

订阅DeepL Pro以编辑此演示文稿。  
访问www.DeepL.com/Pro，了解更多信息。

**关于抄袭的重要信息**

在你开始在上海理工大学学习之前，你要意识到最重要的一点是 "抄袭 "的含义。

*剽窃是指把别人的工作或想法当作自己的工作或想法的做法。它是将他人的工作作为自己的工作进行虚假陈述。它是学术盗窃；严重违反了大学的荣誉准则，而后者是你有责任维护的。抄袭或任何其他作弊行为将被报告给学校领导，并将产生严重后果。避免任何形式的抄袭是为了你自己的利益。如果你抄袭，而且只是在后期才被揭发，这不仅会对学校造成不良影响，也会影响你的形象/职业机会。*

## 抄袭是学术不端行为，我们在上海理工大学非常严肃地对待它。在过去，我们有很多与抄袭有关的问题，尤其是新来的学生，所以在前期就把这个问题搞清楚很重要。

**你可能...**

* ......与你的同龄人讨论课程材料。
* ......一般性地讨论编程语言、某些功能或抽象的代码行。只要它与任何作业没有直接关系，而是以一般的、抽象的方式表述，这种讨论是可以接受的。
* ... 互相分享测试案例。
* ...在建立开发环境等方面相互帮助。

**你可能不...**

* ...阅读、拥有、复制或提交其他人（包括本课程或大学以外的人）的解决方案代码
* ......接受别人的直接帮助（即直接交流一些代码行，不管是视觉的、口头的还是书面的）!
* ...给别人以直接的帮助。帮助你的一个同伴，让他阅读你的代码，或者以书面或口头形式交流哪怕只是一部分的解决方案，都会有同样的后果。
* ......进入另一个人的账户，无论是否经过许可。
* ......将你的账户访问权交给其他学生。保护你的账户安全是你的责任，要经常注销，并选择一个安全的密码。在没有事先锁定和关闭自动登录功能的情况下，不要随便与其他学生分享你的电脑。不要让你的电脑不加锁，即使只是为了5分钟的休息。
* ......以小组形式工作。你们可以见面讨论有关材料，但任何关于作业的工作都要单独完成，并保持隐私。记住，你不能允许任何人甚至只是阅读你的源代码。

有了互联网，"粘贴"、"分享 "是很容易的操作。不要以为隐藏起来很容易，我们不会发现你，我们有同样容易使用的全自动智能工具，可以识别任何潜在的抄袭案例。也不要认为作为原作者会有什么不同。与他人分享原创解决方案与使用他人的作品一样不道德。

**CS100家庭作业7（2021年春季**

提交期限。

2021-06-0923:59

**问题1.热身活动。有理运算和CMake**

在这个热身问题中，你将在分离的头文件和源文件中实现一个有理数类，并使用CMake编译一个多文件项目。CMake是本问题中唯一需要的新知识，超出了上次作业（作业6）的范围。

所有的**有理数**都可以写成商p/q，其中p是一个整数，q是一个正整数。两个有理数的加、减、乘、除的结果仍然是一个有理数。[1](#_bookmark0) 你需要为有理数类实现这些运算，以及其他一些函数，如本模板中所示。

类有理数

{

公众。

// 构造函数 Rational(); Rational(int value);

Rational(int p, unsigned int q)。

// 赋值运算符

Rational& operator=(const Rational& that);

// 算术运算符

Rational operator+(Rational that) const; Rational& operator+=(Rational that); Rational operator-(Rational that) const; Rational& operator-=(Rational that); Rational operator\*(Rational that) const; Rational& operator\*=(Rational that); Rational operator/(Rational that) const; Rational& operator/=(Rational that) 。

// 比较运算符：等于和小于 bool operator==(Rational that) const;

bool operator<(Rational that) const;

// 输出

friend std::ostream& operator<<(std::ostream& os, const Rational& number); private:

int m\_numerator;

无符号int m\_denominator。

};

