《第五次上机实验》解题报告

1. 重复计数

1.1 题目描述

分数 100
作者 谷方明
单位 吉林大学

在一个有限的正整数序列中，有些数会多次重复出现。请你统计每个数的出现次数，然后按数字在序列中第一次出现的位置顺序输出数及其次数。

输入格式:
第 1 行，1 个整数 N，表示整数的个数，(1≤N≤50000)。
第 2 行，N 个正整数，每个整数 x 都满足 1 ≤ x ≤2000000000。

输出格式:
若干行，每行两个用一个空格隔开的数，第一个是数列中出现的数，第二个是该数在序列中出现的次数。

1.2 思路

该题关键在于 1.保证输出顺序；2.统计每个数字的出现次数；
因此可以考虑使用 multiset 记录每个数据的次数，使用一个数组记录输入顺序；

1.3 代码：

```cpp
#include<iostream>
#include<set>
inline int read() {
    int x = 0; int w = 1; register char c = getchar();
    for (; c ^ '-' && (c < '0' || c > '9'); c = getchar());
    if (c == '-') w = -1, c = getchar();
    for (; c >= '0' && c <= '9'; c = getchar()) x = (x << 3) + (x << 1) + c - '0';
    return x * w;
}
using namespace std;
const int maxn = 50005;
int dat[maxn],cnt[maxn];

int main() {
    multiset<int>ms;
    int n = 0,cnt=1,x;
    n = read();
    for (int i = 1; i <= n; i++) {
        x = read();
        if (ms.count(x) == 0)dat[cnt++] =x;
        ms.insert(x);
    }
    for (int i = 1; i < cnt; i++) {
        printf("%d %d", dat[i], (int)ms.count(dat[i]));
        if (i != cnt)printf("\n");
    }
}
```

2. 字符串的周期

2.1 题目描述

分数 100
作者 谷方明
单位 吉林大学

一个字符串可以看成由某个长度为 k 的前缀重复若干次得到，称 k 为该字符串的周期。
例如："abababab"以 2、4、8 为周期。
请计算一个字符串的最小周期。

输入格式:
输入包含多行。
每行包含一个字符串 s，s 至少 1 个字符，最多 1000000 个字符。
最后一行是一个点，表示输入结束，不必求解。
输入较大，建议使用 scanf。

输出格式:

多行，每行一个整数，对应每一行输入 s 的最小周期。
输入样例:
在这里给出一组输入。例如：
abcd
aaaa
abababab
.
输出样例:
在这里给出相应的输出。例如：
4
1
2

2.2 思路

使用 next 数组的方法大佬已详细给出，这里记录一种可以通过的朴素方法。
具体思路较直接：对一个长为 n 的字符串，从周期 i=1 开始，将字符串划分为 n/i 个子串，用第一个子串分别与所有子串比较，直到比较成功,输出；
优化：当 n%i==0 时，才进行比较，当 i>sqrt(n)时停止比较；

2.3 代码

```cpp
#include<iostream>
#include<string.h>

using namespace std;
const int maxl = 1e6 + 5;
char s[maxl];

inline bool cmps(char a1[], int s2, int len) {
    for (int i = 0, j = s2; i < len; i++, j++) {
        if (a1[i] != a1[j])return false;
    }
    return true;
}
int main() {
    while (1) {
        fgets(s, maxl, stdin);
        char* find = strchr(s, '\n');
        if (find) *find = '\0';
        if (s[0] == '.' && strlen(s) == 1)break;
        int len = strlen(s);
        for (int i = 1; i <= len; i++) {
            if (len % i != 0)continue;
            int j = 0, flag = 1;
            for (; j < len; j += i) {
                if (cmps(s, j, i) == 0) { flag = 0; break; }
            }
            if (flag && j == len) {
                printf("%d\n", i); break;
            }
        }
    }
}
```

3.交换次数

3.1 题目描述

分数 100
作者 谷方明
单位 吉林大学

序列 A 中有 N 个整数。

求对 A 进行冒泡排序发生的元素交换次数。

输入格式:
第一行输入整数 N(2<=N<=10^6).
接下来一行 N 个正整数数 A[i] (1≤i≤N ， A[i]<=10^6)。

输出格式:
一行，有一个整数，表示元素交换的次数。

输入样例:
在这里给出一组输入。例如：
4
2 4 3 1

输出样例:
在这里给出相应的输出。例如：
4

3.2 思路

归并排序求逆序对：在归并排序数组 a 的过程中，合并两个下标从 s 到 e 的相邻区间 a1、a2 时，当 a[j]（j>mid）并入新数组，整体减少的逆序对数量为 a1 中剩余元素的数量；当合并完成，局部的逆序对数量为 0;
因此只需要在合并过程中，每当有 a1(a2)中元素并入新数组，就累加 a2（a1）的剩余元素数;

