信息学奥赛课堂笔记11

map，STL

[P1102]A-B 数对https://www.luogu.com.cn/problem/P1102

题目背景

出题是一件痛苦的事情!

相同的题目看多了也会有审美疲劳，于是我舍弃了大家所熟悉的 A+B Problem，改用 A-B 了哈哈!

题目描述

给出一串正整数数列以及一个正整数 C，要求计算出所有满足 $A-B=C$ 的数对的个数（不同位置的数字一样的数对算不同的数对）。

输入格式

输入共两行。

第一行，两个正整数 N, C。

第二行，N 个正整数，作为要求处理的那串数。

输出格式

一行，表示该串正整数中包含的满足 $A-B=C$ 的数对的个数。

样例 #1

样例输入 #1

```
4 1
1 1 2 3
```

样例输出 #1

```
3
```

提示

对于 75% 的数据，$1 \leq N \leq 2000$。

对于 100% 的数据，$1 \leq N \leq 2 \times 10^5$，$0 \leq a_i < 2^{30}$，$1 \leq C < 2^{30}$。

思路

首先看到这道题目，不难能想出这题的一个暴搜算法是什么?

不是要找有多少对A和B他们的差是C吗？，那我们就枚举一个A,枚举一个B,如果$A-B==C$成立则$ans++$

那我们可以写出如下代码

```
#include <bits/stdc++.h>
using namespace std;
int n, c, ans;
int a[200010];
int main() {
    cin >> n >> c;
    for (int i = 0; i < n; i++) {
        cin >> a[i];
    }
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++) {
            if (a[i] - a[j] == c) {
                ans++;
            }
        }
    }
    cout << ans << endl;
    return 0;
}
```

> 时间复杂度$O(n^2)$，n表示数组的长度，我们枚举A时需要$O(n)$,枚举B时需要$O(n)$，嵌套循环。
>
> 空间复杂度为$O(n)$,n表示数组的长度，我们需要为n个数开辟长度为n的空间

那么。大家觉得，暴搜这个算法到底是在哪里导致了时间过长呢?

假如说，现在我有这样一组数据:

> 4 1
>
> 1 1 2 2

那么大家可以直接观察得出，一共有4组$2-1=1$ 在暴搜的过程中，我们是不是找寻了太多次重复的例子了，比如说当我们找到第一个数1的时候，我其实只需要知道$x-1=1$，也就是说，$x=2$，有多少个，是不是就意味着有多少组$2-1=1$，那么对于$i=0$的时候，我就可以使用$O(1)$的时间复杂度就可以求出答案的个数了，对于刚刚的解释，我们给出一个更为书面的表达方式:

既然题目要求的是一共能选取多少组不同的$A-B=C$，那我们对等式移项，$B+C=A$,已知目前我们拿到的数是B,又知道$B+C=A$，所以我们可以找找有多少个A存在，就代表有多少组$B+C=A$，就代表有多少组$A-B=C$。

那么，要统计每个数出现了几次，该用什么STL？欢迎各位同学们参加世纪大赛——《用哪个STL比赛》

使用map，也就是哈希表。

本题标准的解法代码如下:

```cpp
#include <bits/stdc++.h>
using namespace std;
int n, c, a[200010];
long long ans; //已经说烂了，要开long long，我看还有谁还记不住
map<int, int> mp;  //既然我们要统计每个数出现了几次，则应该是int->int的映射
int main() {
    cin >> n >> c;
    for (int i = 0; i < n; i++) {
        cin >> a[i];
        mp[a[i]]++;//读完数之后就把该数出现的次数++
    }
    for (int i = 0; i < n; i++) {
        ans += mp[a[i] + c];//答案直接加上a[i] + c出现的次数
    }
    cout << ans << endl;
    return 0;
}
```