* 这个类有3个构造函数。它们分别创建一个0、值和p/q的有理数。
  + 如果在第三个构造函数中q为0，你应该打印一个信息

"ERROR: initializing with a denominator of 0!" 换一行，并将这个有理数设为0。

* + 你创建的有理数应该被简化。例如，行

有理数r(-6, 10); 应该把数字初始化为-3/5。

1 在数学的意义上，具有这种性质的集合（所有有理数的集合）被称为场。

* 赋值运算符是不言自明的。
* 算术运算符也是不言自明的。唯一需要注意的是。
  + 当除以0的情况发生时，你应该打印一条信息 "ERROR：除以0！"并换行，你可以将结果设置为任何值。事实上，这种情况不会出现在Autolab测试或问题2的使用中。如果在问题2中你看到你的错误信息，你的计算可能是错误的。
  + 这些运算符的任何结果都应该被简化。例如，表达式

有理数(1, 3)+有理数(1, 6)应该求值为1/2。

* + 注意运算符+和运算符+=的区别。 提示：运算符+=

修改数字本身并返回一个引用。

* 在下一个问题中，至少需要两个比较运算符==和<。
* std::ostream的操作者<<应该将数字以p/q的形式打印到os中，p代表分子，q代表分母。
  + 如果数字是一个整数，你应该打印成一个整数，即省略"/q "部分。

关于多文件和CMake的指导。

* 你的 "Rational.hpp "文件应该只包含Rational类的定义和它的函数。
* 你的 "Rational.cpp "文件应该包含你的Rational类中的函数的实现。
* 你应该写一个 "CMakeLists.txt"，在其中你应该。
  + 将最小版本限制设置为 "VERSION 2.8"。
  + 创建一个名为 "理性 "的项目。
  + 为你的计算机生成一个构建系统，使其将 "rational.cpp "与 "main.cpp "文件一起编译（你可以在调试时创建这个文件，但不要提交，我们会在分级时提供一个 "main.cpp"。），并创建一个名为 "rational "的可执行文件。

### 如果CMake在你的电脑上有不同的表现（例如，创建Visual Studio解决方案）也没关系。在Autolab服务器（Linux）上，运行cmake将生成一个 "Makefile"。然后运行make将构建可执行的 "理性"。

* 你的 "reason.hpp "应该有一个适当的**包含防护**。也就是说，如果你的代码在编译时有以下内容的 "main.cpp"，它就必须被编译。

#include "rational.hpp" #include "rational.hpp" int main(){ return 0; }

提交。

请向Autolab提交一个包含三个文件的tarball（\*.tar）文件。"rational.hpp", "rational.cpp", 和 "CMakeLists.txt"。这个tarball文件的名称可以是任意的。

如果你在制作tarball时遇到困难，你可以下载任何存档软件。(推荐：[Bandizip](https://www.bandisoft.com/bandizip/))

# 问题2.模板化的高斯-乔丹消除

在这个问题中，你将编写一个Matrix的类模板，以创建双倍和三倍的矩阵。

有理，并进行高斯-乔丹消除（也称为高斯消除）。

[高斯-乔丹消除](https://en.wikipedia.org/wiki/Gaussian_elimination)法是一种解决线性方程组的方法，它通过行运算将其转化[为行梯形形式](https://en.wikipedia.org/wiki/Row_echelon_form)。这种方法也可以用来计算矩阵的秩，平方矩阵的行列式，以及可逆矩阵的逆。[2](#_bookmark1) 我们将提供一个详细的算法的伪代码，这样，即使你不熟悉线性代数，也可以实现。

现在，让我们来看看模板化类Matrix的结构。

模板< typename T>类 矩阵

{

公众。

矩阵()。

Matrix(unsigned int rows, unsigned int cols, const T& value = T()); T& operator()(unsigned int r, unsigned int c)。

void Print()。

void GaussJordan(); private:

// 你的实现在这里。

};

* 你的矩阵应该以向量的向量来存储其数据。
* 第二个构造函数用给定的行和列创建一个矩阵，每个条目都初始化为给定的值。注意，T()的一个默认参数，即空构造函数，被赋值，所以当你声明一个带有默认条目的矩阵时，可以省略它。
* 操作符()用于分配给矩阵中的一个特定条目。
* Print()方法将矩阵打印到std::cout，格式简单如下。
  + 对于每个条目，将其打印到std::cout，后面是一个空格""。
  + 在每一行之后，打印一个换行。允许每行以尾部空格结束，最后一行之后是新行。
  + 不需要setw()或setprecision()。
* 最后是GaussJordan()方法。这个方法在原地进行Gauss-Jordan消除（改变矩阵的内容），所以矩阵是行梯形形式（而不是减行梯形形式）。由于得到的矩阵不是唯一的，你应该按照给定的伪代码来实现该算法。

