

线段树区间修改

人员

王毅博、褚锦轩、王承周、许睿谦、阮文璋 到课

上周作业检查

上周作业链接: <https://cppoj.kids123code.com/contest/1588>

比赛概况 题目列表 选择题列表 提交记录 ★ 实时榜单 ★ 选择题排行榜								
王向东老师周日一点半C++线段树								
刷新								
#	用户名	姓名	编程分	时间	A	B	C	D
1	wangyibo	王毅博	400	2440	100	100	100	100
2	ruanwenzhang	阮文璋	400	2891	100	100	100	100
3	xuruiqian	许睿谦	300	808	100	100		100
4	chujinxuan	褚锦轩	300	3300	100	100	100	
5	wangchengzhou	王承周	300	3722	100	100	100	

本周作业

<https://cppoj.kids123code.com/contest/1689> (课上讲了 A ~ C 题, 课后作业是 D 题)

课堂表现

今天的 C 题会比较复杂一些, 需要同时维护 乘 和 加 两个懒标记, 大部分同学课上都没写完, 课下需要把这道题沉住气好好写一写。

课堂内容

Dice Product 3 (上周作业)

直接记忆化搜索即可, 时间复杂度其实就是 n 的因数级别的

```
#include <bits/stdc++.h>
#define int long long

using namespace std;

const int mod = 998244353;

int qmod(int a, int k) {
    int res = 1;
    while (k) {
        if (k&1) res = res*a % mod;
        a = a*a % mod;
        k >>= 1;
    }
}
```

```

    }
    return res;
}
int inv(int x) { return qmod(x, mod-2); }

int inv_5;

map<int, int> mp;
int dfs(int n) {
    if (mp.count(n)) return mp[n];

    int res = 0;
    for (int i = 2; i <= 6; ++i) {
        if (n%i == 0) res += dfs(n/i)*inv_5, res %= mod;
    }
    mp[n] = res;

    return mp[n];
}

signed main()
{
    int n; cin >> n;
    inv_5 = inv(5); mp[1] = 1;
    cout << dfs(n) << endl;
    return 0;
}

```

【模板】线段树 1

线段树区间修改模板题, 用一个懒标签加速区间修改操作, 额外维护一个 pushdown 操作

```

#include <bits/stdc++.h>
#define int long long

using namespace std;

const int maxn = 1e5 + 5;
struct node {
    int l, r, sum, add;
    int len;
} tr[maxn*4];
int w[maxn];

void pushup(int u) { tr[u].sum = tr[u*2].sum + tr[u*2+1].sum; }

void pushdown(int u) {
    if (tr[u].add) {
        tr[u*2].add += tr[u].add, tr[u*2].sum += tr[u].add * tr[u*2].len;
        tr[u*2+1].add += tr[u].add, tr[u*2+1].sum += tr[u].add * tr[u*2+1].len;
        tr[u].add = 0;
    }
}

```

```

    }
}

void build(int u, int l, int r) {
    tr[u] = {l, r, 0, 0, r-l+1};
    if (l == r) { tr[u].sum = w[l]; return; }
    int mid = (l + r) / 2;
    build(u*2, l, mid), build(u*2+1, mid+1, r);
    pushup(u);
}

void modify(int u, int l, int r, int k) {
    if (tr[u].l >= l && tr[u].r <= r) {
        tr[u].add += k, tr[u].sum += k * tr[u].len; return;
    }

    pushdown(u);
    int mid = (tr[u].l + tr[u].r) / 2;
    if (l <= mid) modify(u*2, l, r, k);
    if (r > mid) modify(u*2+1, l, r, k);
    pushup(u);
}

int query(int u, int l, int r) {
    if (tr[u].l >= l && tr[u].r <= r) return tr[u].sum;

    pushdown(u);
    int mid = (tr[u].l + tr[u].r) / 2, res = 0;
    if (l <= mid) res += query(u*2, l, r);
    if (r > mid) res += query(u*2+1, l, r);
    return res;
}

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

    while (m -- ) {
        int op, l, r; cin >> op >> l >> r;
        if (op == 1) {
            int k; cin >> k; modify(1, l, r, k);
        } else cout << query(1, l, r) << endl;
    }
    return 0;
}

