PAT 甲级刷题实录——1002

作者：aopstudio
原文链接：https://ld246.com/article/1578386402401
来源网站：链滴
许可协议：署名-相同方式共享 4.0 国际 (CC BY-SA 4.0)

原题

This time, you are supposed to find A+B where A and B are two polynomials.

Input Specification:

Each input file contains one test case. Each case occupies 2 lines, and each line contains the in ormation of a polynomial:

K\ N_1\ a_{N_1}\ N_2\ a_{N_2}...N_K\ a_{N_K}

where K is the number of nonzero terms in the polynomial, N_i (i=1,2,...,K) and a_{N_i} (i=1,2,...,K) are th exponents and coefficients, respectively. It is given that 1≤ K≤10，0≤ N_K <...< N_2 <N_1 ≤ 1000.

Output Specification:

For each test case you should output the sum of A and B in one line, with the same format as he input. Notice that there must be NO extra space at the end of each line. Please be accurate to 1 decimal place.

Sample Input:

2 1 2.4 0 3.2
2 2 1.5 1 0.5

Sample Output:

3 2 1.5 1 2.9 0 3.2

思路

一开始完全没搞明白题目在问什么，后来看了好几遍，再结合输入和输出的例子，才明白题目想问什。其实把题目转换为数学式子就是用它规定的格式给你两个多项式，比如 y_1=ax+bx^2+cx^3,y_2=i +jx^2+kx^3 ，让你算出 y_1+y_2 并且用它规定的格式输出。像题目中给的案例输入的式子就是 y_1 2.4x+3.2,y_2=1.5x^2+0.5x ，输出的就是 y_3=1.5x^2+2.9x+3.2 。

首先面临的问题是该如何存储输入的数据，可以看到输入的数据主要有三种类型，第一种是每行的第个数字，也就是每个式子包含的元素个数，这个很简单，我们用一个 int 类型去存储即可。还有两种型分别代表指数和系数，并且两者是一一对应的，因此我的想法是用结构体存，因为有好几个元素，且给定了元素个数，所以一开始我的想法是用结构体动态数组存储。

中间的运算过程我们暂且不管。确定了输入数据的存储方式，接下来应当确定输出数据的存储方式。开始我想的还是结构体数组，但是这里出现了问题，我并不知道计算结果的式子中包含几个元素，也无法确定数组的大小，当然后来发现题目里给定了元素个数的最大值，按照那个最大值乘以二就可以但一开始我想的是用链表或者其他可变的数据结构存储，而且尽管这道题给定了个数范围，但在实际用中很难会有给定的数值范围，必须考虑可变长度的数据结构。

经过一番搜寻，找到了 vector 这个神器，相当于可变长度的数组，而且使用方法非常方便。有了 vec or，索性输入数据也用它存吧，因为动态数组还得 new 和 delete，比较麻烦。

我们已经有了两个 vector 存储结构体变量，并且可以实现遍历操作，接下来就要考虑算法了。这个法其实非常类似二路归并排序最后合并过程的算法，如果不了解二路归并算法的同学可以先了解一下基本思路如下：使用两个下标同时对两个 vector 进行比较和遍历，哪个 vector 中指向的元素的指数大，则将这个元素添加到第三个 vector，也就是存储结果的 vector 中去，同时该元素所在的 vector 下标加 1；当两个元素的指数相等时，就把两个元素的系数加起来产生新的元素，并存储到第三个 vector 中去，同时前两个 vector 的下标都加 1（注意，这里存在一个问题，下文会讲到），代码如下：

