# 实践1：实现快速算法

一、算法思想说明：

快速排序代码是在暑假的project：并行快速排序上写的，所以只对其中一些代码进行了删减。

快速排序它的基本思想是：通过一次排序将数据分割成独立的两部分，其中一部分的所有数据都比另外一部分的所有数据都要小，然后再按照此方法对这两部分数据分别进行快速排序，整个排序过程可以递归并行处理，以此达到整个数据变成有序序列。

其过程主要分为三个阶段:

1) 在待排序的序列中找出一个枢轴;

2) 根据枢轴将待排序的序列划分成两个不相交的子序列，其中一个子序列里的元素全部不小于枢轴，另一个子序列里的元素全部不大于枢轴;

3) 分别对两个子序列递归进行快速排序，直到划分出的子序列的长度为1。

二、优化内容：

1、对于枢轴的选取：

由于采用递归来进行排序，当序列的长度较小时，频繁的递归操作也会影响排序的性能。使用“三点取中”的方法，用序列中的头、尾和中点这三个关键字的中间值来作为枢轴，有效地避免了快速排序在最坏情况下的性能恶化。

2、排序方法的选取：

快速排序的递归操作在序列长度较小时会影响排序的效率，应该使用其他非递归算法来处理小序列。直接插入排序是对小序列进行处理的有效方法，为此设置一个阈值M，当快速排序过程中划分出的子序列的长度小于或等于M 时，不再递归调用快速排序而使用直接插入排序，以避免对小序列排序时的频繁递归，这里的阈值为16，quick\_sort中若(last-first)<= 16就return执行直插。

三、运行结果：

输入：300个随机数

输出：背景图案

描述已自动生成

# 实践2：实现Dijkstra算法

1. 算法说明

Dijkstra计算图G中的最短路径时，需要指定起点s(即从顶点s开始计算)。此外，引进两个集合S和U。S的作用是记录已求出最短路径的顶点(以及相应的最短路径长度)，而U则是记录还未求出最短路径的顶点(以及该顶点到起点s的距离)。

初始时，S中只有起点s；U中是除s之外的顶点，并且U中顶点的路径是"起点s到该顶点的路径"。然后，从U中找出路径最短的顶点，并将其加入到S中；接着，更新U中的顶点和顶点对应的路径。 然后，再从U中找出路径最短的顶点，并将其加入到S中；接着，更新U中的顶点和顶点对应的路径。 ... 重复该操作，直到遍历完所有顶点。

二、运行结果：

输入：6\*6的邻接矩阵

graph[6][6]={

{0, 10, 20, 0, 0, 0},

{10, 0, 0, 50, 10, 0},

{20, 0, 0, 20, 33, 0},

{0, 50, 20, 0, 20, 2},

{0, 10, 33, 20, 0, 1},

{0, 0, 0, 2, 1, 0}};

输出：图片包含 表格

描述已自动生成

# 实践3：实现最长公共子序列算法

1. 算法说明

给定两个字符串 text1 和 text2，返回这两个字符串的最长公共子序列的长度。如果不存在公共子序列，返回0。一个字符串的 子序列是指这样一个新的字符串：它是由原字符串在不改变字符的相对顺序的情况下删除某些字符（也可以不删除任何字符）后组成的新字符串。例如，"ace" 是 "abcde" 的子序列，但 "aec" 不是 "abcde" 的子序列。两个字符串的公共子序列是这两个字符串所共同拥有的子序列。

二、算法改进

原问题应该是只需要输出最长公共子序列的长度，这里也输出了具体的最长子序列，大致方法是通过标记，自底向下对子序列进行输出。

三、运行结果

输入：string X= "AGGTAB";

string Y = "GXTXAYB";

输出：图形用户界面, 应用程序

描述已自动生成

# 实践4：实现PSO、GA、SGD算法

1. 算法说明

一般来说，机器学习中的反向传播包含两个步骤：

前馈通道：将训练数据集通过网络并记录神经网络的输出，并计算网络的误差。

向后传播：误差信号通过网络传回，神经网络的权重使用梯度下降进行优化。

全局优化算法通过将神经网络编码为权重矢量来完成，每个权重表示神经网络中连接的权重。我们使用元启发式搜索算法来训练神经网络。这种技术在深度神经网络中效果不佳，因为它会使矢量变得太大。

此次训练用的是mnist的数据集，建立神经网络模型进行训练，损失度和准确度来衡量算法的优越性。

1. SGD算法

初始化：随机选取取值范围内的任意数

循环操作：

计算梯度；

修改新的变量；

判断是否达到终止：如果前后两次的函数值差的绝对值小于阈值，则跳出循环；否则继续；

输出最终结果；

1. PSO算法

为了训练使用PSO的神经网络，每个神经网络都被表示为一个权重向量，并根据其从全球最佳粒子及其个人最佳位置进行调整。

在完成训练数据集的一个前馈通过之后，适应度函数被计算为重建的神经网络的总和平方误差，这种方法主要考虑的是权重更新的速度。这是因为如果权重调整过快，神经网络的总和平方误差将停滞并且不会发生学习。

总的来说：粒子群中每个粒子的位置表示BP神经网络当前迭代中权值的集合，每个粒子的维数由网络中起连接作用的权值的数量和阈值个数决定，以给定训练样本集的神经网络输出误差作为神经网络训练问题的适应度函数，适应度值表示神经网络的误差，误差越小则表明粒子在搜索中具有更好的性能，粒子在权值空间内移动搜索使得网络输出层的误差最小，改变粒子的速度也就是更新网络权值，以减少均方误差。

1. 遗传算法

为了使用遗传算法训练神经网络，首先构建代表神经网络的矢量群体。然后，将三个遗传算法应用于该群体，以发展更好的神经网络。这三个遗传算法是：

选择：使用一次前馈通过后计算的每个网络的总和平方误差，我们对神经网络的总体进行排序。总体中最高的x％被选择为“生存”到下一代，并用于交叉。

交叉：总体基因的顶部x％被允许相互交叉，这个过程形成“后代”。在上下文中，每个后代都将代表一个来自两个“父”神经网络的权重的新神经网络。

突变：该遗传算法需要保持群体的遗传多样性。选择一小部分进行突变，这些神经网络中的一些权重将在特定范围内随机调整。

二、运行结果

输入：minist数据集

输出：





GA:

图表, 折线图

描述已自动生成

图表, 折线图

描述已自动生成

分析原因：GA的训练效果不佳可能是因为数据集过大或者是参数不对导致的，但启发算法都没有随机梯度下降算法好。这是因为使用元启发式算法在深度神经网络中会使矢量变得太大。