**代码搜索引擎设计文档**

二○一八年七月

目录

[1、设计板块 3](#_Toc520489147)

[2、总体架构 3](#_Toc520489148)

[3、详细设计 4](#_Toc520489149)

[3.1 数据库设计 4](#_Toc520489150)

[3.1.1 E-R图 4](#_Toc520489151)

[3.1.2 数据表设计 4](#_Toc520489152)

[3.2 爬取技术 6](#_Toc520489153)

[3.2.1 爬取网页 6](#_Toc520489154)

[3.3 正文搜索 11](#_Toc520489155)

[3.3.1 功能描述 11](#_Toc520489156)

[3.3.3 ES索引类比 11](#_Toc520489157)

[3.3.4 ES检索文档 11](#_Toc520489158)

[3.4 代码搜索引擎 14](#_Toc520489159)

[3.4.1 词法分析 14](#_Toc520489160)

[3.4.2 语法分析 14](#_Toc520489161)

[3.4.3 分析工具 15](#_Toc520489162)

[3.4.4 代码相似性度量 15](#_Toc520489163)

[3.5 Online Judge 判题模块 18](#_Toc520489164)

[3.5.1 判题流程 18](#_Toc520489165)

[3.5.2 安全性保证 19](#_Toc520489166)

# 1、设计板块

1）数据库设计

2）全文搜索引擎设计

3）代码搜索引擎设计

4）OJ评测设计

# 2、总体架构

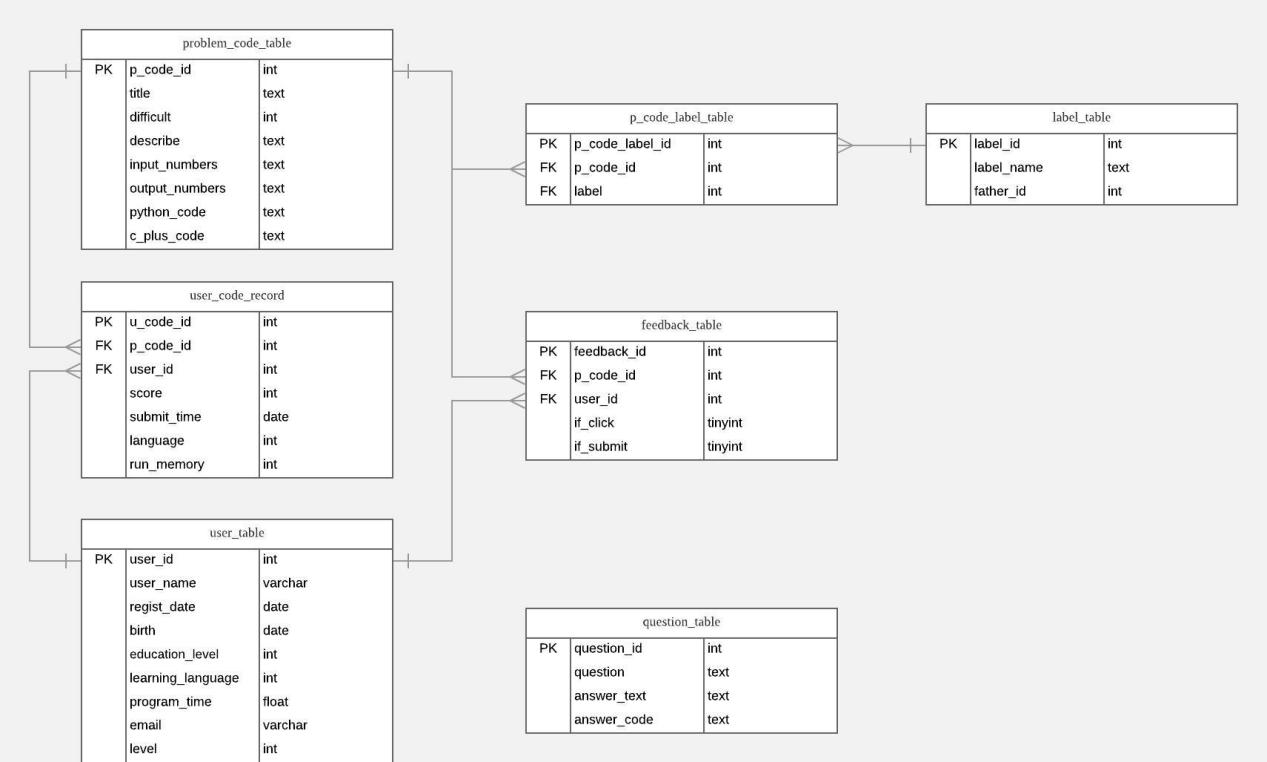
**2.1 系统功能表**

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 功能模块 | | 功能描述 |
| 1 | 用户管理 | 用户注册 | 用户注册 |
| 2 | 用户登录 | 用户登录 |
| 3 | 用户注销 | 用户注销 |
| 4 | 用户信息管理 | 信息修改 |
| 5 | 搜索功能 | 文本搜索 | 普通搜索 |
| 6 | 高级搜索 |
| 7 | 代码搜索 | 代码搜索 |
| 8 | OJ评测 | 代码评测 | Python代码评测 |
| 9 | C++代码评测 |

# 3、详细设计

## 3.1 数据库设计

### 3.1.1 E-R图



### 3.1.2 数据表设计

① 表名：problem\_code\_table

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 列名 | 数据类型 | 长度 | 主键 | 外键 | 允许空 |
| P\_code\_id | Int |  | 是 |  | 否 |
| Title | Text |  |  |  | 是 |
| Difficult | Int |  |  |  | 是 |
| describe | Text |  |  |  | 是 |
| Input\_numbers | Text |  |  |  | 是 |
| Output\_numbers | Text |  |  |  | 是 |
| Python\_code | Text |  |  |  | 是 |
| C\_plus\_code | Text |  |  |  | 是 |

② 表名：label\_table

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 列名 | 数据类型 | 长度 | 主键 | 外键 | 允许空 |
| Label\_id | Int |  | 是 |  | 否 |
