软件安全实验三 恶意代码特征

1181000420 韦昆杰

2021 年 10 月 24 日

目录

1 实验项目描述

面向网络恶意代码的特征提取

1.1 理解基于最长公共子序列的协议特征提取方法

1. 掌握网络恶意代码特征的提取流程
2. 学习最长公共子序列的提取算法

1.2 实现字符串最长公共子序列的提取算法

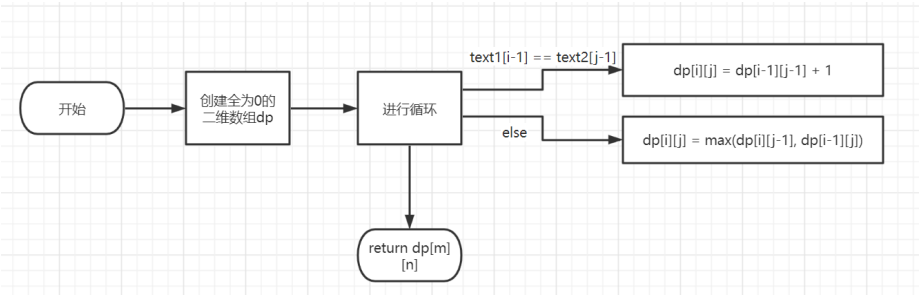
1. 利用动态规划的方法实现字符串最长公共子序列的提取
2. 依据输入的字符串构建 L(m,n) 数组，利用 L(m,n) 数组查找两个字符串之间的最长公共子序列

2 实验要求

1. 实验数据准备。利用 ASCII 字符集做为输入集, 不考虑多字节编码的中文、英文字符集。
2. 程序的输入部分：2 个字符串。输出部分：这 2 个字符串的最长公共子序列，如有多个一同给出。
3. 实验结果和实验数据一起给出。

3 实验结果

3.1 设计的算法流程图，以及算法说明



算法说明:

设 $X = (x1, x2...x_m)$ 和 $Y = (y1, y2...y_n)$ 是两个序列，将 X 和 Y 的最长公共子序列记为 $LCS(X, Y)$，可得以下公式

设 X=(x1,x2,.....xn) 和 Y=y1,y2,.....ym 是两个序列，将 X 和 Y 的最长公共子序列记为 LCS(X,Y)，从 X 和 Y 的最后一个字符开始比较，可知

$$LCS(X_i, Y_j) = \begin{cases} 0, & if \ \ i = 0 \ or \ j = 0 \\ LCS(X_{i-1}, Y_{j-1}) + 1, & if \ \ i, j > 0 \ and \ x_i = y_i \\ max(LCS(X_i, Y_{j-1}), LCS(X_{i-1}, Y_j)), & if \ \ i, j > 0 \ and \ x_i \neq y_i \end{cases}$$

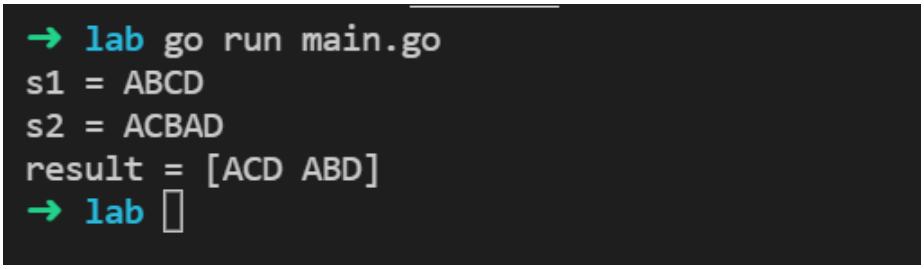
3.2 关键的数据结构，及简单说明

二维数组 dp，假设 2 个字符串分别为 s1 和 s2，那么 dp[i][j] 代表 s1[0:i] 和 s2[0:j] 的最长公共子序列的长度

二维数组 res，res[i][j] 表示 s1[0:i] 和 s2[0:j] 的最长公共子序列

最终返回结果是 res[m][n]，m 为 s1 的长度，n 为 s2 的长度

3.3 实验结果的截图



3.4 源程序

```go
func longestCommonSubsequence(s1, s2 string) []string {
  m, n := len(s1), len(s2)
  dp := [][]int{}
  res := [][][]string{}
  for i := 0; i <= m; i++ {
    dp = append(dp, make([]int, n+1))
    res = append(res, make([][]string, n+1))
  }
  for j := 0; j <= n; j++ {
    res[0][j] = []string{""}
  }
  for i := 1; i <= m; i++ {
    for j := 1; j <= n; j++ {
      if s1[i-1] == s2[j-1] {
        dp[i][j] = dp[i-1][j-1] + 1
        for _, v := range res[i-1][j-1] {
          res[i][j] = append(res[i][j], v+string(s1[i-1]))
        }
      } else {
        dp[i][j] = max(dp[i-1][j], dp[i][j-1])
        if dp[i-1][j] < dp[i][j-1] {
          res[i][j] = append([]string{}, res[i][j-1]...)
        } else if dp[i-1][j] > dp[i][j-1] {
          res[i][j] = append([]string{}, res[i-1][j]...)
        } else {
          m := map[string]bool{}
          for _, v := range res[i][j-1] {
            m[v] = true
          }
          for _, v := range res[i-1][j] {
            m[v] = true
          }
          for v := range m {
            res[i][j] = append(res[i][j], v)
          }
        }
      }
    }
  }
  return res[m][n]
}
```