树状数组题目讨论

chenguo

问题1：HH的项链

离线+树状数组

仔细观察题目，会发现题目要求的操作中没有修改操作，所以我们可以把所有询问存储(离线)下来，按照某种顺序处理。

由于我们是从左往右扫描，当扫描到一个询问的右端点的时候就要得到答案。所以我们按询问区间的右端点进行升序排序，如果右端点相同按左端点升序排序。

通过观察我们可以开一个树状数组来维护每个数字只出现一次的次数，如果某个数字出现多处，我们要以后面出现的为准，所以当发现一个数字在之前出现过了，我们就要把前面这次的影响消除掉，所以我们要维护每个数字上一次出现的位置。

详情见参考代码。

问题1： HH的项链

参考代码：

```
int main() {
    scanf("%d", &n);
    for (int i = 1; i <= n; i++) {
        scanf("%d", &t[i]);
    }
    cin >> m;
    for (int i = 1; i <= m; i++) {
        scanf("%d %d", &node[i].l, &node[i].r);
        node[i].id = i;
    }
    sort(node + 1, node + 1 + m, cmp);
    int j = 1;
    for (int i = 1; i <= m; i++) {
        while (j <= node[i].r) {
            if (pre[t[j]]) {//判断是否出现过
                add(pre[t[j]], -1);//若出现则去掉之前出去的值
            }
            pre[t[j]] = j;
            add(j, 1);
            j++;
        }
        ans[node[i].id] = getSum(node[i].r) - getSum(node[i].l - 1);
    }
    for (int i = 1; i <= m; i++) {
        printf("%d\n", ans[i]);
    }
    return 0;
}
```

问题2：序列操作

要求得所有数字的平均值，其实只需知道序列的长度和sum即可。

观察三种操作：
1. 长度不变、sum+=Ai*Xi
2. 长度+1，sum+=Ki
3. 长度-1，sum-=最后一个数字

故我们需要快速维护出序列的最后一个数。

回想我们对一段区间整体加1的处理方法。
L位置++， R+1位置--，然后求前缀和的办法。

我们可以把这种思想运用到树状数组中，详情参考代码。

问题2： 序列操作

```
void add(int i, int x) {
    for (; i <= n; i += i & -i) c[i] += x;
}
int sum(int i) {
    int s = 0;
    for (; i; i -= i & -i) s += c[i];
    return s;
}
int main() {
    int t, l = 1, a, x, k;
    double ans = 0;
    scanf("%d", &n);
    for(int i = 1; i <= n; i++) {
        scanf("%d", &t);
        if (t == 1) {
            scanf("%d%d", &a, &x);
            add(1, x), add(a + 1, -x);
            ans += a * x;
        } else if (t == 2) {
            scanf("%d", &k); ans += k; ++l;
            add(l, k); add(l + 1, -k);
        } else if (l >= 2) {
            k = sum(l); ans -= k;
            add(l, -k); add(l + 1, k);
            --l;
        }
        printf("%.6lf\n", ans / l);
    }
    return 0;
}
```

问题3: 走丢的奶牛

二分+树状数组

树状数组的 SUM(X) 用于记录 编号X后面满足小于等于 X 的已经用掉了的编号的个数；
F[i] 是题目给出的第 i 个牛前面比 第i 个牛的编号小的编号的个数；
我们需要二分的就是编号X，判断 X是不是当前第 i 头牛的编号。

如果 (X-1)-SUM(X - 1) == F[i] (即编号X 前面剩下的小于 X 的编号的数量恰好等于 第 i 头牛编号的条件， 则 X 就是 第 i 头牛的编号）
如果(X-1)-SUM(X - 1) > F[i]　（说明编号偏大）
如果(X-1)-SUM(X - 1) < F[i]　 (说明编号偏小）

问题3： 走丢的奶牛

参考代码：

```
int main() {
    scanf("%d", &N);
    num[1] = 0;
    for (int i = 2; i <= N; i++) scanf("%d", &f[i]); //前面比编号为i的数小的数的个数
    num[N] = f[N] + 1;
    add(num[N], 1);
    for (int i = N - 1; i > 0; i--) {
        int l = 1, r = N;
        while (r > l) {
            int mid = (l + r) >> 1;
            if (mid - 1 - sum(mid) >= f[i]) {
                r = mid;
            } else {
                l = mid + 1;
            }
        }
        num[i] = l;
        add(num[i], 1);
    }
    for (int i = 1; i <= N; i++) printf("%d\n", num[i]);
    return 0;
}
```