| Label\_name | varchar |  |  |  | 否 |
| Father\_id | int |  |  |  | 否 |

③ 表名：p\_code\_label\_table

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 列名 | 数据类型 | 长度 | 主键 | 外键 | 允许空 |
| Pl\_id | Int |  | 是 |  | 否 |
| P\_code\_id | Int |  |  | 是 | 否 |
| Label\_id | int |  |  | 是 | 否 |

④ 表名：user\_table

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 列名 | 数据类型 | 长度 | 主键 | 外键 | 允许空 |
| User\_id | int |  | 是 |  | 否 |
| User\_name | Varchar |  |  |  | 否 |
| Regist\_date | Date |  |  |  | 否 |
| Birth | Date |  |  |  | 否 |
| Education\_level | Int |  |  |  | 否 |
| Learning\_language | Int (1,2,3) |  |  |  | 否 |
| Programm\_time | Float |  |  |  | 否 |
| Email | Varchar |  |  |  | 是 |
| Level | Int |  |  |  | 否 |

⑤ 表名：user\_code\_record

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 列名 | 数据类型 | 长度 | 主键 | 外键 | 允许空 |
| Ucode\_id | Int |  | 是 |  | 否 |
| P\_code\_id | Int |  |  | 是 | 否 |
| User\_id | int |  |  | 是 | 否 |
| Score | Int |  |  |  | 否 |
| Submit\_time | Date |  |  |  | 否 |
| Language | Int |  |  |  | 否 |

⑥ 表名：feedback\_table

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 列名 | 数据类型 | 长度 | 主键 | 外键 | 允许空 |
| Feedback\_id | Int |  | 是 |  | 否 |
| User\_id | Int |  |  | 是 | 否 |
| P\_code\_id | Int |  |  | 是 | 否 |
| If\_click | Tinyint |  |  |  | 否 |
| If\_submit | Tinyint |  |  |  | 否 |

⑦ 表名：question\_table

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 列名 | 数据类型 | 长度 | 主键 | 外键 | 允许空 |
| Question\_id | int |  | 是 |  | 否 |
| Question | text |  |  |  | 否 |
| Answer\_text | text |  |  |  | 是 |
| Answer\_code | text |  |  |  | 是 |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