2 [高斯消除法 - 维基百科](https://en.wikipedia.org/wiki/Gaussian_elimination)

h :=1 /\**初始化支点行* \*/

k := 1 /\* *枢轴列的初始化* \*/

**而**h≤m**和**k≤n

/\* *找到第k个支点* \*/

/\* *记录第h行后第k列的最大值和最小值* \*/ i\_max := argmax (i = h ... m, A[i, k])

i\_min := argmin (i = h ... m, A[i, k])

**如果**A[i\_max, k] = 0

/\* max = 0，*将max与min交换* \*/ i\_max := i\_min

**如果**A[i\_max, k] = 0

/\* max = min = 0，*这一列没有支点，传到下一列* \*/ k := k+1

**否则**

**交换行**(h, i\_max)

/\* *对枢轴以下的所有行进行处理。*\*/

**对于**i = h + 1 ... m。

f := A[i, k] / A[h, k]

/\* *用零填充枢轴列的下半部分。*\*/ A[i, k] := 0

/\* *对当前行的所有剩余元素进行处理。*\*/

**为**j = k + 1 ... n。

A[i, j] := A[i, j] - A[h, j] \* f

/\* *增加中枢行和列* \*/ h := h + 1

k := k + 1

* + 请注意，突出显示的部分已被修改：原来的算法选择绝对值最大的行作为枢轴，但问题1中的Rational类不能使用与double相同的方法来计算绝对值，所以我们需要同时检查max和min，优先考虑max。

测试。

我们在给定的文件 "tests.cpp "中对你的矩阵提供了两个简单的测试函数。你的程序的预期输出已包含在注释中。

关于多文件和CMake的指导。

* 由于模板类的声明和实现在头文件和源文件中是很棘手的（因为不同类型的成员函数是在模板类型下生成和编译的），你应该把所有的代码写在一个名为 "matrix.hpp "的文件中。
* 你应该写一个 "CMakeLists.txt"，在其中你应该。
  + 将最小版本限制设置为 "VERSION 2.8"。
  + 创建一个名为 "矩阵 "的项目。
  + 为你的计算机生成一个构建系统，使其将 "rational.cpp "与 "main.cpp "文件一起编译（你可以在调试时创建这个文件，但不要提交，我们会在分级时提供一个 "main.cpp"），并创建一个名为 "matrix "的可执行程序。你不需要同时编译 "matrix.hpp"，因为它是一个头文件，它将被包含在 "main.cpp "中。

### 如果CMake在你的电脑上有不同的表现（例如，创建Visual Studio解决方案）也没关系。在Autolab服务器（Linux）上，运行cmake将生成一个 "Makefile"。然后运行make将构建可执行的 "矩阵"。

* 你的 "matrix.hpp "应该有一个适当的**包含防护**。也就是说，如果你的代码在编译时有以下内容的 "main.cpp"，它就必须被编译。

#include "matrix.hpp" #include "matrix.hpp" int main(){ return 0; }

* 你的 "matrix.hpp "不应该#include "rational.hpp"。

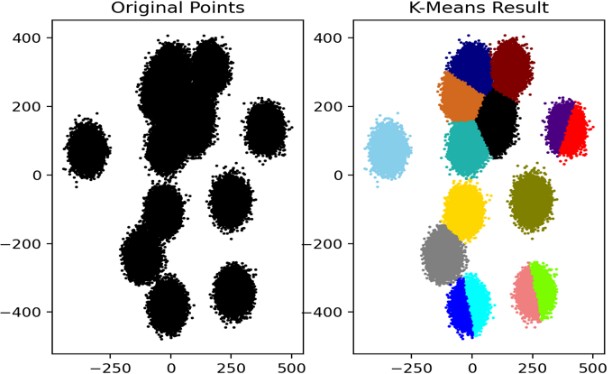
提交。

请向Autolab提交一个包含四个文件的tarball（\*.tar）文件。"matrix.hpp"、"reason.hpp"、"reason.cpp "和 "CMakeLists.txt"。这个tarball文件的名字可以是任意的。

# 问题3.平行K-Means聚类法

[*K-Means*聚类算法](https://en.wikipedia.org/wiki/K-means_clustering)是数据科学中最直接的聚类算法之一。它的目的是解决*矢量量化*问题。

