian } ; if ($?) { .\Mondrian }
cost 401ms time
LM is 0.690530
PS G:\360MoveData\Users\Lenovo\Desktop> cd "g:\360MoveData\Users\Lenovo\Desktop\" ; if ($?) { gcc Mondrian.c -o Mondrian } ; if ($?) { .\Mondrian }
cost 261ms time
LM is 0.739209
PS G:\360MoveData\Users\Lenovo\Desktop> cd "g:\360MoveData\Users\Lenovo\Desktop\" ; if ($?) { gcc Mondrian.c -o Mondrian } ; if ($?) { .\Mondrian }
cost 370ms time
LM is 0.659348
PS G:\360MoveData\Users\Lenovo\Desktop> cd "g:\360MoveData\Users\Lenovo\Desktop\" ; if ($?) { gcc Mondrian.c -o Mondrian } ; if ($?) { .\Mondrian }
cost 295ms time
LM is 0.849176
PS G:\360MoveData\Users\Lenovo\Desktop> cd "g:\360MoveData\Users\Lenovo\Desktop\" ; if ($?) { gcc Mondrian.c -o Mondrian } ; if ($?) { .\Mondrian }
cost 212ms time
LM is 0.829968
PS G:\360MoveData\Users\Lenovo\Desktop> cd "g:\360MoveData\Users\Lenovo\Desktop\" ; if ($?) { gcc Mondrian.c -o Mondrian } ; if ($?) { .\Mondrian }
cost 419ms time
LM is 0.667442
PS G:\360MoveData\Users\Lenovo\Desktop> cd "g:\360MoveData\Users\Lenovo\Desktop\" ; if ($?) { gcc Mondrian.c -o Mondrian } ; if ($?) { .\Mondrian }
cost 171ms time
LM is 0.953259
PS G:\360MoveData\Users\Lenovo\Desktop>
```

平均LM为0.756308，平均运行时间为302.8ms

大致可分析出，K越大，LM越大，泛化效果越差，递归深度也越浅从而使得运行时间越短。

选做部分

•Samarati算法可能会有很多解满足要求，调研并探究如何选择输出保证结果的可用性尽可能大，说说你的启发，(e.g.:选用合适的评价指标评价不同的输出)（存疑）

只需要修改一下算法

Find_vector

INPUT: Table $T_i = \mathsf{PT}[QI]$ to be generalized, anonymity requirement k, suppression threshold MaxSup, lattice VL_{DT} of the distance vectors corresponding to the domain generalization hierarchy DGH_{DT}, where DT is the tuples of the domains of the quasi-identifier attributes.

OUTPUT: The distance vector *sol* of a generalized table GT_{sol} that is a k-minimal generalization of $\mathsf{PT}[QI]$ according to Definition 4.3.

METHOD: Executes a binary search on VL_{DT} based on height of vectors in VL_{DT}.

1. $low := 0$; $high := height(\top, \mathsf{VL}_{DT})$; $sol := \top$
2. **while** $low < high$
   - 2.1 $try := \lfloor \frac{low+high}{2} \rfloor$
   - 2.2 $Vectors := \{vec \mid height(vec, \mathsf{VL}_{DT}) = try\}$
   - 2.3 $reach\_k := \mathtt{false}$
   - 2.4 **while** $Vectors \neq \emptyset \wedge reach\_k \neq \mathtt{true}$ **do**
     - Select and remove a vector *vec* from *Vectors*
     - **if** satisfies(vec, k, T_i,MaxSup) **then** $sol := vec$; $reach\_k := \mathtt{true}$
   - 2.5 **if** $reach\_k = \mathtt{true}$ **then** $high := try$ **else** $low := try + 1$
3. **Return** *sol*

把reach_k ≠ true这一循环条件消去，然后在每次泛化结束后即计算LM，并且和目前为止最优的LM进行比较，如果小于则更新最优的LM为当前的LM，如果等于则比较运行时间，如果此次运行时间更短，则同样更新，否则保持并记录最优的LM和对应泛化具体阶层。具体已经体现在Samarati.c的代码里面。

•Mondrian算法处理categorical（如Gender）

我选择QI = {Age, Gender}。对于Gender同样用int数组来存，即输入时，判断此时暂存数组，如果是"Female"，则存1，如果是"Male"，则存0。泛化之后的可能有"[00-00]"、"[00-01]"、"[01-01]"三种，分别对应"Male"、"*"、"Female"，输出之前转换回来即可。其他处理无异，具体已经体现在plus.c的代码里面。

plus.c使用说明

你需要更改的是第22行和第298行输入输出的文件路径，另外如果想测试不同的K，需要更改第六行define中K的值。

讨论与总结

通过本次实验让我更深一步了解了K匿名的本质，让我明白了K、MAXSUP与LM的关系，参考结果分析部分概括来说就是，**在合理范围内K越小、MAXSUP越大、对应LM越小**，当然要结合实际情况来设计K与MAXSUP。另一方面可以明显发现**Mondrian算法求得的LM与Smaratai算法求得的LM要更优**，原因是**前者分割的更加细致。**但总得来说，**Samarati算法更适合处理categorical类型，而Mondrian更适合处理数值类型。**我们在处理categorical类型时，也可用考虑像选做部分那样，对每一类数据都赋一个与众不同的数值，然后当作数值类型来使用Mondrian算法泛化。两种思路都给予了我不少的启发。