## 3.2 爬取技术

针对python、c++代码Scrapy，Python开发的一个快速、高层次的屏幕抓取和web抓取框架，用于抓取web站点并从页面中提取结构化的数据。

### 3.2.1 爬取网页

#### 3.2.1.1 南阳理工学院OJ

题目、一个测试样例、难度、分类（考察知识点）

1340道题

只判断c/c++/java

<http://acm.nyist.edu.cn/JudgeOnline/problemset.php>







#### 3.2.1.2 Python2.7 100例

题目、源代码

<http://www.runoob.com/python/python-100-examples.html>

#### 3.2.1.3 c＋＋经典编程题汇总

题目、源代码

100道

<https://blog.csdn.net/qq_36864672/article/details/76037595>

#### 3.2.1.4 杭州电子科技大学online judge

题目、一个测试数据、相关题id推荐

5200道题



#### 3.2.1.5 蓝桥杯习题

题目、程序代码（90%C,部分C++和java）、测试数据（多组）

200道

90%习题使用C语言解答，部分使用C++或者Java

https://blog.csdn.net/rodestillfaraway/article/details/50529597

题目在网页上，测试样例在压缩包中

#### 3.2.1.6 CSDN C＋＋经典编程题汇总

题目、程序代码

100道题

<https://blog.csdn.net/qq_36864672/article/details/76037595>

#### 3.2.1.7 CSDN刷题汇总python版

题目，代码，知识点(字符串、动态规划），分析

35道

https://blog.csdn.net/dongrixinyu/article/details/78775057

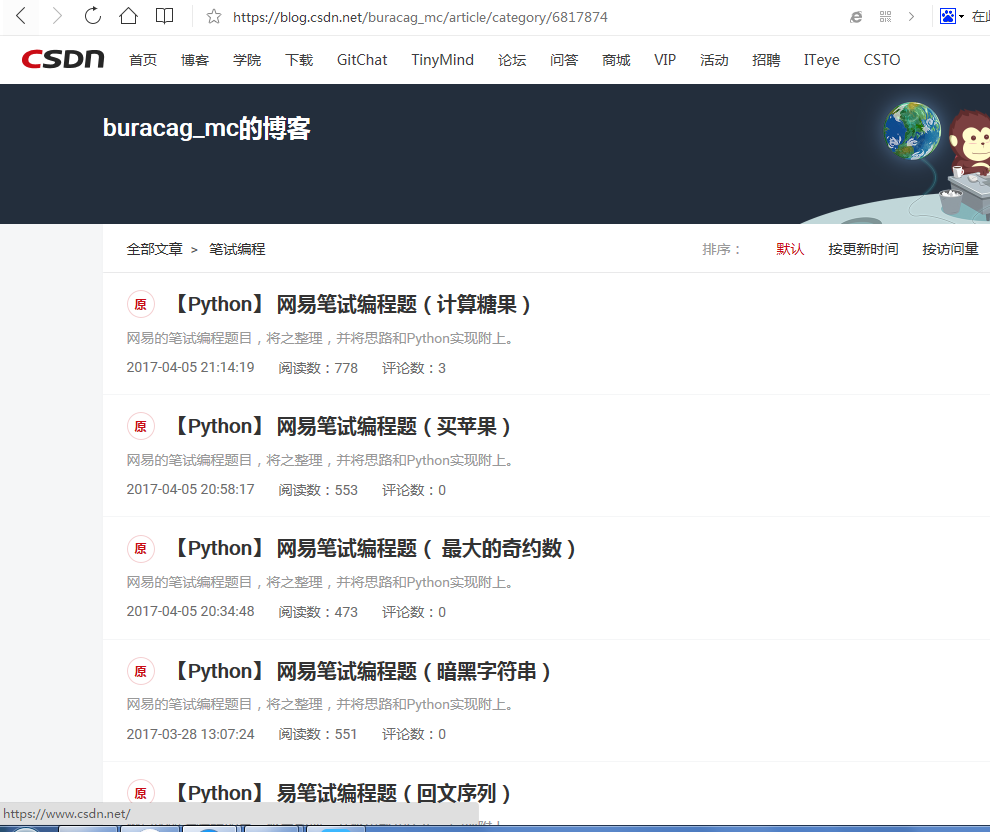


#### 3.2.1.8 CSDN网易笔试编程题python实现

题目、输入/输出例子、源代码

9道

<https://blog.csdn.net/buracag_mc/article/category/6817874>



#### 3.2.1.9 NOIP历年题目(全国青少年信息学奥林匹克联赛

题目、输入/输出格式、一个测试样例、答案代码（C）

<https://www.cnblogs.com/shenben/category/840423.html>

## 3.3 正文搜索

正文搜索模块的主要功能是在用户提交字符串题面信息，完成相关题的搜索。系统会通过搜索引擎查找服务器上的相关题，传给用户进行选择。

### 3.3.1 功能描述

普通查询：通过向搜索框输入字符串(题面)完成相关题搜索，并可通过事物的这些属性（例如：题的难度、语言类型、知识点）不断筛选过滤搜索得到的题目，按照相关性大小排序。

高级查询：按照表单要求输入标题、题面、知识点、语言、难度(可为空)进行检索,按照相关性大小排序。

**3.3.2 搜索引擎选型**：

**Elasticsearch (基于Apache Lucene(TM)的开源搜索引擎)**

特点： 分布式的实时文件存储，每个字段都被索引并可被搜索;

分布式的实时分析搜索引擎;(近实时)

可以扩展到上百台服务器，处理PB级结构化或非结构化数据;(集群处理)

**ES作为搜索引擎常用查询**：

全文本查询：针对文本类型的数据；

字段级别查询(结构化查询): 针对结构化数据，如数字、日期等

### 3.3.3 ES索引类比

在Elasticsearch中，文档归属于一种类型(type),而这些类型存在于索引(index)中，简单的对比图来类比传统关系型数据库：

Relational DB -> Databases -> Tables -> Rows -> Columns

Elasticsearch -> Indices -> Types -> Documents -> Fields

