SJTU 4124

这是一道非常强力的线段树题目。

首先考虑维护什么。我们用 $res_{i,k}$ 表示一个从节点 i 对应的区间中选 k 的球的总得分。显然，对于一个节点 i，$res_{i,1},\cdots,res_{i,10}$ 全都要维护。

然后考虑怎么把答案凑出来。对于 i，res_i 等于左儿子 res、右儿子 res、两个儿子 res 的卷积三者的和。这是因为从 i 对应的区间中取数，这些数要么全部来自一个儿子，要么来自左边的一部分和右边的一部分。

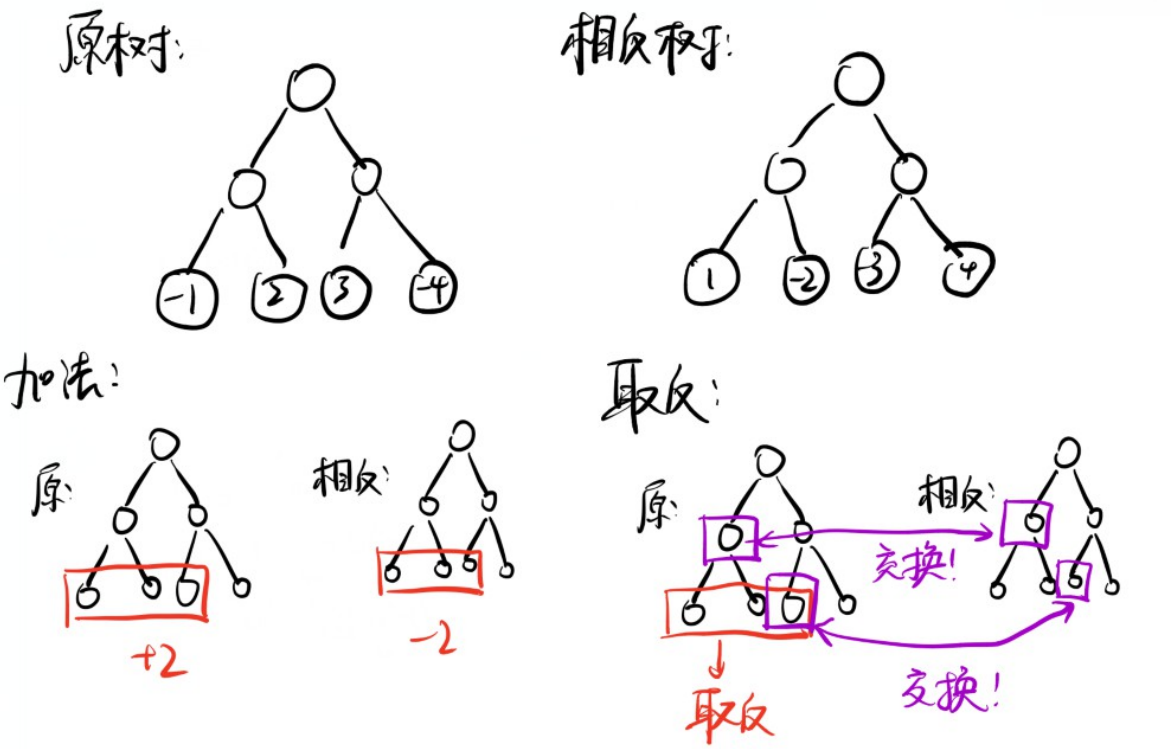
在考虑如何处理这些操作。对于加法操作，维护一个加法标记。然后比较麻烦的事情在于加法之后怎么更新 res。这里直接给出更新方法：如果对区间长度为 len 的节点 i 加了 e，那么

$$res_{i,k}=\sum_{t=0}^{k}e^t res_{i,k-t}\binom{len-k+t}{t}$$

具体怎么来的可以手动推一下。

对于相反数操作，我们也维护一个标记。似乎可以通过规定取反和加法的顺序实现维护。但是困难之处在于对 res 的更新，直接对 res 取反是错误的！因为 res 是若干个积的和，积的符号受到乘数的奇偶性的影响！

