

实验报告

数据结构与算法分析

学生姓名/学号

专业班级

指导老师

梅炳寅 202108010206

计科2102

夏艳

2022年3月29日

目录

[1.问题分析 3](#_Toc11326)

[1.1处理的对象（数据） 3](#_Toc26524)

[1.2实现的功能 3](#_Toc12448)

[1.3处理后的结果如何显示 3](#_Toc28768)

[1.4请用题目中样例，详细给出样例求解过程。 3](#_Toc2547)

[2.数据结构和算法设计 3](#_Toc18436)

[2.1抽象数据类型设计 3](#_Toc9976)

[2.2物理数据对象设计（不用给出基本操作的实现） 4](#_Toc9679)

[2.3算法思想的设计 5](#_Toc20818)

[2.4关键功能的算法步骤（不能用源码） 5](#_Toc23417)

[3. 算法性能分析 5](#_Toc11626)

[3.1时间复杂度 5](#_Toc21520)

[3.2空间复杂度 5](#_Toc29257)

4.不足与反思 6

1.问题分析

1.1处理的对象（数据）

处理的对象为数字及字符组成的字符串，也就是说，我们可以用char来解决，因此，模板类可以赋为char。处理对象以字符串的形式读入，并通过下标访问相应位置的字符。

1.2实现的功能

判断输入的字符串是否是回文（回文，即首尾对称的字符串），也就是判断是否=该字符串是否对称。

1.3处理后的结果如何显示

处理后的结果输出yes表示是回文，输出no表示不是回文。

1.4请用题目中样例，详细给出样例求解过程。

【样例输入1】sdsfegrhglp

1. 读入字符串sdsfegrhglp，存入单链表中；
2. 当前指针指向头，返回0位置值为“s”；
3. 当前指针指向尾，返回末位置值为“p”；
4. 判断s不等于p，触发flag=false，并跳出判断循环；
5. Flag=false，输出no，不是回文。

【样例输入2】 helloolleh

1、读入字符串helloolleh，存入单链表中；

2、当前指针指向头，返回0位置值为“h”；

3、当前指针指向尾，返回末位置值为“h”；

4、判断h等于h，继续下一循环；

5、重复2-3步骤多次，直至o等于o，此时两指针相邻，结束循环；

6、flag=true，输出yes，是回文。

【样例输入3】 asas232sasa

1、读入字符串helloolleh，存入单链表中；

2、当前指针指向头，返回0位置值为“a”；

3、当前指针指向尾，返回末位置值为“a”；

4、判断a等于a，继续下一循环；

5、重复2-3步骤多次，直至两指针重合，结束循环；

6、flag=true，输出yes，是回文。

2.数据结构和算法设计

2.1抽象数据类型设计

template <typename E> class List { // List ADT

private:

void operator =(const List&) {} // Protect assignment

List(const List&) {} // Protect copy constructor

public:

List() {} //构造函数

virtual ~List() {} //析构函数

virtual void insert(const E& item) = 0;//在当前位置插入

virtual void moveToStart() = 0; //将当前指针移到头指针位置

virtual void prev() = 0; //将当前指针向前移动一位

virtual void next() = 0; //将当前指针向后移动一位

virtual int length() const = 0; //返回链表长度

virtual void moveToPos(int pos) = 0; //将当前指针移到pos位置

virtual const E& getValue() const = 0; //返回当前指针指向结点的值

};

2.2物理数据对象设计（不用给出基本操作的实现）

本实验中用到的物理存储结构为单链表结构，设计是用一curr指针指向当前位置，同时保存头指针和尾指针的位置。另外还有char型数据域用于保存当前位置的字母值。

template <typename E> class Link {

public:

E element; //当前节点的值

Link \*next; //指向下一节点的指针

Link(const E& elemval, Link\* nextval =NULL);

Link(Link\* nextval =NULL);

};

template <typename E> class LList: public List<E> {

private:

Link<E>\* head; //链表头指针

Link<E>\* tail; //链表尾指针

Link<E>\* curr; //当前指针

int cnt; //链表长度

void init(); //初始化

public:

LList(int size=100); //构造函数

~LList(); //析构函数

void removeall(); //析构函数调用的清空操作

void insert(const E& it); //在当前位置插入

void moveToStart() //将当前指针移到头指针位置

void prev() //将当前指针向前移动一位

void next() //将当前指针向后移动一位

int length() //返回链表长度

void moveToPos(int pos) //将当前指针移到pos位置

const E& getValue() const //返回当前指针指向结点的值

};

2.3算法思想的设计

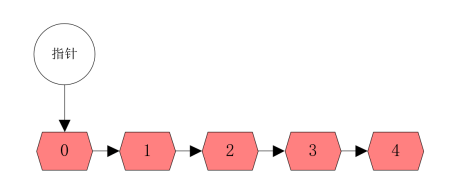
两个指针left与right，分别从头与尾两端同时向中间前进，每次访问一个节点，并比较left与right所对应的值是否一致。初始一个flag为true，一旦发现有不一样的地方就将这个flag赋为false，当两个指针交换位置或者重合时结束循环，退出并检查flag是否为true，若true则表示是回文，反之就不是回文。

考虑到这个单链表左右指针向中间进近的步长相同，可以用控制变量i来模拟这种进近。i限制在0到总长度的一半之间。

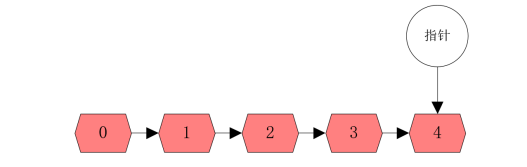
2.4关键功能的算法步骤（不能用源码）

如何用单指针实现两端取值

1、指针在头，char型left取0号位置所对应的值



1. 指针跳至尾，char型right取末位置所对应的值



3、重复以上步骤多次，直至触发最终终止条件

1. 算法性能分析

3.1时间复杂度

该算法的时间复杂度是O（），因为主要矛盾是关键代码函数，在第一层for (int i=0;i<=n;i++)上，积累了n的复杂度，然后mylist.moveToPos(i);这里积累了n的复杂度，与此并列的mylist.moveToPos(mylist.length()-1-i);也积累了n的复杂度，简单来说n\*n，为O（）的复杂度。

严格意义来说mylist.insert(str[i])是1，嵌套for循环，是n。再加上上面主要矛盾。可得，

3.2空间复杂度

该算法的空间复杂度是O（n）

1. 不足与反思

因为实际上我们只能操纵一个指针，所以上述的做法实际上是模拟了两个指针。那我们不妨考虑另外一种形式，建立两个链表，存同样的东西，然后一个链表操纵指针从前往后，另外一个链表操纵指针由后往前，这样就可以在物理层面上实现两个指针的操作，在思维上理解起来也会快一点。

实际上，如果不考虑链表的实现方式，本题其实用线性表的顺序表来做是非常简单的，可以实现O（n）的时间复杂度，通俗来说就是用数组来实现。这样会很简单，但可能这是教师想要锻炼我们的链表能力。