3.3 代码

```cpp
#include<iostream>

inline int read() {
    int x = 0; int w = 1; register char c = getchar();
    for (; c ^ '-' && (c < '0' || c > '9'); c = getchar());
    if (c == '-') w = -1, c = getchar();
    for (; c >= '0' && c <= '9'; c = getchar()) x = (x << 3) + (x << 1) + c - '0';
    return x * w;
}
using namespace std;
const int maxn = 1e6 + 5;
int a[maxn];
long long cnt = 0;
void Merge(int* a, int* x, int t, int m, int n) {//t、m为第一个序列的头尾指针，n为第二个序列尾
    int f1 = t, f2 = m + 1, xp = t;
    while (f1 <= m && f2 <= n) {
        if (a[f1] <= a[f2]) {
            x[xp++] = a[f1++];
        }
        else {
            x[xp++] = a[f2++];
            cnt +=(long long) m - f1+1;//加上前一个文件剩余的元素数
        }
    }
    while (f1 <= m)x[xp++] = a[f1++];
    while (f2 <= n)x[xp++] = a[f2++];
}

void MPass(int* a, int len, int L, int* x) {//将所有长度小于等于L的两两合并
    int i = 0;
    for (; i + L * 2 - 1 <= len; i += L * 2) {//每次将L*2长内的序列两两合并
        Merge(a, x, i, i + L - 1, i + 2 * L - 1);
    }
    if (i + L - 1 < len) {
        Merge(a, x, i, i + L - 1, len);
    }
    else {
        for (int j = i; j <= len; j++)x[j] = a[j];
    }
}
//归并排序
void MSort(int* a, const int len) {
    int* x = new int[len + 1];
    for (int i = 1; i <= len; i *= 2) {
        MPass(a, len, i, x);
        i *= 2;
        MPass(x, len, i, a);
    }
    delete[] x;
}
int main() {
    int n = read();
    for (int i = 0; i < n; i++)
        a[i] = read();
    MSort(a, n-1);
    printf("%lld", cnt);
    return 0;
}
```

维护序列

4.1 题目描述

分数 100
作者 谷方明
单位 吉林大学

一个序列初始为空。给出 N(N<=1000000)个操作维护序列。

输入格式:
第 1 行 整数 N；
然后有 N 行，每行两个数，ch 和 k
ch=1 表示插入一个值为 k 的数
ch=2 表示查询第 k 小的数（k 合法）
ch=3 表示删除值为 k 的数(k 一定存在)

输出格式:
输出查询操作的结果值。每个一行。

输入样例:
在这里给出一组输入。例如：
5
1 2
1 3
2 1
3 2
2 1

输出样例:
在这里给出相应的输出。例如：
2
3

4.2 思路

本题使用伸展树第三个点运行超时；借鉴某位大佬，使用线段树；
因本题测试数据大小均在区间[0,1e6]中，可以使用线段树维护序列。
时间复杂度：
线段树插入、删除、找第 k 小的时间复杂度均为 O(logn)，故总体为 O(N*logn)

4.3 代码

```cpp
#include<iostream>

#define ls (i << 1)
#define rs (i << 1 | 1)
inline int read() {
    int x = 0; int w = 1; register char c = getchar();
    for (; c ^ '-' && (c < '0' || c > '9'); c = getchar());
    if (c == '-') w = -1, c = getchar();
    for (; c >= '0' && c <= '9'; c = getchar()) x = (x << 3) + (x << 1) + c - '0';
    return x * w;
}

using namespace std;
const int maxn = 1e6 + 5;


struct segment {
    struct node {
        int l, r, cnt;
    }st[maxn << 2];

    void build(int i, int l, int r) {
        st[i] = { l, r, 0 };

        if (l == r) return;
        int mid = (l + r) >> 1;
        build(ls, l, mid);
        build(rs, mid + 1, r);
    }

    inline void insert(int i, int val) {
        if (st[i].l == st[i].r) st[i].cnt++;
        else {
            if (val <= st[ls].r) insert(ls, val);
            else insert(rs, val);
            st[i].cnt++;
        }
    }
    inline void del(int i, int val) {
        if (st[i].l == st[i].r) st[i].cnt--;
        else {
            if (val <= st[ls].r)del(ls, val);
            else del(rs, val);
            st[i].cnt--;
        }
    }
```

截图(Alt + A)

```
    inline int find_kth(int i, int k) {
        if (st[i].l == st[i].r)return st[i].l;
        if (k <= st[ls].cnt)return find_kth(ls, k);
        else return find_kth(rs, k - st[ls].cnt);
    }

}ST;
int main() {
    int n = read(),cm,x,out;
    ST.build(1, 0, maxn);
    for (int i = 0; i < n; i++) {
        cm = read(); x = read();

        if (cm == 1) ST.insert(1, x);
        if (cm == 2) { out = ST.find_kth(1, x); printf("%d\n",out); }
        if (cm == 3) ST.del(1, x);
    }
    return 0;
}
```