默认情况下，文档中的所有字段都会被索引（拥有一个倒排索引），可被搜索。

### 3.3.4 ES检索文档

我们只要执行HTTP GET请求并指出文档的“地址”——索引、类型和ID既可。根据这三部分信息，我们就可以返回原始JSON文档。

**简单搜索：GET /megacorp/employee/\_search?q=last\_name:Smith**

megacorp索引和employee类型，在结尾使用关键字\_search来搜索姓氏中包含‘Smith’的数据。

使用DSL语句查询:DSL(Domain Specific Language特定领域语言)以JSON请求体的形式出现。

GET /megacorp/employee/\_search

{

"query" : {

"match" : {

"last\_name" : "Smith"

}

}

}

**全文搜索: GET /megacorp/employee/\_search**

{

"query" : {

"match" : {

"about" : "rock climbing"

}

}

}

**短语搜索：确切的匹配若干个单词或者短语。**

例子：

GET /megacorp/employee/\_search

{

"query" : {

"match\_phrase" : {

"about" : "rock climbing"

}

}

}

**查询返回John Smith的文档：**

{

...

"hits": {

"total": 1,

"max\_score": 0.23013961,

"hits": [

{

...

"\_score": 0.23013961,

"\_source": {

"first\_name": "John",

"last\_name": "Smith",

"age": 25,

"about": "I love to go rock climbing",

"interests": [ "sports", "music" ]

}

}

]

}

}

## 3.4 代码搜索引擎

### 3.4.1 词法分析

一个Python程序先由词法分析器生成token序列，然后再将token序列输入语法分析器，进行语法分析生成抽象语法树。

词法分析：将一段字符序列 （如一段程序代码），转化成为一个token序列。

**标识符：**

由字母 (包含大小写)，数字，下划线 (\_) 组成，其中，标识符的首位必须为字母或下划线，不能为数字。

**关键字（保留字）:**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| False | await | else | import | pass | None | break |
| except | in | raise | True | class | finally | is |
| return | and | continue | for | lambda | try | as |
| def | from | nonlocal | while | assert | del | global |
| not | with | async | elif | if | or | yield |

**操作符：**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| + | - | \* | \*\* | / | // | % |
| @ | << | >> | & | | | ^ | ~ |
| < | > | <= | >= | == | != |  |

**定界符：**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ( | ) | [ | ] | { | } | , |
| : | . | ; | @ | = | -> | += |
| -= | \*= | /= | //= | %= | @= | &= |

