算法设计与分析

第 x 次编程作业-最长回文子串

Lecturer: 杨启哲

Last modified: 2025 年 9 月 16 日

1 问题描述

LeetCode #5-最长回文子串

给你一个字符串 s，找到 s 中最长的回文子串。

2 算法设计

子串指的是连续不间断的一个子序列。问题需要做两件事，首先找到回文子串，并找出其中最长的。我们使用动态规划来解决这个问题。定义状态 $\mathrm{tmp}[i][j]$ 表示字符串 s 的子串 $s[i\ldots j]$ 是否为回文串。状态转移方程如下：

$$\mathrm{tmp}[i][j] = \begin{cases} \mathrm{true}, & \text{if } i == j \\ s[i] == s[j], & \text{if } j == i+1 \\ s[i] == s[j] \wedge \mathrm{tmp}[i+1][j-1], & \text{if } j > i+1 \end{cases}$$

其中，i 和 j 分别表示子串的起始和结束位置。初始时，所有的 $\mathrm{tmp}[i][i]$ 都为 true。然后我们从长度为 2 的子串开始，逐渐增加子串的长度，直到达到字符串 s 的长度。每次更新 $\mathrm{tmp}[i][j]$ 时，如果发现 $\mathrm{tmp}[i][j]$ 为 true 且 $j-i+1$ 大于当前记录的最长回文子串的长度，则更新最长回文子串的起始位置和长度。

循环终止后，我们再通过从长到短的顺序检查 tmp 数组，找到最长的回文子串并返回。

3 实现代码

本次作业使用 C++ 语言实现，代码如下：

```cpp
class Solution {
public:
    string longestPalindrome(string s) {
        int length=s.length();
        int **tmp=new int*[length];
        for(int i=0;i<length;i++)
```

```
        {
            tmp[i]=new int[length];
            for(int j=0;j<length;j++)
            {
                if(j==i)
                    tmp[i][j]=1;
                else
                    tmp[i][j]=0;
            }
        }
        for(int i=1;i<length;i++)
        {
            for(int j=0;j<length-i;j++)
            {
                if(s[j]==s[j+i])
                {
                    if(i==1||tmp[j+1][j+i-1]==1)
                        tmp[j][j+i]=1;
                }
                else
                    tmp[j][j+i]=0;
            }
        }
        for(int i=length-1;i>=0;i--)
        {
            for(int j=0;j<length-i;j++)
            {
                if(tmp[j][j+i]==1)
                {
                    //cout<<j<<" "<<i<<endl;
                    return s.substr(j,i+1);
                }
            }
        }
        return s;
    }
};
```

4 测试结果

执行用时分布为：

消耗内存分布为：