69390-删除公共字符

链接：https://www.nowcoder.com/practice/f0db4c36573d459cae44ac90b90c6212?tpId=85&&tqId=29868&rp=1&ru=/activity/oj&qru=/ta/2017test/question-ranking

【题目解析】：

本题描述很简单，题目描述很清楚，读题即可

【解题思路】：

本题如果使用传统的暴力查找方式，如判断第一个串的字符是否在第二个串中，在再挪动字符删除这个字符的方式，效率为O(N^2)，效率太低，很难让人满意。

1. 将第二个字符串的字符都映射到一个hashtable数组中，用来判断一个字符在这个字符串。
2. 判断一个字符在第二个字符串，不要使用删除，这样效率太低，因为每次删除都伴随数据挪动。这里可以考虑使用将不在字符添加到一个新字符串，最后返回新新字符串。

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

int main()
{
    // 注意这里不能使用cin接收，因为cin遇到空格就结束了。
    // oj中IO输入字符串最好使用getline。
    string str1,str2;
    //cin>>str1;
    //cin>>str2;
    getline(cin, str1);
    getline(cin, str2);

    // 使用哈希映射思想先str2统计字符出现的次数
    int hashtable[256] = {0};
    for(size_t i = 0; i < str2.size(); ++i)
    {
        hashtable[str2[i]]++;
    }

    // 遍历str1，str1[i]映射hashtable对应位置为0，则表示这个字符在
    // str2中没有出现过，则将他+=到ret。注意这里最好不要str1.erases(i)
    // 因为边遍历，边erase，容易出错。
    string ret;
    for(size_t i = 0; i < str1.size(); ++i)
    {
        if(hashtable[str1[i]] == 0)
            ret += str1[i];
    }

    cout<<ret<<endl;
    return 0;
}
```

58539-连续最大和

https://www.nowcoder.com/practice/5a304c109a544aef9b583dce23f5f5db?tpId=85&&tqId=29858&rp=1&ru=/activity/oj&qru=/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】：

假设sum[i-1]是以数组中第nums[i-1]为最后一个元素的一段子数组最大和，

sum[i]是以数组中第nums[i]为最后一个元素的一段子数组最大和，

那么**sum[i] = max(sum[i-1], 0) + nums[i]**，理解了这个，下面代码中用sum1表示sum[i-1]，sum2表示sum[i]，如果计算出更大的子数组和则保存到result中。如果sum[i]，及sum2都小于0了，则置为0，因为他加上数组下一个数，不会计算出更大的子数组和。

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

int main()
{
    int size;
    cin >> size;
    vector<int> nums(size);
    for(size_t i = 0; i < size; ++i)
        cin >> nums[i];

    int result = nums[0];
    int sum1 = 0, sum2 = 0;
    for (int i = 0; i < nums.size(); i++)
    {
       // 计算到num[i]的子数组的最大和
       sum2 = sum1 >= 0 ? sum1+nums[i] : nums[i];
       if(sum2 > result)
           result = sum2;
       if(sum2 < 0)
           sum2 = 0;

        sum1 = sum2;
    }

    cout<<result<<endl;

    return 0;
```

36891-24点游戏算法

https://www.nowcoder.com/questionTerminal/fbc417f314f745b1978fc751a54ac8cb?toCommentId=142189

【题目解析】：

题意很简单，4个数一起运算，只要能算出24，就满足。

【解题思路】：

这里其实使用了DFS的方式，穷举了4个数在一起所有的计算结果，如果可以算出24，就是可以的。

假设这四个数为A B C D，那么这里的计算先深度计算A + B + C + D：

回退一层，再计算 A + B + C - D，回退一层，再计算 (A + B + C) * D，回退一层，再计算 (A + B + C) -/ D。

再回退一层，再计算 A + B - C + D，回退一层，再计算 A + B - C - D...

如此重复，如果算出等于24，则结束，没有则继续计算。

```
//判断vector<double>a中的数进行加减乘除后得到的结果，与result进行加减乘除任意一操作后是否能得到total
#include<iostream>
#include<vector>
using namespace std;
bool is24(const vector<double>& a, int total, double result) {//这种题做多了才有灵感
    //result的类型不要写成int了
    if (a.empty()){//递归终止条件
        return total == result;
    }
    for (int i = 0; i < a.size(); i++){
        vector<double>b(a);
        b.erase(b.begin() + i);
        if (is24(b, total, result + a[i]) ||
            is24(b, total, result - a[i]) ||
            is24(b, total, result * a[i]) ||
            is24(b, total, result / a[i])){
            return true;
        }
    }
    return false;
}

int main(){
    vector<double>a(4, 0);
    while (cin >> a[0] >> a[1] >> a[2] >> a[3]){
        if (is24(a, 24, 0)){
            cout << "true" << endl;
        }
        else{
            cout << "false" << endl;
        }
    }
    system("pause");
    return 0;
}
```