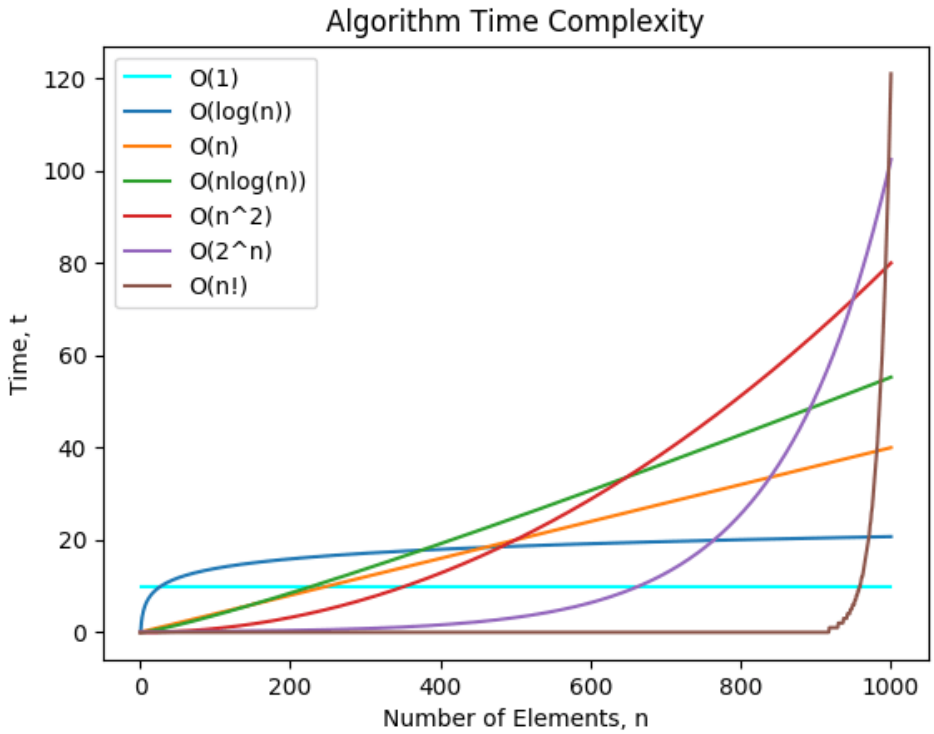
编 程 日

编程题 1：排序算法竞赛

- 主办方建立抽象类 Benchmark
  - 建立好不同规模的乱序数据集(10^3 到 10^8)
  - 写好统计时间性能的方法 long time(){...}
  - 将 runAlgorithm()方法设为 abstract，交给参赛方实现
- 参赛方建立实现类 BenchmarkImpl
  - 实现至少四种排序算法(每个算法可以对应一个单独的 BenchmarkImpl)
  - 对不同规模的乱序数据集进行排序，并使用 time()方法计算时间性能
  - 写 validate()方法，对排序后的数据集进行正确性验证
  - 对于每个排序算法，画出时间性能随数据规模变化的图(建议采用 excel 画)
  - 将 java.util.Arrays.sort()方法作为基线方法，画出时间性能图
  - 将三种算法(自己写的两种和 java 提供的基线方法)的时间性能图画在一张图上
  - 时间性能图类似于下图，以数据规模为横坐标、时间耗费为纵坐标，建议使用指数坐标系，不要求对算法的时间复杂度进行分析



编程题 2：基于 MVC 架构的 ATM 机

- 构建三个类，作为 MVC 架构的三层
  - User 类，作为底层的模型层，存储用户信息(用户名、姓名、性别、密码、账号创建时间)
  - Bank 类，作为中间的控制层，完成用户要求的查询余额、存钱、取钱等操作
  - ATM 类，作为上层的视图层，向用户呈现菜单，接收用户的输入，并向用户显示正确或错误信息
- 功能要求：
  - 查询余额：int queryBalance()
  - 存钱：void deposit(int amount),注意需要做 amount 的合法性验证
  - 取钱：void withdrawl(int amount),注意需要做 amount 的合法性验证，且不可以取超过余额的钱
- 菜单要求：
  - 创建新用户
    - 请输入姓名

        - 请输入密码
    - 登录
        - 请输入姓名
        - 请输入密码
            - 您好，X 先生/女士！欢迎来到 XXX 银行的 ATM！
            - 请输入您的操作：
            - 存钱
            - 取钱
            - 查询余额
            - 退出当前用户
    - 退出系统
- 单用户、非持久化的处理方法
    - 简化菜单，无需创建新用户，也无需进行用户切换
    - 固定一个用户 admin，其默认密码为 123456，允许用户修改密码
- **多用户和持久化的处理方法(选做)*
    - 使用 HashMap 进行多用户信息的存储
    - 在退出系统时，使用 ObjectOutputStream 将所有用户的信息导出
    - 在进入系统时，使用 ObjectInputStream 将所有用户的信息导入
- **对每个类进行 JUnit Test(选做)*