**实值：**

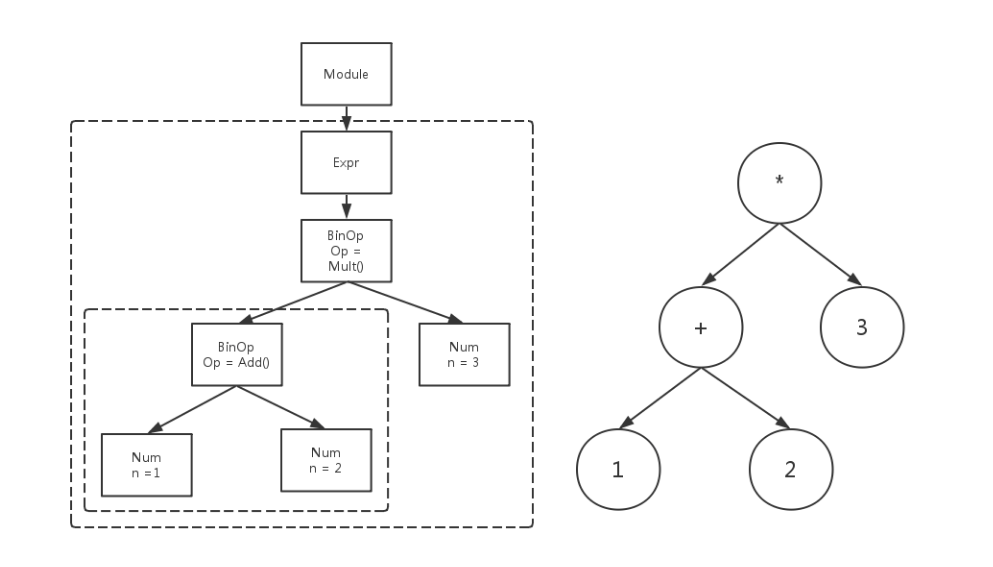
整数型，浮点型，字符串型

### 3.4.2 语法分析

语法分析器基于特定语言的语法，将token序列（由词法分析器生成）转换成一个抽象语法树。

抽象语法树（Abstract Syntax Tree，AST），或简称语法树（Syntax tree），是源代码语法结构的一种抽象表示。它以树状的形式表现编程语言的语法结构，树上的每个节点都表示源代码中的一种结构。之所以说语法是 “抽象” 的，是因为这里的语法并不会表示出真实语法中出现的每个细节。比如，嵌套括号被隐含在树的结构中，并没有以节点的形式呈现。

例如，（1 + 2）\* 3的语法树如下：



### 3.4.3 分析工具

Python语言的抽象语法树的生成工具考虑使用以下两者：

#### 3.4.3.1 PLY package

PLY（python lex yacc）是由Python实现的lex和yacc工具。

PLY 的设计目标是尽可能的沿袭传统 lex 和 yacc 工具的工作方式，提供丰富的输入验证、错误报告和诊断。

#### 3.4.3.2 Python ast module

Python的ast模块，能够通过parse()内置函数生成抽象语法树，并提供对抽象语法树的遍历。

结点类型有：Num, Str, Name, Expr, keyword, Assign, Print, Import, If, For, While, Return......(99种)

