

ST表

人员

杨俊彦、杨咏丞、刘奕辰、周治润、李锦澍、李雨谦、隋天乙、袁晨峻 到课

上周作业检查

上周作业链接: <https://cppoj.kids123code.com/contest/792>

2025-0914 周日8:30 (综合练习)										
<div>🔄 刷新</div>										
#	用户名	姓名	编程分	时间	A	B	C	D	E	F
1	liuyichen	刘奕辰	600	1263	100	100	100	100	100	100
2	lijinshu	李锦澍	600	1276	100	100	100	100	100	100
3	yangjunyan	杨俊彦	600	1389	100	100	100	100	100	100
4	yangyongcheng	杨咏丞	600	2153	100	100	100	100	100	100
5	liyuqian	李雨谦	600	2161	100	100	100	100	100	100
6	yuanchenjun	袁晨峻	500	1776	100	100	100	100	100	
7	wangluwenlong	王陆文龙	400	453	100	100	100	100		
8	zhouzhirun	周治润	400	921	100	100	100	100		
9	suitianyi	隋天乙	337	242	100	100	100		37	
10	xuruiqian	许睿谦	300	267	100	100	100		0	
11	caoyuan	曹源	160	157	100	60				
12	chenxinmiao	陈欣妙	100	149	100					
13	yuyue	于跃	30	0	30					

本周作业

<https://cppoj.kids123code.com/contest/845> (课上讲了 A ~ D 题, 课后作业是 E 题)

课堂表现

今天上课给同学们讲解了 ST 表, ST 表的思想不是很难, 同学们课上都听懂了, 但是要求同学们必须要把 ST 表的代码写的非常熟才可以

课堂内容

[CSP-S 2024] 决斗 (上周作业)

排序后, 双指针扫一遍即可

```
#include <bits/stdc++.h>

using namespace std;

const int maxn = 1e5 + 5;
int w[maxn];
```

```

int main()
{
    int n; cin >> n;
    for (int i = 1; i <= n; ++i) cin >> w[i];
    sort(w+1, w+n+1);

    int res = n;
    for (int i = 2, j = 1; i <= n; ++i) {
        if (w[i] > w[j]) --res, ++j;
    }
    cout << res << endl;
    return 0;
}

```

【模板】ST 表 & RMQ 问题

ST 表模板题

$f[i][k]$ 代表以 i 作为开头, 往后长度为 2^k 的区间中的最大值是多少

所以很明显, $f[i][0] = a[i]$, $f[i][k] = \max(f[i][k-1], f[i+(1 \ll (k-1))][k-1])$

因此, 可以 $n \log n$ 的时间复杂度预处理 f 数组, 后续可以 $O(1)$ 求区间最大值

```

#include <bits/stdc++.h>

using namespace std;

const int N = 1e5 + 5, M = 20;
int w[N], f[N][M], _lg2[N];

int get_max(int l, int r) {
    int len = r - l + 1;
    int k = _lg2[len];
    return max(f[l][k], f[r-(1<<k)+1][k]);
}

int main()
{
    for (int i = 0; (1<<i) < N; ++i) _lg2[1<<i] = i;
    for (int i = 1; i < N; ++i) {
        if (!_lg2[i]) _lg2[i] = _lg2[i-1];
    }

    int n, m; cin >> n >> m;
    for (int i = 1; i <= n; ++i) cin >> w[i], f[i][0] = w[i];
    for (int k = 1; k < M; ++k) {
        for (int i = 1; i+(1<<k)-1 <= n; ++i) {
            f[i][k] = max(f[i][k-1], f[i+(1<<(k-1))][k-1]);
        }
    }
}

```

```

while (m -- ) {
    int l, r; scanf("%d%d", &l, &r);
    cout << get_max(l,r) << "\n";
}
return 0;
}

```

忠诚

把模板中的找区间 max 替换为找区间 min 即可

```

#include <bits/stdc++.h>

using namespace std;

const int N = 1e5 + 5, M = 20;
int w[N], f[N][M], _lg2[N];

int get_min(int l, int r) {
    int len = r - l + 1;
    int k = _lg2[len];
    return min(f[l][k], f[r-(1<<k)+1][k]);
}

int main()
{
    for (int i = 0; (1<<i) < N; ++i) _lg2[1<<i] = i;
    for (int i = 1; i < N; ++i) {
        if (!_lg2[i]) _lg2[i] = _lg2[i-1];
    }

    int n, m; cin >> n >> m;
    for (int i = 1; i <= n; ++i) cin >> w[i], f[i][0] = w[i];
    for (int k = 1; k < M; ++k) {
        for (int i = 1; i+(1<<k)-1 <= n; ++i) {
            f[i][k] = min(f[i][k-1], f[i+(1<<(k-1))][k-1]);
        }
    }

    while (m -- ) {
        int l, r; cin >> l >> r;
        cout << get_min(l,r) << " ";
    }
    cout << endl;
    return 0;
}

