北京郵電大學

实验报告

实验名称: 使用队列和栈判断回文串

班级 : 2019211309 姓名 : 陈悦 学号 : 2019211413 分工 : 文档

班级 : 2019211309 姓名 : 马晓亮 学号 : 2019211400 分工 : 代码

2020 年 *10* 月 *18* 日

一　需求分析

本程序要完成的功能为，判断字符串是否为回文字符串。输入要求是一串字符串，并且要以 # 结束。例如，avava#。如果有必要，可以读入多个字符串串。ava#abaaba#asd88#。我们对输入做出以下限制:

1. 输入的串的总长度不能超过 10^6
2. 输入的所有字符只能在 [0 9][a z][A Z] 中 #

在正常情况下，本程序输出 T 或 F 判断正误。在有多个字符串输入的情况下，判断结果将按顺序输出。以上面两个输入样例为例。
Match
Match Match Dismatch

二　概要设计

主程序，即在 main 函数中调用输入和创建初始化数据结构的函数，进行数据处理，最后输出结果。
我们需要定义以下数据结构

1. 队列，只需要支持队列的基本操作
2. 栈，只需要支持栈的基本操作
3. 处理数据的 solution

对于读入非 # 字符，main 会调用 Solution 中的 insert 函数插入数据。每当读入一个 # 的时候，main 中就会调用 Solution 中 check 函数判断之前读入的串是否为回文串。

三　详细设计

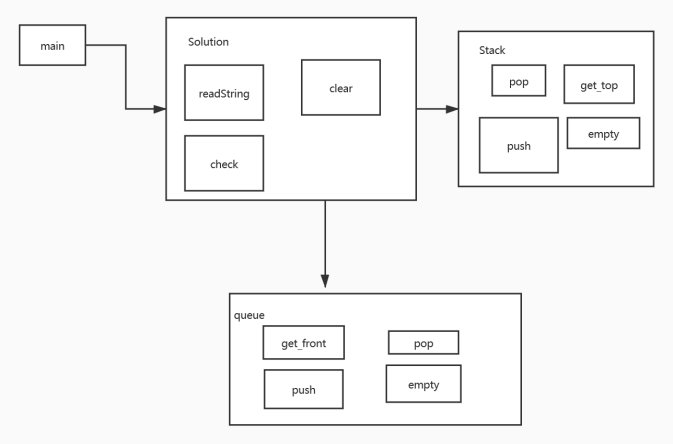

图 1: 函数调用关系图

先分别定义队列和栈数据结构
队列中含有的元素为:

1. 头指针,head
2. 尾指针,tail
3. 插入 insert
4. 弹出数据 pop_tail
5. 获取数据 get_head

同样有栈的定义如下

1. 栈顶指针 top
2. 取得栈顶的数据 get_top
3. 弹出栈顶的数据 pop_top

Algorithm 1 Solution 结构定义

Input: 输入串
Output: 判断串是否回文

```
1: 插入数据函数
2: function INSERT(a)
3:     if a 不合法 then
4:         return FAILURE
5:     end if
6:     将 a 添加到栈中
7:     将 a 添加到队列中
8:     return SUCCESS
9: end function
10:
11: 检查是否为回文串函数
12: function CHECK(void)
13:     while 栈和队列非空 do
14:         从栈中取出一个元素 stack_data
15:         从队列中取出一个元素 queue_data
16:         if stack_dat ≠ queue_data then
17:             return FAILURE
18:         end if
19:         从栈和队列中弹出数据
20:     end while
21:     return SUCCESS
22: end function
```

四　调试分析报告

我们在这里复用了实验二的 stack.h

在使用线性数据结构的操作下，算法复杂度几乎已经达到了下界，即 $O(N)$。但是空间复杂度上面还有许多可以优化的地方。我们人为定义的数据上限为 10^6。但是显然，在大多数情况下用户使用的空间会远远低于这个上界。所以一次申请 10^6 显然是过度浪费内存的。

所以我们将栈的数据存储到一个叫做 Vector 的动态数组中，这个数组开始时数据边界为 10^2 级别。在使用过程中，这个数组会自动调用 *resize*() 扩大边界。我们设一个系数 k，每次 *resize*() 后的边界为原来的 k 倍。接下来我们将测试不同 k 下由 *resize* 造成的浪费。

N	10^4	10^5	10^6
1.6	29634	121688	121688
2.0	32704	65472	65472
2.4	20807	120056	120056

观察发现，在 $k = 1.6$ 与 $k = 2.4$ 时，数据表现较为极端，而 $k = 2.0$ 时较为平稳，所以我们选择 $k = 2.0$ 作为我们的 vector 的 resize 系数。

五　用户使用说明

用户可以使用 IDE 或者手动编译源代码 stack.cpp，获得可执行文件。

笔者使用的 gcc 版本为 8.1.0 运行可执行文件后，用户可以选择文件输入或者交互式输入。在文件输入下输出将会从定向到 output.in。结束程序可以输入 EOF

六　测试结果

我们使用我们手工构造的数据 ava#abaaba#asd88#

根据程序输出，我们判断手工与程序输出相同。即通过测试。