> 时间复杂度$O(nlogn)$，n表示数组的长度，每个数加入map的时间复杂度近似为$O(logn)$。
>
> 空间复杂度为$O(n+C)$,n表示数组的长度，C代表哈希表需要开辟常数的空间，为了保证能把所有的结果存进去

进阶

在本题的基础上，老师给大家说一个进阶的知识点

map所使用的底层是**红黑树**，这是一个**二叉搜索树(Binary Search Tree)**，为了保证map中的key是有序的，需要每次插入时对这个树进行旋转操作。所以每次往map中插入元素和修改的时候，时间复杂度为$O(logn)$

C++的标准STL还为我们提供了一个容器，它叫做$unordered_map$，它的使用方法和map完全一样，唯一的区别就是，$unordered_map$的底层使用的是**哈希表**，各位同学可以自行查阅资料什么是数据结构中的哈希表，它为我们提供了一个像数组一样存储数据的**散列函数**，将原始数据分散在哈希表的不同键上方便我们存储，在这个过程中，插入和修改元素的时间复杂度仅为$O(1)$,这是最小数量级的时间复杂度，那什么时候该使用map，什么时候该使用$unordered_map$呢?

老师先为大家列出他们的差异

	map	$unordered\_map$
插入、修改时间复杂度	$O(logn)$	$O(1)$
是否按照Key有序	是	否
使用的底层逻辑	红黑树	哈希表

那么，当我们对key有要求它必须是有序的时候，只能使用map了，但是例如本题中，我们不关心map里的元素是否需要有序，只关心它能帮助我们统计每个数出现了几次，此时为了进一步降低压缩代码的时间开销，我们应该选用更优的$unordered_map$

将代码修改后如下：

```cpp
#include <bits/stdc++.h>
using namespace std;
```

```cpp
int n, c, a[200010];
long long ans;
unordered_map<int, int> mp;
int main() {
    cin >> n >> c;
    for (int i = 0; i < n; i++) {
        cin >> a[i];
        mp[a[i]]++;
    }
    for (int i = 0; i < n; i++) {
        ans += mp[a[i] + c];
    }
    cout << ans << endl;
    return 0;
}
```

> 时间复杂度$O(n)$，n表示数组的长度，每个数加入哈希表的时间复杂度为$O(1)$。
>
> 空间复杂度为$O(n+C)$,n表示数组的长度，C代表哈希表需要开辟常数的空间。

[P1071] 潜伏者 https://www.luogu.com.cn/problem/P1071

题目描述

R 国和 S 国正陷入战火之中，双方都互派间谍，潜入对方内部，伺机行动。历尽艰险后，潜伏于 S 国的 R 国间谍小 C 终于摸清了 S 国军用密码的编码规则：

1. S 国军方内部欲发送的原信息经过加密后在网络上发送，原信息的内容与加密后所得的内容均由大写字母 A ～ Z 构成（无空格等其他字符）；
2. S 国对于每个字母规定了对应的密字。加密的过程就是将原信息中的所有字母替换为其对应的密字；
3. 每个字母只对应一个唯一的密字，不同的字母对应不同的密字。密字可以和原字母相同。

例如，若规定 A 的密字为 A，B 的密字为 C（其他字母及密字略），则原信息 ABA 被加密为 ACA。

现在，小 C 通过内线掌握了 S 国网络上发送的一条加密信息及其对应的原信息。小 C 希望能通过这条信息，破译 S 国的军用密码。小 C 的破译过程是这样的：扫描原信息，对于原信息中的字母 x（代表任一大写字母），找到其在加密信息中的对应大写字母 y，并认为在密码里 y 是 x 的密字。如此进行下去直到停止于如下的某个状态：

1. 所有信息扫描完毕，A ～ Z 所有 26 个字母在原信息中均出现过并获得了相应的密字；
2. 所有信息扫描完毕，但发现存在某个（或某些）字母在原信息中没有出现；
3. 扫描中发现掌握的信息里有明显的自相矛盾或错误（违反 S 国密码的编码规则）。

