# 代码搜索引擎

## 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：**

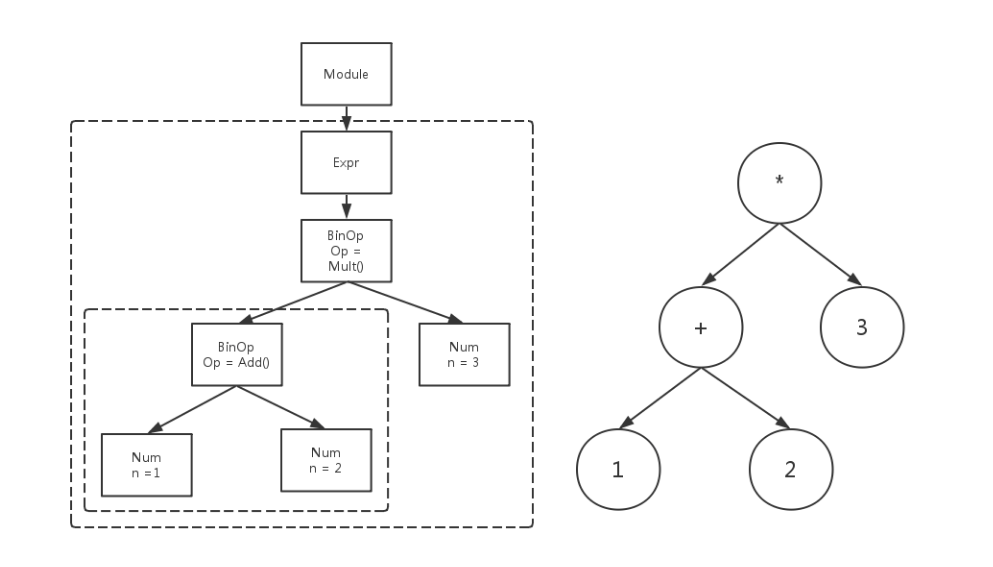
整数型，浮点型，字符串型

## 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：



## 4.3 词法分析，语法分析工具

Python语言的抽象语法树的生成工具考虑使用以下两者：

### 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。

### 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)

## 4.4 代码相似性度量

### 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：

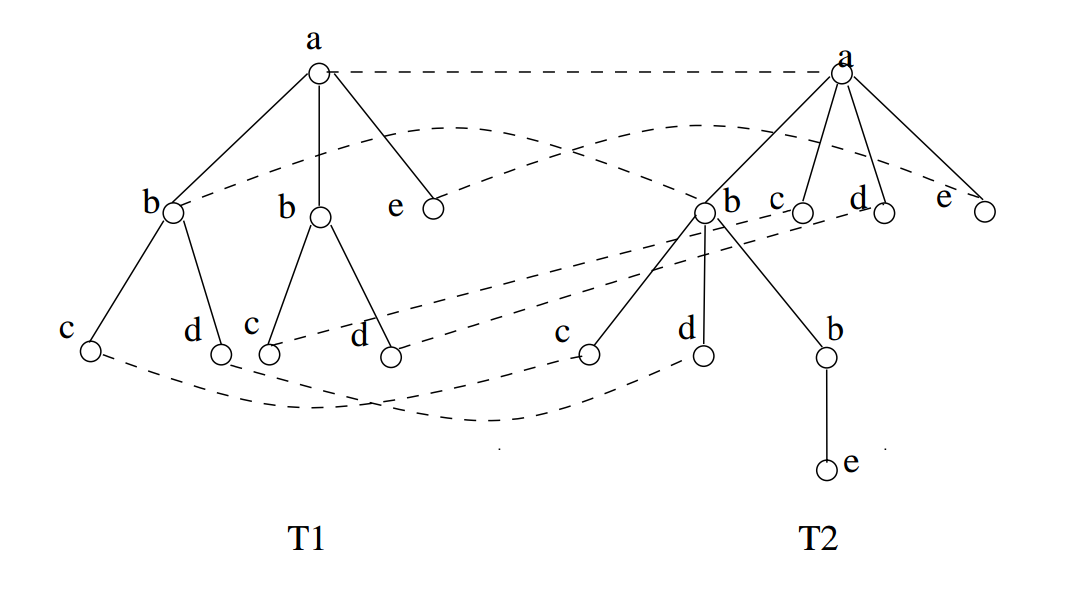
两个树之间的编辑距离确实可以有效捕捉他们之间的相似性，但是这并不是一种高效的算法，因为：

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

### 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。

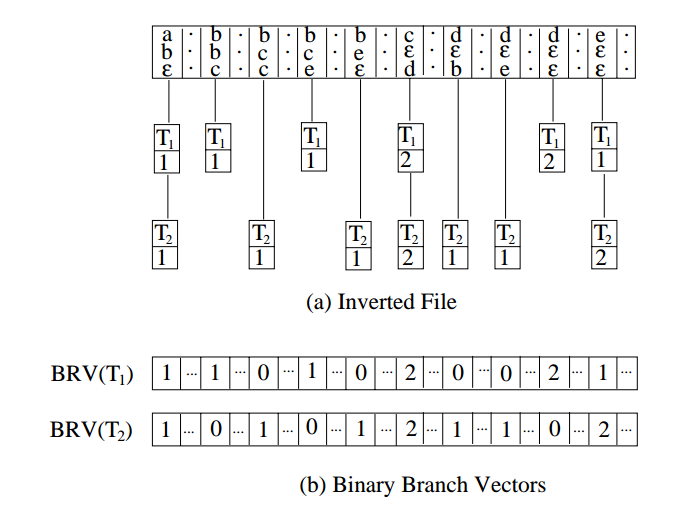
#### 4.4.2.1 建立索引



首先将多叉树转化成二叉树（左儿子，右兄弟表达）

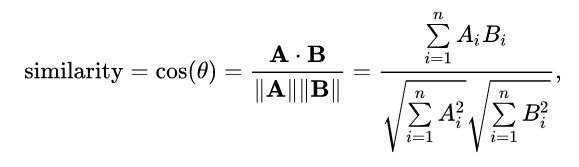
对所有的语法树的子分支（子树）建立倒排索引，索引包含两个部分：

1. 词典：数据库代码中存在的所有子树分支
2. 倒排表：每一个词项的倒排表记录了该词项在对应的语法树中出现的次数

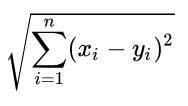


如上图所示，每个语法树的向量表达可以从索引文件中得到，向量的每个维度代表每个子树分支在对应语法树中出现的次数。

#### 4.4.2.2 计算相似性



夹角余弦值：



欧式距离：