Problem-I

Description

给定长度为n的数列$\{a_i\}$和q次询问，每次询问给出l,r,k，你需要输出值v，满足$a_l\dots a_r$中等于v的数的数量$\geq \lceil \frac{r-l+1}{k} \rceil$（若存在多个则输出任意一个），若不存在，输出$-1$。

$n \leq 10^6, a_i \leq n, q \leq 10^4, 1 \leq k \leq 4$

Solution

随机算法

对于每一个$v = 1\dots n$，将所有满足$a_i = v$的i存在 `std::vector` 中，那么可以在 `std::vector` 中二分、在$O(\log n)$的时间内对于l,r,v求出区间$a_l\dots a_r$内满足$a_i = v$的数量。

对于询问l,r,k，我们在区间$[l,r]$内随机一个整数i，取出该位置的值a_i，查询$a_l\dots a_r$内有多少位置满足$a_j = a_i$，并检查该数量是否达到$\lceil \frac{r-l+1}{k} \rceil$，如果达到，则找到答案。

将这个随机过程重复T次，若仍未找到答案，那么认为无解、输出-1。

最坏情况下，我们每次询问都成功得到答案的概率是$(1-0.75^T)^q$，q取最大值10^4：

- 当$T=30$时，正确率为16.8%
- 当$T=35$时，正确率为65.5%
- 当$T=40$时，正确率为90.4%
- 当$T=50$时，正确率为99.4%

当$T=50$时基本可以保证答案正确。（实测T取35可过）

时间复杂度$O(qT\log n)$。

根号分类

设$B=\sqrt{n}$，对于询问l,r,k:

- 若$r-l+1 \leq B$，我们直接暴力计数，单次时间复杂度$O(B)$
- 若$r-l+1 > B$，若答案存在，则答案满足其出现次数$\geq \lceil \frac{r-l+1}{k} \rceil \geq \frac{B}{k}$，由于总数只有$n$，那么出现次数达到$\frac{B}{k}$的值的数量不超过$\frac{n}{\frac{B}{k}} = \frac{kn}{B} \leq 4\sqrt{n}$。我们可以将次数达到$\frac{B}{k}$的每种值都检查一遍，每次用$O(\log n)$的时间查询出现次数是否达到$\lceil \frac{r-l+1}{k} \rceil$，单次时间复杂度$O(\frac{kn}{B}\log n)$。

总时间复杂度为$O(q(B+\frac{kn}{B}\log n)) = O(q(B+\frac{n\log n}{B}))$，当$B=\sqrt{n}$时，时间复杂度为$O(q\sqrt{n}\log n)$。

令$B=\frac{n\log n}{B}$，得到$B=\sqrt{n\log n}$，此时时间复杂度为$O(q\sqrt{n\log n})$。

Code

Code1

```cpp
int n, a[MAXN], cnt[MAXN];
vector<int> pos[MAXN], CO;

int main() {
    n = read<int>();
    for (int i = 1; i <= n; ++ i)
        a[i] = read<int>(), pos[a[i]].PB(i);
    for (int m = read<int>(); m --; ) {
        int l = read<int>(), r = read<int>(), k = read<int>();
```

```
        int mk = 0;
        for (int T = 35; T --; ) {
            int co = a[l + rand() % (r - l + 1)];
            auto it1 = lower_bound(pos[co].begin(), pos[co].end(), l);
            if (it1 == pos[co].end()) continue;
            auto it2 = upper_bound(pos[co].begin(), pos[co].end(), r);
            if (it2 - it1 >= (r - l + 1 + (k - 1)) / k) {
                printf("%d\n", co);
                mk = 1;
                break;
            }
        }
        if (!mk) puts("-1");
    }
    return 0;
}
```

Code2

```
int n, a[MAXN], cnt[MAXN];
vector<int> pos[MAXN], CO;

int main() {
    n = read<int>();
    for (int i = 1; i <= n; ++ i)
        a[i] = read<int>(), pos[a[i]].PB(i);
    int B = 2000;
    for (int i = 1; i <= n; ++ i)
        if ((int)pos[i].size() > B / 4) CO.PB(i);
    for (int m = read<int>(); m --; ) {
        int l = read<int>(), r = read<int>(), k = read<int>();
        if (r - l + 1 <= B) {
            int id = 0;
            for (int i = l; i <= r; ++ i) {
                ++ cnt[a[i]];
                if (cnt[a[i]] > cnt[id]) id = a[i];
            }
            if (cnt[id] >= (r - l + 1 + (k - 1)) / k) printf("%d\n", id);
            else puts("-1");
            for (int i = l; i <= r; ++ i) -- cnt[a[i]];
        } else {
            int mk = 0;
            for (int co: CO) {
                auto it1 = lower_bound(pos[co].begin(), pos[co].end(), l);
                if (it1 == pos[co].end()) continue;
                auto it2 = upper_bound(pos[co].begin(), pos[co].end(), r);
                if (it2 - it1 >= (r - l + 1 + (k - 1)) / k) {
                    printf("%d\n", co);
                    mk = 1;
                    break;
                }
            }
            if (!mk) puts("-1");
        }
    }
    return 0;
}
```