例：

如某条信息 XYZ 被翻译为 ABA 就违反了“不同字母对应不同密字”的规则。

在小 C 忙得头昏脑涨之际，R 国司令部又发来电报，要求他翻译另外一条从 S 国刚刚截取到的加密信息。现在请你帮助小 C：通过内线掌握的信息，尝试破译密码。然后利用破译的密码，翻译电报中的加密信息。

输入格式

共三行，每行为一个长度在 1 到 100 之间的字符串。

第一行，为小 C 掌握的一条加密信息；

第二行，为第一行的加密信息所对应的原信息；

第三行，为 R 国司令部要求小 C 翻译的加密信息。

输入数据保证所有字符串仅由大写字母 `A` ～ `Z` 构成，且第一行长度与第二行相等。

输出格式

共一行。

若破译密码停止时出现 $2, 3$ 两种情况，请你输出 `Failed`；

否则请输出利用密码翻译电报中加密信息后得到的原信息。

样例 #1

样例输入 #1

```
AA
AB
EOWIE
```

样例输出 #1

```
Failed
```

样例 #2

样例输入 #2

```
QWERTYUIOPLKJHGFDSAZXCVBN
ABCDEFGHIJKLMNOPQRSTUVWXY
DSLIEWO
```

样例输出 #2

```
Failed
```

样例 #3

样例输入 #3

```
MSRTZCJKPFLQYVAWBINXUEDGHOOILSMIJFRCOPPQCEUNYDUMPP
YIZSDWAHLNOVFUCERKJXQMGTBPPKOIYKANZWPLLVWMQJFGQYLL
FLSO
```

样例输出 #3

```
NOIP
```

提示

【输入输出样例一说明】

原信息中的字母 A 和 B 对应相同的密字，输出 Failed。

【输入输出样例二说明】

字母 Z 在原信息中没有出现，输出 Failed。

75分做法

这道题目极长，各位同学要注意，信息学奥赛不仅考验的使我们写代码能力，更有阅读理解能力，通过阅读题目，我们可以提炼出重要的信息。

输入数据有三行字符串，分别代表加密信息，原信息，要求翻译的加密信息，也就是A,B,C串

对于输入的串来说有些时候并不能成功完成破译，需要输出$Failed$，那么有如下3条约束

1. 所有信息扫描完毕，A ～ Z 所有 26 个字母在B串中均出现过并获得了相应的密字；
2. 所有信息扫描完毕，但发现存在某个（或某些）字母在B出演中没有出现；
3. 扫描中发现掌握的信息里有明显的自相矛盾或错误（违反 S 国密码的编码规则）。

   那我们对这三句话进行一个简单解读就是：

   B串必须要包含26个英文字母，且A串到B串的映射当中不能出现同一字符映射不同的情况(例如样例一，A先映射到A，A再映射到B)

A串到B串这是题目给出我们的一条标准的映射规则，要让我们求的就是C串怎么映射到D串，咱们先不管$Failed$,可以写出如下代码

```cpp
#include <bits/stdc++.h>
using namespace std;
string a, b, c;
unordered_map<char, char> mp;//从字符到字符的映射
int main() {
    cin >> a >> b >> c;
    for (int i = 0; i < a.size(); i++) {
        mp[a[i]] = b[i];//以a[i]作为key可以获得一个b[i]的映射
    }
    for (int i = 0; i < c.size(); i++) {
        c[i] = mp[c[i]];//我们把c[i]都替换成c[i]的映射
    }
    cout << c << endl;
    return 0;
}
```

> 时间复杂度$O(n+m)$， n代表字符串a，b的长度，m代表c串的长度
>
> 空间复杂度$O(n+m+C)$，n代表储存a，b字符串的长度，m代表储存c串的长度，C代表哈希表要开辟常数的空间，本题出现的字符为26个小写字母，所以C最大为常数26。