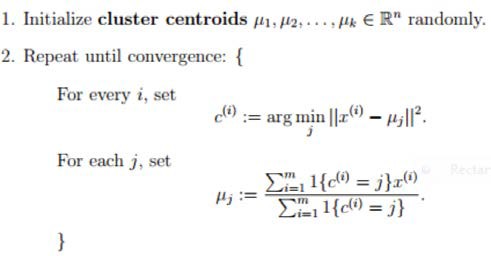
我们将在一个二维数据集上处理K-Means。如下图左侧所示，给定N个点（坐标）和一个数字K，你需要将这些点分为K个聚类。例如，下图右侧显示了一个示例数据集的K-Means聚类结果，数据样本点的数量N=300,000，所需的聚类数量K=15（每种颜色表示一个聚类）。



这就是K-Means算法的程序。

1. 每个中心点c代表一个群组的中心（即平均值）。
2. 对于每个样本点p和每个中心点𝑐𝑐𝑖，计算p和𝑐𝑐𝑖之间的距离：d( p, 𝑐𝑖 ) 。
3. 将每个样本点p分配到集群x，其中d(p, 𝑐𝑐x)是所有d(p, 𝑐𝑐𝑖 )s中最小的。
4. 更新每个聚类i的中心点𝑐𝑐𝑖为该聚类内所有点𝑝𝑝𝑖的当前平均值。
5. 转到2.并重复，除非在这次迭代中没有发生更新（即收敛）。

这是K-Means算法的伪代码，而它可能很难理解。



在这个作业中，我们为你提供了一个该算法的框架。在这个框架中，每个数据点都是一个Point结构的实例。我们为你提供了一些结构点的函数。你只需要实现这3个函数。

* double Point::Distance(Point& Other)
  + 计算该点与 "其他 "之间的欧几里得距离
* std::ostream& operator<< (std::ostream& os, Point& pt)
  + 将Point&pt的内容输出到os，格式是：首先输出pt.x，然后输出一个空格（""），再输出pt.y。注意你不需要开始一个新行（即不会打印"/n"）。例如，你应该用 "3 4.5 "来输出Point(3, 4.5)。
* std::istream& operator>> (std::istream& is, Point& pt)
  + 从std::istream&中扫描Point&pt的 "x "和 "y"，输入的是两个由一个空格（""）分隔的双数。

对于K-Means类，你需要实现构造函数和函数Kmeans::Run。

* Kmeans::Kmeans(const std::vector< Point>&, const std::vector< Point>& initialCenters)
  + K-Means类的构造函数，point是一个包含数据样本点坐标的向量，initialCenters包含初始中心的坐标。
* std::vector< index\_t> Kmeans::Run(int maxIterations)
  + 运行K-Means算法。返回所有数据点的索引（颜色）。例如，如果你有1000个数据点，你想把它们分为5个聚类。那么，这个函数的返回值是一个大小为1000的向量，表示每个点属于哪个簇。返回的向量的值是{0, 1, 2, 3, 4}。例如，result[9]=3意味着point[9]被分到了第3簇）。
  + 聚类循环最多运行maxIterations次。如果它没有在

maxIterations次数，就从循环中断开并返回当前结果。

* + **重要的是：你必须使用多线程来加速它**。正如课堂上所讨论的，我们可以使用std::thread来加快程序的速度，通过多

穿针引线。请注意，点和中心的距离计算是单独的，所以很容易并行计算。我们建议你在Autolab上使用**4个线程**，使用太多的线程会使它变得更慢。

这里有6个文件提供给你。你需要修改 "kmeans.cpp "和 "kmeans.hpp "中的代码。不要修改 "main.cpp "中的任何代码，否则你的代码可能不被OJ所接受。你可以使用 "CMakeLists.txt "来构建这个项目，使用 "generate.py "来生成测试和调试的样本，使用 "visualize.py "来显示聚类结果。

希望你已经在自己的计算机上安装了Python3。对于这个作业，你可能需要通过下面的命令来安装依赖项。

python -m pip install numpy sklearn matplotlib

之后，你可以通过这3条命令测试你的程序（假设你的可执行文件是"./kmeans"）。

python generate.py in.txt

./kmeans out.txt < in.txt python visualize.py out.txt