

**课 程 实 验 报 告**

**课程名称： 串行与并行数据结构及算法**

**专业班级： ACM1701**

**学 号： U201714635**

**姓 名： 伍瀚缘**

**指导教师： 陆枫**

**报告日期： 　2018.11**

**计算机科学与技术学院**

## Lab1 括弧匹配实验

### 1. 实验要求

给定一个由括号构成的串，若该串是合法匹配的，返回串中所有匹配的括号对中左右括号距离的最大值；否则返回NONE。左右括号的距离定义为串中二者之间字符的数量，即*max { j - i + 1 | (si, sj)* 是串s中一对匹配的括号}。要求分别使用枚举法和分治法求解。

### 2. 实验思路

#### 2.1 分治法求解思路

|  |
| --- |
|  |

### 3. 回答问题

#### 3.1 关于枚举法求解

Task 5.2 (5%). What is the work and span of your brute-force solution? You should assume subseq has *O(1)* work and span

|  |
| --- |
|  |

#### 3.2 关于分治法求解

Task 5.4 (20%). The speciﬁcation in Task 5.3 stated that the work of your solution must follow a recurrence that was parametric in the work it takes to view a sequence as a tree. Naturally, this depends on the implementation of SEQUENCE.

1. Solve the work recurrence with the assumption that *Wshowt∈Θ(lg n)* where *n* is the length of the input sequence.

2. Solve the work recurrence with the assumption that *Wshowt∈Θ(n)* where n is the length of the input sequence.

3. In two or three sentences, describe a data structure to implement the sequence α seq that allows showt to have *Θ(lg n)* work.

4. In two or three sentences, describe a data structure to implement the sequence α seq that allows showt to have *Θ(n)* work.

|  |
| --- |
|  |

#### 3.3 关于渐进复杂度分析

Task 6.1 (5%). Rearrange the list of functions below so that it is ordered with respect to O—that is, for every index *i*, all of the functions with index less than i are in big-O of the function at index *i*. You can just state the ordering; you don’t need to prove anything.

1.

2. *f(n) = 2n1.5*

3. *f(n) =(nn)!*

4. *f(n) = 43n*

5. *f(n) = lg(lg(lg(lg(n))))*

6. *f(n) = 36n52 + 15n18 + n2*

7. *f(n) = nn!*

|  |
| --- |
| 5261437 |

Task 6.2 (15%). Carefully prove each of the following statements, or provide a counterexample and prove that it is in fact a counterexample. You should refer to the deﬁnition of big-O. Remember that verbose proofs are not necessarily careful proofs.

1. *O* is a transitive relation on functions. That is to say, for any functions *f*, *g*, *h*, if *f∈O(g)* and *g∈O(h)*, then *f∈O(h)*.

2. *O* is a symmetric relation on functions. That is to say, for any functions *f* and *g* , if *f∈O(g)* , then *g∈O(f)* .

3. *O* is an anti-symmetric relation on functions. That is to say, for any functions *f* and *g* , if *f∈O(g)* and *g∈O(f)*, then *f*=*g*.

|  |
| --- |
|  |