```

[蓝桥杯 2023 国 C] 最大区间

枚举每个 i 作为最小值, 就需要让 L 尽量靠左, R 尽量靠右, 同时满足 $L \sim R$ 区间中的最小值是 $a[i]$ 才可以

找 L 和找 R 的过程可以用 二分套ST表 完成

```
#include <bits/stdc++.h>

using namespace std;

typedef long long LL;
const int N = 3e5 + 5, M = 20;
int w[N], f[N][M], _lg2[N];
int n;

int get_min(int l, int r) {
    int len = r - l + 1;
    int k = _lg2[len];
    return min(f[l][k], f[r-(1<<k)+1][k]);
}

int find_L(int id) {
    int l = 1, r = id;
    while (l <= r) {
        int mid = (l + r) / 2;
        if (get_min(mid, id) == w[id]) r = mid-1;
        else l = mid+1;
    }
    return l;
}

int find_R(int id) {
    int l = id, r = n;
    while (l <= r) {
        int mid = (l + r) / 2;
        if (get_min(id, mid) == w[id]) l = mid+1;
        else r = mid-1;
    }
    return r;
}

LL calc(int id) {
    int l = find_L(id), r = find_R(id);
    return (LL)w[id] * (r-l+1);
}

int main()
{
    for (int i = 0; (1<<i) < N; ++i) _lg2[1<<i] = i;
    for (int i = 1; i < N; ++i) {
        if (!_lg2[i]) _lg2[i] = _lg2[i-1];
    }

    cin >> n;
    for (int i = 1; i <= n; ++i) cin >> w[i], f[i][0] = w[i];
    for (int k = 1; k < M; ++k) {
        for (int i = 1; i+(1<<k)-1 <= n; ++i) {
```

```

        f[i][k] = min(f[i][k-1], f[i+(1<<(k-1))][k-1]);
    }
}

LL res = 0;
for (int i = 1; i <= n; ++i) res = max(res, calc(i));
cout << res << endl;
return 0;
}

```

[JRKSJ R2] 01 序列

查询1: 在 $l \sim r$ 中找一个 i , 让 $pre0[i] - pre0[l-1] + suf1[i] - suf1[r+1]$ 最大, 即在 $l \sim r$ 中找一个最大的 $pre0[i] + suf1[i]$, 可以用 ST 表查询

查询2: 判断 $l \sim r$ 区间中是否存在一个 01 的连续段, 可以用前缀和维护

```

#include <bits/stdc++.h>

using namespace std;

typedef long long LL;
const int N = 1e6 + 5, M = 22;
int w[N], f[N][M], _lg2[N];
int pre0[N], suf1[N];
int p01[N];

int get_max(int l, int r) {
    int len = r - l + 1;
    int k = _lg2[len];
    return max(f[l][k], f[r-(1<<k)+1][k]);
}

int main()
{
    ios::sync_with_stdio(false);
    cin.tie(0); cout.tie(0);

    for (int i = 0; (1<<i) < N; ++i) _lg2[1<<i] = i;
    for (int i = 1; i < N; ++i) {
        if (!_lg2[i]) _lg2[i] = _lg2[i-1];
    }

    int n, m; cin >> n >> m;
    for (int i = 1; i <= n; ++i) cin >> w[i];
    for (int i = 1; i <= n; ++i) pre0[i] = pre0[i-1] + (w[i]==0);
    for (int i = n; i >= 1; --i) suf1[i] = suf1[i+1] + (w[i]==1);
    for (int i = 1; i <= n; ++i) f[i][0] = pre0[i] + suf1[i];

    for (int i = 2; i <= n; ++i) p01[i] = p01[i-1] + (w[i-1]==0&&w[i]==1);
}

```

```
for (int k = 1; k < M; ++k) {
    for (int i = 1; i+(1<<k)-1 <= n; ++i) {
        f[i][k] = max(f[i][k-1], f[i+(1<<(k-1))][k-1]);
    }
}

while (m -- ) {
    int op, l, r; cin >> op >> l >> r;
    if (op == 1) {
        cout << get_max(l,r) - pre0[l-1] - suf1[r+1] << "\n";
    } else {
        cout << (p01[r]==p01[l] ? 1 : 2) << "\n";
    }
}
return 0;
}
```