在不考虑$Failed$的情况，我们可以直接通过本题的**75**分！这已经是个不小的分数了。

90分做法

既然我们刚刚没有考虑$Failed$，现在我们把这个约束条件加入，该如何判断B串是否出现有26个字母呢？我们可以继续使用一个map，或者是set来存储，每次遇到B串的一个字符，直接加入这个集合，最后我们只需要检查集合的元素是否等于26即可

代码如下：

```cpp
#include <bits/stdc++.h>
using namespace std;
string a, b, c;
unordered_map<char, char> mp;
unordered_set<char> s;
int main() {
    cin >> a >> b >> c;
    for (int i = 0; i < a.size(); i++) {
        mp[a[i]] = b[i];
        s.insert(b[i]);//每次直接将B[i]加入集合
    }
    if (s.size() != 26) {//如果不是26个字母均出现，直接输出Failed
        cout << "Failed" << endl;
        return 0;//输出完后要记得退出程序
    }
    for (int i = 0; i < c.size(); i++) {
        c[i] = mp[c[i]];
    }
    cout << c << endl;
    return 0;
}
```

> 时间复杂度$O(n+m)$， n代表字符串a，b的长度，m代表c串的长度
>
> 空间复杂度$O(n+m+C)$，n代表储存a，b字符串的长度，m代表储存c串的长度，C代表哈希表要开辟常数的空间，本题出现的字符为26个小写字母，所以C最大为常数26。

100分做法

那现在这道题目我们还差最后一个情况没考虑到了，那就是当字符发生了冲突该怎么办。

我们要思考一下，这个冲突到底是什么冲突了？首先，如果A串中当前我们发现的字符在先前已经出现了，是不是才需要判断它是不是矛盾了。所以如果这是第一次找到这个字符$a[i]$，咱们就直接忽略它

那如果在先前的遍历当中已经访问到了$a[i]$，现在给出对应映射的$b[i]$是和之前的旧值一样呢？说明没有发生冲突，继续跳过，只有当当前的新值与旧值发生了冲突，也就是不一样的时候才需要$Failed$退出。代码如下

```cpp
#include <bits/stdc++.h>
using namespace std;
string a, b, c;
unordered_map<char, char> mp;
unordered_set<char> s;
int main() {
    cin >> a >> b >> c;
    for (int i = 0; i < a.size(); i++) {
        if (mp.count(a[i]) && mp[a[i]] != b[i]) {//如果a[i]已经出现过了，并且旧值和新值不同。
            cout << "Failed" << endl;//直接输出Failed退出程序
```

```
            return 0;
        }
        mp[a[i]] = b[i];
        s.insert(b[i]);
    }
    if (s.size() != 26) {
        cout << "Failed" << endl;
        return 0;
    }
    for (int i = 0; i < c.size(); i++) {
        c[i] = mp[c[i]];
    }
    cout << c << endl;
    return 0;
}
```

> 时间复杂度$O(n+m)$， n代表字符串a，b的长度，m代表c串的长度
>
> 空间复杂度$O(n+m+C)$，n代表储存a，b字符串的长度，m代表储存c串的长度，C代表哈希表要开辟常数的空间，本题出现的字符为26个小写字母，所以C最大为常数26。

[P9680]string[_view] https://www.luogu.com.cn/problem/P9680

题目背景

C++ 的 string 类是一个功能强大的字符串类，然而由于其字符串算法和内存管理绑定的机制，所以在处理 C 风格字符串时效率低下。

为了解决这个问题，C++17 标准引入了 string_view 类型，将内存管理和字符串算法分离，从而更好地适配了 C 风格字符串的处理。

题目描述

你需要模拟一个简单的 C++ 程序，该程序的每一行必然为如下两种形式之一：

- `string <variable-name>(<initializer>);`
- `string_view <variable-name>(<initializer>);`

