

## C++方向编程题答案

### 答案说明：

大家如果对本次题目或者答案有问题，可以联系下方的出题老师答疑。

### 出题老师：

选择题：时亮益 qq: 569334855

代码题：高博 qq: 1262913815

## 第一周

### day5

#### 题目ID: 45842-统计回文

链接: <https://www.nowcoder.com/practice/9d1559511b3849deaa71b576fa7009dc?tpId=85&&tqId=29842&rp=1&ru=/activity/oj&qu=/ta/2017test/question-ranking>

#### 【题目解析】：

首先以后面对这种题目描述比较长的题，不要害怕，它里面的大部分描述都只是为题做铺垫，所以读题时抓住重点。

什么是回文字符串，题目里面说就是一个正读和反读都一样的字符串，回文串也就是前后对称的字符串。本题是判断是否是回文串的变形题。字符串本身不一定是回文，把第二个字符串插入进去看是否是回文。

#### 【解题思路】：

本题使用暴力求解方式计算即可，遍历str1，将str2 insert进入str1的每个位置，判断是否是回文，是就++count；需要注意的是这里不能str1.insert(i, str2)，这样的话str1改变了，判断下一个位置就不对了。所以每次使用str1拷贝构造一个str，然后str.insert(i, str2)，再判断。

```
#include<iostream>
#include<string>
using namespace std;

// 判断是否是回文
bool IsCircleText(const string& s)
{
    size_t begin = 0;
    size_t end = s.size()-1;
    while(begin < end)
    {
        if(s[begin] != s[end])
            return false;

        ++begin;
        --end;
    }
}
```

```

        return true;
    }

    int main()
    {
        std::string str1, str2;
        getline(cin, str1);
        getline(cin, str2);

        size_t count = 0;
        for(size_t i = 0; i <= str1.size(); ++i)
        {
            // 将字符串2插入到字符串1的每个位置，再判断是否是回文
            string str = str1;
            str.insert(i, str2);
            if(IsCircleText(str))
                ++count;
        }

        cout<<count<<endl;
        return 0;
    }

```

### 58539-连续最大和

<https://www.nowcoder.com/practice/5a304c109a544aef9b583dce23f5f5db?tpId=85&&tqId=29858&rp=1&ru=/activity/oj&qr=/ta/2017test/question-ranking>

#### 【题目解析】：

本题是一个经典的动规问题，简称dp问题，但是不要害怕，这个问题是非常简单的dp问题，而且经常会考察，所以大家一定要把这个题做会。本题题意很简单，就是求哪一段的子数组的和最大。

#### 【解题思路】：

状态方程式：  $\max(dp[i]) = \max(\max(dp[i-1]) + arr[i], arr[i])$

$dp[i]$  就是以数组下标为  $i$  的数做为结尾的最大子序列和，注意是以  $i$  为结尾，比如说现在有一个数组 {6,-3,-2,7,-15,1,2,2}， $dp[2]$  就是以 -2 为结尾的，那么显然  $dp[2]$  的最大值就是 1 (6, -3, -2)， $dp[3]$  要以 7 结尾那么以 7 结尾的子序列最大和就是 8 (6, -3, -2, 7)。现在我们开始细细品一下上面这个递推式，求  $dp[i]$  的时候是不是有两种可能，要么就是像上面的  $dp[3]$  一样， $dp[2]$  求出来是 1 了，再加上自己  $array[3]$  是最大的，那么还有一种可能就是说如果  $dp[2]$  我求出来是 -100，那如果我也是  $dp[2] + array[3]$  的话是 -93，这时候  $dp[2]$  反而是累赘，最大就是自己（因为前面定义了必须以  $i$  为结尾，也就说必须以 7 结尾）。

```

#include <iostream>
#include<vector>
using namespace std;
int GetMax(int a, int b) //得到两个数的最大值
{
    return (a) > (b) ? (a) : (b);
}
int main()

```

```
{  
    int size;  
    cin >> size;  
    vector<int> nums(size);  
    for(size_t i = 0; i < size; ++i)  
        cin >> nums[i];  
  
    int Sum = nums[0];    //临时最大值  
    int MAX = nums[0];    //比较之后的最大值  
  
    for (int i = 1; i < size; i++)  
    {  
        Sum = GetMax(Sum + nums[i], nums[i]);    //状态方程  
        if (Sum >= MAX)  
            MAX = Sum;  
    }  
  
    cout << MAX << endl;  
    return 0;  
}
```

比特就业课