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

struct poly
{
    int exp;
    float coe;
};
int main()
{
    int a,b;
    cin>>a;
    vector<poly> poly1;
    vector<poly> poly2;
    vector<poly> sum;
    for(int i=0;i<a;i++)
    {
        poly p;
        cin>>p.exp>>p.coe;
        poly1.push_back(p);
    }
    cin>>b;
    for(int i=0;i<b;i++)
    {
        poly p;
        cin>>p.exp>>p.coe;
        poly2.push_back(p);
    }
    int i=0,j=0;

    while(i<a&&j<b)
    {
        if(poly1[i].exp>poly2[j].exp)
        {
            sum.push_back(poly1[i]);
            i++;
        }
        else if(poly1[i].exp<poly2[j].exp)
        {
            sum.push_back(poly2[j]);
            j++;
        }
        else
        {

            poly add;
```

```
            add.exp=poly1[i].exp;
            add.coe=poly1[i].coe+poly2[j].coe;
            sum.push_back(add);
            i++;
            j++;
        }
    }
    while(i<a)
    {
        sum.push_back(poly1[i]);
        i++;
    }
    while(j<b)
    {
        sum.push_back(poly2[j]);
        j++;
    }
    int n=sum.size();
    cout<<n;
    for(int index=0;index<n;index++)
    {
        printf(" %d %.1f",sum[index].exp,sum[index].coe);
    }

}
```

一开始犯了很多低级错误，比如定义 i,j 是这样定义的 int i,j=0 ，结果就导致只给 j 赋了值，出来的果也必定是错的，都没敢输入评测系统，因为本地调试就是错的。改掉了这个问题后，至少看起来对，就输入评测系统试了试，但是还只是部分正确，后来重新看了一遍题，发现题目要求保留一位小数但我没保留。原本我想用 c++ cout 输出的方式来保留一位小数的，但是网上查了一下还要添加头文件 omanip ，还要用一个很复杂的语句 cout<<setiosflags(ios::fixed)<<setprecision(1) ，我就放弃，改用 c 语言 printf 的方式， %.1f 就行，非常简单粗暴，反正 C++ 是完全兼容 C 语言的。

但是改掉这些低级错误后，依然过不了评测系统，只是分数提高了一点，结果还是部分正确。这下我真想不到问题出在哪里了。没办法，只好上网去看人家的做法，终于发现一个大问题，就是当两个指相同的时候，如果两个系数正好相反，那么得出的元素应该被舍去而不是把系数 0 存储进结果。这下豁然开朗，修改方法也很简单，只要在相加的时候加个判断语句就行。修改后的代码如下：

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

struct poly
{
    int exp;
    float coe;
};
int main()
{
    int a,b;
    cin>>a;
    vector<poly> poly1;
    vector<poly> poly2;
```

```
    vector<poly> sum;
    for(int i=0;i<a;i++)
    {
        poly p;
        cin>>p.exp>>p.coe;
        poly1.push_back(p);
    }
    cin>>b;
    for(int i=0;i<b;i++)
    {
        poly p;
        cin>>p.exp>>p.coe;
        poly2.push_back(p);
    }
    int i=0,j=0;

    while(i<a&&j<b)
    {
        if(poly1[i].exp>poly2[j].exp)
        {
            sum.push_back(poly1[i]);
            i++;
        }
        else if(poly1[i].exp<poly2[j].exp)
        {
            sum.push_back(poly2[j]);
            j++;
        }
        else
        {
            if((poly1[i].coe+poly2[j].coe)!=0)  //判断系数和是否为0，只有当非零的时候才加入sum，
则直接跳过
            {
                poly add;
                add.exp=poly1[i].exp;
                add.coe=poly1[i].coe+poly2[j].coe;
                sum.push_back(add);
            }
            i++;
            j++;
        }
    }
    while(i<a)
    {
        sum.push_back(poly1[i]);
        i++;
    }
    while(j<b)
    {
        sum.push_back(poly2[j]);
        j++;
    }
    int n=sum.size();
    cout<<n;
```

```
    for(int index=0;index<n;index++)
    {
        printf(" %d %.1f",sum[index].exp,sum[index].coe);
    }

}
```

终于完全正确了，C++ 的执行效率还是快啊。