### 3.4.4 代码相似性度量

#### 3.4.4.1 编辑距离

定义：给定两个树，编辑距离就是将a树转化成b树所需要的最少操作次数。

操作只允许以下三种：

* 删除一个结点
* 插入一个结点
* 替换一个结点

通过对程序代码进行语法分析，得到程序的AST，然后计算两段代码的AST之间的编辑距离，即可度量两段代码的相似性。

注意：

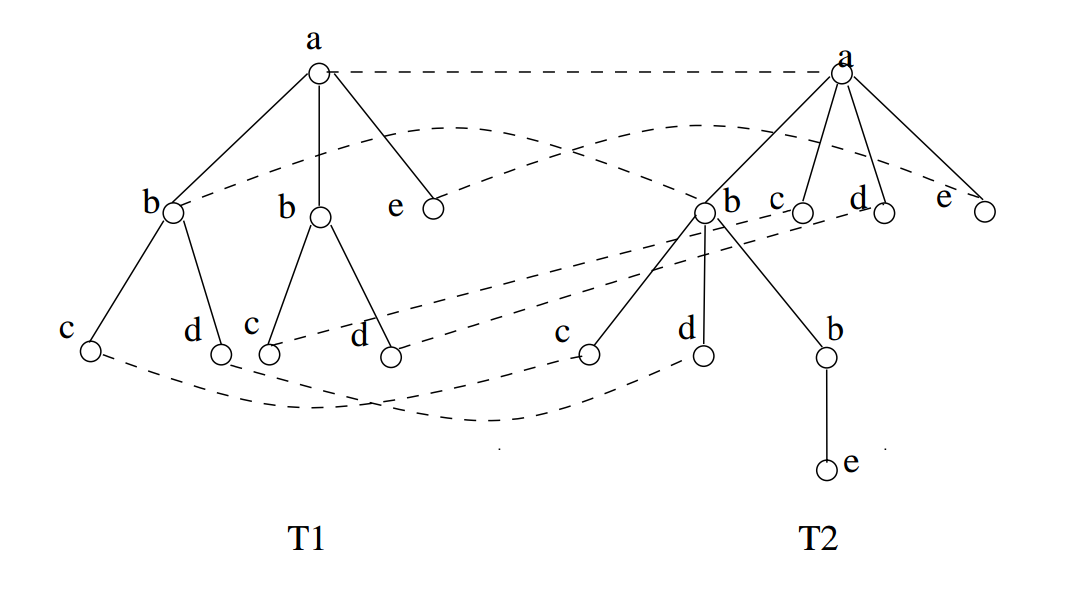
两个树之间的编辑距离确实可以有效捕捉他们之间的相似性，但是这并不是一种高效的算法，因为：

1. the complexity of computing the editing distance between two trees is expensive.
2. it requires many pairwise comparisons to locate similar code.

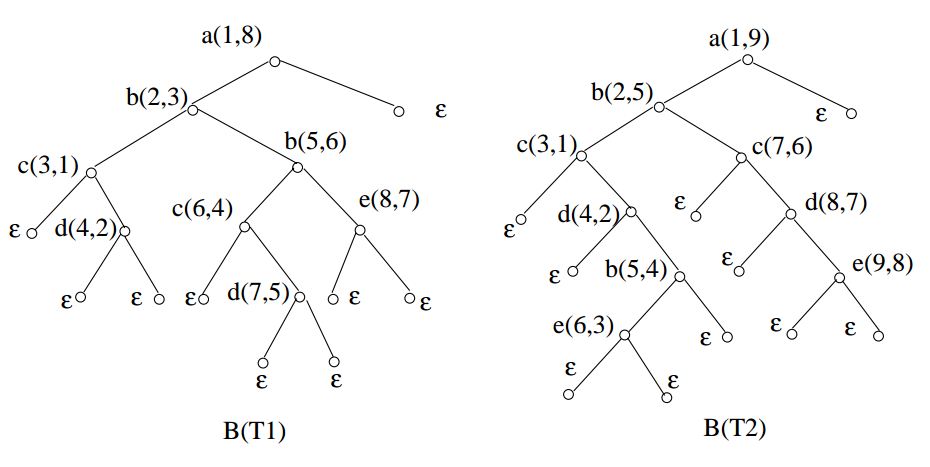
#### 3.4.4.2 夹角余弦值 / 欧氏距离

使用数值型向量来近似表达树的结构信息，从而将计算树的相似性问题转化为计算两个向量之间的相似性问题。

##### 3.4.4.2.1 建立索引



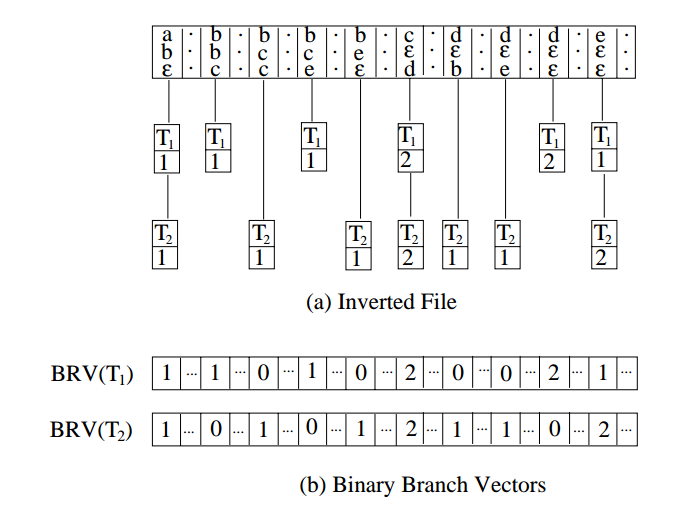
将多叉树转化成二叉树（左儿子，右兄弟）, 并对二叉树进行规范化，即语法树中没有左孩子的结点，用ε结点作为其左孩子，没有右孩子的，用ε结点作为其右结点，规范化之后的二叉树所有叶节点都是ε结点。



