**Python练习研究报告**

**小组信息**

|  |  |  |  |
| --- | --- | --- | --- |
| 学号 | 姓名 | 邮箱 | Python练习完成情况 |
| 181250168 | 薛人玮 | 181250168@smail.nju.edu.cn | 106/200 |
| 181250083 | 林希澄 | 181250083@smail.nju.edu.cn | 199/200 |
| 181250119 | 任一丁 | 181250119@smail.nju.edu.cn | 200/200 |

**小组人数：**3人

**组员分工职责：**

薛人玮：负责数据提取和代码风格评价的代码编写；负责前后端连接件的编写。

林希澄：负责前端页面，后端代码架构和代码相似度/面向用例程度的代码编写。

任一丁：负责对代码的时间/空间复杂度判定的代码编写；负责人工测试和数据统计调查。

**研究问题**

**本项目研究的问题为：**如何评价单一学生在学生样本集当中的编程能力高低；如何评价单一学生的某个代码样本，在学生样本集所提供的代码样本集中的优劣程度。

**本项目应用的场景：**用于小体量的学生python算法练习评价。

**代码开源地址**

**研究方法**

**一 学生编程能力评价**

**原理：**分析每个代码样本的各项代码评价指标（时间复杂度、空间复杂度、代码风格等），列出这些指标，并与同一题目的样本集作比较，给出评价。通过对单一学生所有代码评价指标的统计，可以得出学生在样本集中的相对水平高低。

**限制：**大部分复杂度分析方法的代码根据python语言的特点编写，并且使用纯文本分析方法，不涉及语法树的构建以及语义分析。因此该方法较适用于小型的python算法题（代码行数<50）评价，在逻辑结构复杂、编写方法较多的代码样本中方法判别的错误率会显著升高。

**评测：**对提交的42800+份代码进行**等距抽样，**样本距离为100，共抽取428份代码样本进行人工评测。其中符合人工评测期望的机器评测样本数量为402份，机器判别有效率可达94%左右。考虑到设计逻辑上的部分缺陷，实际的有效率会比评测的有效率略低一些。相关统计详情见附录中的EXCEL表格以及对人工评测标准的相关说明。

**分析思路：**

1. 复杂度分析采用了**文本分析**方法，针对python特殊的缩进式语法结构构建**缩进列表**（缩进树），并利用该列表判断嵌套结构。复杂度分析使用了定位算法题中常见的逻辑结构保留字（for、while），以及常用的算法复杂度语言特征（二分查找、简单尾递归等），使得在小型算法题中可以较为准确地判断循环层数和单一方法的复杂度。

该分析方法的缺陷为：方法会忽略import包产生的复杂度；方法在多个方法嵌套调用的情况下（如间接递归），会忽略来自其他方法的复杂度；分析方法的过程中，循环销毁的变量可能会被误判为递归压栈导致空间复杂度上升。

1. 代码风格评价？

**二 代码相似度检测**

**三 学生面向用例情况分析**

**案例分析**

**附录**

**附录1：**《抽样人工评测表单》

表单详情见代码目录下record\_data/code\_complexity\_sample.xlsx

**附录2**：《人工评测标准》

与机器评测逻辑大致相同，给出时间/空间复杂度判断。

时间复杂度：

1. 逻辑中含有循环关键字的，每一层记为n
2. 逻辑中含有二分法（多分法）的，每一层记为log\_n
3. 逻辑中含有常见递归的（如斐波那契数列），按照常见递归表单\*标记

空间复杂度：

1. 在循环结构中对列表项进行添加的，每一层记为n
2. 在二分法（多分法）中对列表项进行添加的，每一层记为log\_n
3. 利用循环结构对列表进行声明的，每一层记为n
4. 在递归中声明列表并循环添加内容的，每一层记为n

**附录3：**《常见递归表单》

T(n)=T(n/2)+O(1) T(n)=O(log\_n)  
T(n)=T(n-1)+O(1) T(n)=O(n)  
T(n)=2\*T(n/2)+O(1) T(n)=O(n)  
T(n)=2\*T(n/2)+O(n) T(n)=O(n\*log\_n)  
T(n)=2\*T(n/2)+O(n\*log\_n) T(n)=O(n\*log^2\_n)  
T(n)=T(n-1)+O(n) T(n)=O(n^2)  
T(n)=2\*T(n-1)+O(1) T(n)=O(2^n)  
T(n)=T(n-1)+T(n-2)+O(1) T(n)=O(2^n)  
其中计算复杂度时，递归式中若包含对同文件其他函数的调用，将其视为O(n)