其中 `variable-name` 为声明的变量名（保证之前未出现过，且长度不超过 10），`initializer` 为初始化该变量的内容，可以是：

- 字符串字面量，即用双引号引起的字符串（形如 `"abc"`）；
- 之前出现过的变量名 `source`，此时应将 `source` 对应的字符串赋给 `variable-name`。

具体而言，将任意一个字符串 s 赋给 string 类型会进行 $|s|$ 次字符拷贝，而赋给 string_view 类型不会拷贝字符。其中 $|s|$ 为字符串 s 的长度。

你需要计算出该程序中字符拷贝的总次数。

输入格式

第一行输入一个整数 L，代表程序行数。

接下来 L 行，输入一段代码。

输出格式

输出一个整数，代表字符拷贝总次数。

样例 #1

样例输入 #1

```
6
string a("cxyakioi");
string_view b("cxyakapio");
string c(b);
string_view d(a);
string_view cxyakioi(c);
string cxyakapio(d);
```

样例输出 #1

```
25
```

提示

对于每组数据，保证代码长度均不超过 10^4（不包括换行符）。

保证字符串字面量（除去两侧引号）和变量名中只有拉丁字母，且给定的代码严格满足题目要求。

子任务

#	特殊性质	分值
0	样例	0
1	所有变量均为 string_view 类型	10
2	只使用字符串字面量初始化	20
3	-	70

思路

关于这道题目，需要同学们对于字符串的操作熟练度要求非常高，而且对于哈希表的操作也仅限于存储变量名的时候要用到，模拟的难度也大大超出了我们J赛。所以老师就不把题目的详细解析写给大家了，对于模拟题来说，1000个人有1000个哈姆雷特，大家做法迥异，老师贴出我的做法，并直接对代码进行讲解。

```cpp
#include <bits/stdc++.h>
using namespace std;
int T, ans;
unordered_map<string, int> mp;
//哈希表中记录的是每个变量所对应的字符串长度是多少。所以是string到int的映射
int main() {
    cin >> T;
    while(T--){
```

```
        string s, t, valName; // 我们把定义变量时的类型和变量名部分拆成2个字符串来看，valName这个变量用来存储“变量名字"
        cin >> s >> t;
        int i = 0, n = t.size();
        for (; i < n; i++) {
            if (t[i] == '(') break;
            //如果当前位置是左括号则停止，左括号前面的部分代表是变量名，而左括号接下来的值代表的是变量要拷贝的元素。括号正好可以作为一个分隔符。
            valName += t[i];
        }
        //循环结束出来后i应该刚好位于括号上
        //substr函数的作用是截取子字符串，用法是s.substr(从哪一位开始， 截取多少个字符);
        t = t.substr(i + 1, n - i - 3);
        //在这里涉及到一个数学运算，在定义一个字符串时的标准用法为 {string 变量名(初始化值);}
        //所以最右侧的) 和 ; 他们一定是固定占有的字符，所以我们截取从i + 1开始，一直到右括号的左边结束，相当于获取到了括号内的元素
        //此时有两种情况，如果括号内的元素是以引号"开头的，说明这个初始化的值是个具体的字符串，反之则是变量
        int len = t.size();
        if (t[0] == '\"') {//初始化值为具体的字符串
            len -= 2;//在获取长度时需要减去两个引号
        } else {//否则说明初始化值是一个变量
            len = mp[t];//我们直接获取到在哈希表中，这个变量t是多长
        }
        mp[valName] = len;//当前的变量名叫valName，我们把valName的长度为t记入哈希表
        if (s == "string") {
            //最后咱们再判断，如果这个变量的类型是个string的话，说明需要深度拷贝，答案应加上len，否则不加
            ans += len;
        }
    }
    cout << ans << endl;
    return 0;
}
```

> 时间复杂度$O(L)$，L代表n个字符串的总长度和，我们需要遍历所有的字符串各一遍
>
> 空间复杂度$O(L)$，L代表n个字符串的总长度和，我们需要用变量对它们进行存储。