

# 数据结构实验报告

姚苏航 PB22061220

## 1 问题描述

### 1.1 实验题目

利用哈希表统计两源程序的相似性。

### 1.2 基本要求

对于两个 C 语言的源程序清单，用哈希表的方法分别统计两程序中使用 C 语言关键字的情况，并最终按定量的计算结果，得出两份源程序的相似性。

### 1.3 测试数据

事先给出的 file 文件夹，包含关键词表和三份源程序文件，程序之间有相近的和差别大的。文件内容详见附录??。

## 2 需求分析

1. 扫描给定的源程序，累计在每个源程序中 C 语言关键字出现的频度 (为保证查找效率，建议自建哈希表的平均查找长度不大于 2)，通过这种方式扫描两个源程序，提取其特征向量。
2. 通过计算向量  $X_i$  和  $X_j$  的相似值来判断对应两个程序的相似性，相似值的判别函数计算公式为：

$$S(X_i, X_j) = \frac{X_i^T \cdot X_j}{|X_i| \cdot |X_j|} \quad (1)$$

通过这种方式，可以初步判断两个源程序的相似性，如图??所示。其中  $S$  反映了两向量的夹角的余弦，当  $S$  趋近于 1 时，两向量夹角趋于 0，即两向量趋于相似，反之亦然。

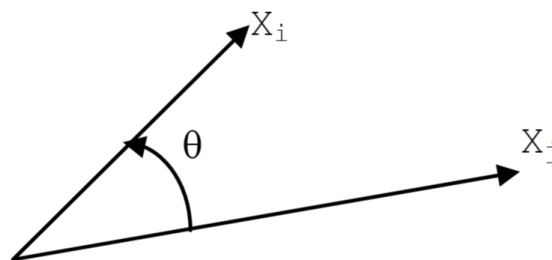


图 1: Similarity

3. 在有些情况下,  $S$  不能很好地反映两向量的相似性, 还需要进一步的考虑。例如, 在  $S$  接近于 1 时, 两向量的模的差距不能很好地被反映, 如图??所示。因此引入几何距离  $D$ , 用于反应两向量终点间的距离。

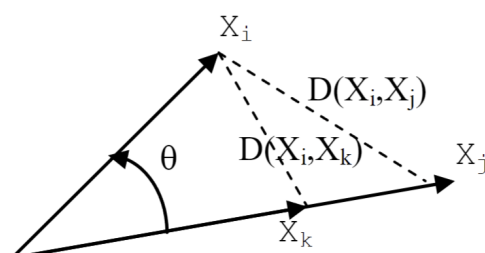


图 2: Distance

4. 通过分别比较很相近和差别很大的三个源代码, 实践这种方法的有效性。

## 3 概要设计

### 3.1 所用到的数据结构及其 ADT

### 3.2 主程序流程及其模块调用关系

### 3.3 核心模块的算法伪码

## 4 详细设计

### 4.1 实现概要设计中的数据结构 ADT

### 4.2 实现每个操作的伪码, 重点语句加注释

### 4.3 主程序和其他模块的伪码

## 5 调试分析

### 5.1 问题分析与体会

### 5.2 时空复杂度分析

## 6 使用说明

用户将事先准备好的关键词表 (keyword.txt) 和需要统计相似性的程序放入 file 文件夹中, 运行时程序将通过关键词表建立哈希表, 并通过查找哈希表构建两程序的向量, 最后通过判别函数计算两程序相似性和向量的几何距离。

在本项目中, 使用事先给出的测试数据, 准备三个编译和运行都无误的 C 程序, 程序之间有相近的和差别大的, 通过 similar.c 和 different.c 两个程序与 main.c 进行比较, 可以直观展现出比较的效果。

---

**Algorithm 1** 用归并排序求逆序数

---

**Require:** *Array* 数组, *n* 数组大小

**Ensure:** 逆序数

```
1: function MERGESORT(Array, left, right)
2:   result  $\leftarrow$  0
3:   if left < right then
4:     middle  $\leftarrow$  (left + right)/2
5:     result  $\leftarrow$  result + MERGESORT(Array, left, middle)
6:     result  $\leftarrow$  result + MERGESORT(Array, middle, right)
7:     result  $\leftarrow$  result + MERGER(Array, left, middle, right)
8:   end if
9:   return result
10: end function
11:
12: function MERGER(Array, left, middle, right)
13:   i  $\leftarrow$  left
14:   j  $\leftarrow$  middle
15:   k  $\leftarrow$  0
16:   result  $\leftarrow$  0
17:   while i < middle and j < right do
18:     if Array[i] < Array[j] then
19:       B[k + +]  $\leftarrow$  Array[i + +]
20:     else
21:       B[k + +]  $\leftarrow$  Array[j + +]
22:       result  $\leftarrow$  result + (middle - i)
23:     end if
24:   end while
25:   while i < middle do
26:     B[k + +]  $\leftarrow$  Array[i + +]
27:   end while
28:   while j < right do
29:     B[k + +]  $\leftarrow$  Array[j + +]
30:   end while
31:   for i = 0  $\rightarrow$  k - 1 do
32:     Array[left + i]  $\leftarrow$  B[i]
33:   end for
34:   return result
35: end function
```

---

## 7 测试结果

### 7.1 输入数据

输入数据从给出的测试文件中读取,读取 keyword.txt 文件生成哈希表,再分别读取 main.c,similar.c,different.c 并进行比较。

### 7.2 输出数据

输出结果显示在终端,内容如下:

X\_(../file/similar.c):

0 1 1 4 0 1 0 3 3 1 2 1 2 0 1 1 4

X\_(../file/different.c):

0 1 0 1 0 0 1 2 1 0 3 0 0 0 1 3 2

X\_(../file/main.c):

0 1 1 3 0 1 0 3 2 1 2 1 2 0 1 1 4

S\_(Sim&Main):0.988174

D\_(Sim&Main):1.41421

S\_(Dif&Main):0.740121

D\_(Dif&Main):4.89898

## A 实验源代码文件

为方便查看,附录中的链接文件均为 txt 格式

[define.h](#)

[main.cpp](#)

[OpenHashing.h](#)

[OpenHashing.cpp](#)

[system.h](#)

[system.cpp](#)

[SimAsses.h](#)

[SimAsses.cpp](#)

## B 实验用测试文件

[main.c](#)

[different.c](#)

[similar.c](#)

[keyword.txt](#)