（原始语法树为T1、T2，转化并规范化之后的语法树为B(T1)、B(T2) ）

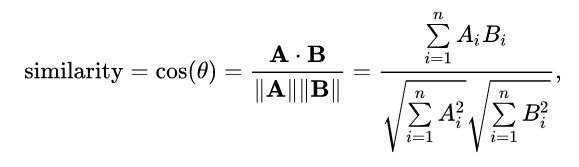
定义上述二叉树中两层二叉子树为Binary Branch，对所有语法树的Binary Branch建立倒排索引，索引包含两个部分：

1. 索引词典：数据库代码语法树中存在的所有Binary Branch
2. 倒排表：每一个索引项的倒排表记录了该索引项在对应的语法树中出现的次数

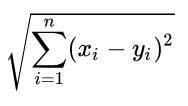


如上图所示，每个语法树的向量表达可以从索引文件中得到，向量的每个维度代表每个Binary Branch在对应语法树中出现的次数。

##### 3.4.4.2.2 计算相似性



夹角余弦值：



欧式距离：

#### 3.4.4.3 Token索引

提供基于Token的索引方式，不进行语法分析。

## 3.5 Online Judge 判题模块

判题模块的主要功能是，编译运行用户提交的代码，测试代码的正确性，并依据一系列的评价标准，给予用户相应的反馈。

### 3.5.1 判题流程

1. 针对一个题目，用户指定一种编程语言，编写该题目对应的解题代码。
2. 用户点击提交按钮，数据库中写入用户-题目-解题代码相关的数据条目。
3. 将用户提交的代码保存为代码文件，并根据题目获取的测试输入文件和测试输出文件。
4. 编译运行该代码文件，输入测试样例，得到输出结果。
5. 判断用户提交代码输出和测试输出之间的匹配情况，以此为据计算最后得分。

在对用户提交的代码进行评测的过程中，可能会出现如下问题:

1. AC (Accepted): 用户提交的代码运行正常，并通过了所有的测试点。计做满分。
2. WA (Wrong Answer): 用户提交的代码正常，但并没有通过所有的测试点，某些测试点错误。部分计分。
3. TLE (Time Limit Excced): 用户提交的代码异常，运行时间过长，超出了该题目的时间限制。按照错误处理。
4. OLE (Output Limit Excced): 用户提交的代码异常，输出内容超出了某个该题目的输出限制。按照错误处理。
5. MLE (Memory Limit Excced): 用户提交的代码异常，运行使用内存超出了该题目的内存使用限制。按照错误处理。
6. RE (Runtime Error): 用户提交的代码通过了编译，在运行时出现了异常。按照错误处理。
7. PE (Presentation Error): 用户提交的代码运行正常，且输出的结果正确，但是展示格式错误。按照错误处理。
8. CE (Compile Error): 用户提交的代码编译错误。按照错误处理。

以面向的用户群体为标准，以上规则可以进行调整。比如，只展示AC和WA两种，即全部通过和部分通过。除此之外的评判标准只在后台保留，不加展示。

### 3.5.2 安全性保证

因为用户提交的代码是不被信任的，所以不应该直接在服务器环境运行，需要限制用户提交代码的运行环境和资源使用情况。主要跟踪用户代码的运行时内存使用情况、CPU占用情况、文件访问情况，保证用户代码在使用正常资源的情况下，能够返回结果，并不对运行环境造成危害。

假设运行环境为Linux操作系统。在实施过程中，需要调用Linux系统API以达到对资源进行追踪限制的目的；需要使用seccomp功能，达到用户代码和系统环境隔离的目的。