问题4: 何老板线

离线+树状数组

按照区间长度排序，小于间隔的最多覆盖一个点，那么树状数组差分解决，大于的肯定覆盖。

问题4： 何老板线

参考代码：

```
int main() {
    scanf("%d%d", &n, &m);
    for(int i = 1; i <= n; ++i) {
        scanf("%d%d", &a[i].l, &a[i].r);
        a[i].len = a[i].r - a[i].l + 1;
    }
    sort(a + 1, a + n + 1);
    for(int i = 1, j = 1; i <= m; ++i) {//枚举停靠间隔
        while(j <= n && a[j].len < i) {//区间长度小于间隔的插入bit
            add(a[j].l, 1);
            add(a[j].r + 1, -1);
            ++j;
        }
        int ans = n - j + 1;//区间长度大于间隔的必然能购买到
        for(int j = i; j <= m; j += i) {//j枚举列车依次停靠的站
            ans += query(j);
        }
        printf("%d\n", ans);
    }
    return 0;
}
```

问题5: 中位数的中位数

二分+树状数组

因为区间数量是平方级别的，所以我们正向的去求解肯定是不行的。

考虑把问题转化为判定性问题，我们二分最后答案中位数x，我们就需要判定**有多少个区间的中位数小于等于x。**

对于二分出的x，将原序列上所有大于x的数记为-1，小于等于x的记为1，问题进一步转化为：”**有多少个序列至少有（序列长度/2 + 1）个1**”，可以发现它等价于**“有多少个序列的和是正数”**。

在修改过后的原序列上做前缀和，序列[l, r]的和 = sum[r] - sum[l-1]。如果该式为正数，则sum[r] > sum[l-1]。
那么，对于所有的sum[x] > sum[y]，如果再满足x > y，则[y+1, x]这个序列就是一个合法序列。

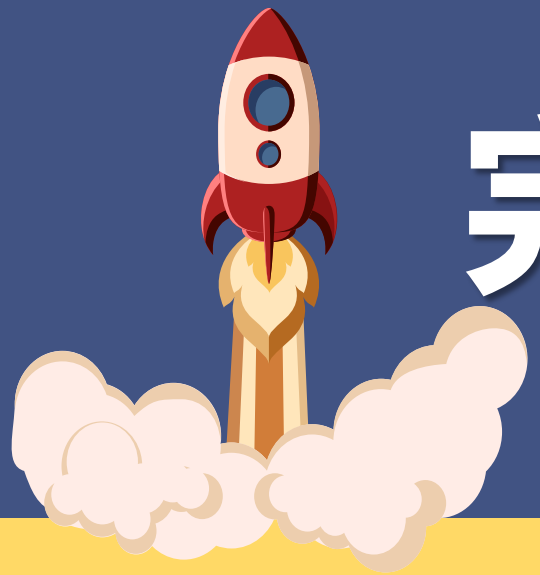
是不是就是在求解**顺序对，**我们可以用类似于逆序对的方法求。

注意：这里的树状数组不能有负数，所以得加个1e5强制转正（因为是下标）

问题5： 中位数的中位数

参考代码：

```
bool check(int x) {
    for (int i = 1; i <= 2 * N; i++) c[i] = 0;
    s[0] = 0;
    for (int i = 1; i <= n; i++) s[i] = s[i - 1] + (a[i] >= x ? 1 : -1);
    ll sum = 0;
    for (int i = 0; i <= n; i++) {
        sum += query(s[i] + N);
        add(s[i] + N);
    }
    return sum >= 1ll * n * (n + 1) / 4;
}
int main() {
    scanf("%d", &n);
    int l = 0, r = 0;
    for (int i = 1; i <= n; i++) {
        scanf("%d", &a[i]);
        r = max(r, a[i]);
    }
    int ans = 0;
    while (l <= r) {
        int mid = (l + r) >> 1;
        if (check(mid))
            l = mid + 1;
        else
            r = mid - 1;
    }
    printf("%d\n", r);
}
```



完！

以梦为码 心之所往