```

[CQOI2006] 简单题

给每个区间维护一个懒标签, 代表这个区间被翻转过奇数次还是偶数次

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

using namespace std;

const int maxn = 1e5 + 5;
struct node {
    int l, r, add;
} tr[maxn*4];

void pushdown(int u) {
    if (tr[u].add) {
        tr[u*2].add ^= 1, tr[u*2+1].add ^= 1;
        tr[u].add = 0;
    }
}

void build(int u, int l, int r) {
    tr[u] = {l, r};
    if (l == r) return;
    int mid = (l + r) / 2;
    build(u*2, l, mid), build(u*2+1, mid+1, r);
}

void modify(int u, int l, int r) {
    if (tr[u].l>=l && tr[u].r<=r) { tr[u].add ^= 1; return; }

    pushdown(u);
    int mid = (tr[u].l + tr[u].r) / 2;
    if (l <= mid) modify(u*2, l, r);
    if (r > mid) modify(u*2+1, l, r);
}

int query(int u, int pos) {
    if (tr[u].l==pos && tr[u].r==pos) return tr[u].add;

    pushdown(u);
    int mid = (tr[u].l + tr[u].r) / 2;
    if (pos <= mid) return query(u*2, pos);
    return query(u*2+1, pos);
}

int main()
{
    int n, m; cin >> n >> m;
    build(1, 1, n);

    while (m -- ) {
        int op; cin >> op;
        if (op == 1) {
            int l, r; cin >> l >> r; modify(1, l, r);
        } else {
            int x; cin >> x;
            cout << query(1, x) << endl;
        }
    }
}
```

```

    }
}
return 0;
}

```

【模板】线段树 2

稍微复杂的线段树操作, 需要维护 2 个懒标签, 分别代表区间整体乘了多少, 以及区间整体加了多少

```

#include <bits/stdc++.h>
#define int long long

using namespace std;

const int maxn = 1e5 + 5;
int mod;
struct node {
    int l, r, sum, mul, add, len;
} tr[maxn*4];
int w[maxn];

void pushup(int u) { tr[u].sum = (tr[u*2].sum + tr[u*2+1].sum) % mod; }

void pushdown(int u) {
    tr[u*2].mul = (tr[u*2].mul*tr[u].mul)%mod;
    tr[u*2].add = (tr[u*2].add*tr[u].mul + tr[u].add) % mod;
    tr[u*2].sum = (tr[u*2].sum*tr[u].mul + tr[u*2].len*tr[u].add) % mod;
    tr[u*2+1].mul = (tr[u*2+1].mul*tr[u].mul)%mod;
    tr[u*2+1].add = (tr[u*2+1].add*tr[u].mul + tr[u].add) % mod;
    tr[u*2+1].sum = (tr[u*2+1].sum*tr[u].mul + tr[u*2+1].len*tr[u].add) % mod;
    tr[u].mul = 1, tr[u].add = 0;
}

void build(int u, int l, int r) {
    tr[u] = {l, r, 0, 1, 0, r-l+1};
    if (l == r) { tr[u].sum = w[l] % mod; return; }
    int mid = (l + r) / 2;
    build(u*2, l, mid), build(u*2+1, mid+1, r);
    pushup(u);
}

void modify(int u, int l, int r, int a, int b) {
    if (tr[u].l>=l && tr[u].r<=r) {
        tr[u].mul = (tr[u].mul*a)%mod, tr[u].add = (tr[u].add*a + b)%mod;
        tr[u].sum = (tr[u].sum*a + tr[u].len*b)%mod;
        return;
    }
    pushdown(u);
    int mid = (tr[u].l + tr[u].r) / 2;
    if (l <= mid) modify(u*2, l, r, a, b);

```

```
    if (r > mid) modify(u*2+1, l, r, a, b);
    pushup(u);
}

int query(int u, int l, int r) {
    if (tr[u].l>=l && tr[u].r<=r) return tr[u].sum;

    pushdown(u);
    int mid = (tr[u].l + tr[u].r) / 2, res = 0;
    if (l <= mid) res = (res + query(u*2, l, r)) % mod;
    if (r > mid) res = (res + query(u*2+1, l, r)) % mod;
    return res;
}

signed main()
{
    int n, m; cin >> n >> m >> mod;
    for (int i = 1; i <= n; ++i) cin >> w[i];
    build(1, 1, n);

    while (m -- ) {
        int op, l, r; cin >> op >> l >> r;
        if (op == 1) {
            int k; cin >> k; modify(1, l, r, k, 0);
        } else if (op == 2) {
            int k; cin >> k; modify(1, l, r, 1, k);
        } else cout << query(1, l, r) << endl;
    }
    return 0;
}
```