为了解决这个问题，我们额外维护一棵**相反的树**，即一开始保存的就是和原来的树相反的值，加法操作也加相反的数。这样的话，对一个区间取相反数就等价于**将对应区间的节点和相反树上的节点进行交换**！如图所示。



这样整个问题就顺利解决了。

整个算法的时间复杂度很高，但居然还是能跑得动，也是比较神奇了。

```
#include <bits/stdc++.h>
#define INF 2000000000
```

```cpp
#define MAXN 131100
#define M 1000000007
using namespace std;
typedef long long ll;
int read(){
    int f = 1, x = 0;
    char c = getchar();
    while(c < '0' || c > '9'){if(c == '-') f = -f; c = getchar();}
    while(c >= '0' && c <= '9')x = x * 10 + c - '0', c = getchar();
    return f * x;
}
struct Q{
    int arr[11];
    Q(){
        for (int i = 1; i <= 10; ++i)
            arr[i] = 0;
    }
};
int n, m, siz;
int fac[MAXN], inv[MAXN], invfac[MAXN];
int logg[MAXN] = {0};
int CC[MAXN][11] = {0}, tmp[11];
bool xfs[MAXN] = {0}, xfs_neg[MAXN] = {0};
int add[MAXN] = {0}, add_neg[MAXN] = {0};
Q res[MAXN], res_neg[MAXN];
int _a, _b, _v, opt;

inline int modadd(int x, int y){
    return (x + y >= M ? x + y - M: x + y);
}
inline int takexfs(int x){
    return (x == 0 ? 0: M - x);
}
inline int C(int n, int m){
    return 1ll * (1ll * fac[n] * invfac[m] % M) * invfac[n - m] % M;
}

void maintain(int i){
    for (int j = 1; j <= 10; ++j){
        res[i].arr[j] = modadd(res[i << 1].arr[j], res[i << 1 |
1].arr[j]);
        res_neg[i].arr[j] = modadd(res_neg[i << 1].arr[j], res_neg[i << 1
| 1].arr[j]);
        for (int t = 1; t < j; ++t)
            res[i].arr[j] = modadd(1ll * res[i << 1].arr[t] * res[i << 1 |
1].arr[j - t] % M, res[i].arr[j]),
            res_neg[i].arr[j] = modadd(1ll * res_neg[i << 1].arr[t] *
res_neg[i << 1 | 1].arr[j - t] % M, res_neg[i].arr[j]);
    }
}
void modify(int id, int len, int x, int x_neg){
    for (int i = 1; i <= 10; ++i)
        tmp[i] = res[id].arr[i];
    tmp[0] = 1;
    for (int i = 1; i <= 10; ++i){
        int fin_res = 0;
        for (int j = 0, xx = 1; j <= i; ++j, xx = 1ll * xx * x % M)
```

```
            fin_res = modadd(fin_res, 1ll * (1ll * xx * tmp[i - j] % M) * CC[len - i + j][j] % M);
        res[id].arr[i] = fin_res;
    }

    for (int i = 1; i <= 10; ++i)
        tmp[i] = res_neg[id].arr[i];
    tmp[0] = 1;
    for (int i = 1; i <= 10; ++i){
        int fin_res = 0;
        for (int j = 0, xx = 1; j <= i; ++j, xx = 1ll * xx * x_neg % M)
            fin_res = modadd(fin_res, 1ll * (1ll * xx * tmp[i - j] % M) * CC[len - i + j][j] % M);
        res_neg[id].arr[i] = fin_res;
    }
}
void pushdown(int id, int len){
    if (xfs[id]){
        xfs[id << 1] = !xfs[id << 1], xfs[id << 1 | 1] = !xfs[id << 1 | 1];
        xfs_neg[id << 1] = !xfs_neg[id << 1], xfs_neg[id << 1 | 1] = !xfs_neg[id << 1 | 1];
        swap(add[id << 1], add_neg[id << 1]), swap(add[id << 1 | 1], add_neg[id << 1 | 1]);
        for (int i = 1; i <= 10; ++i)
            swap(res[id << 1].arr[i], res_neg[id << 1].arr[i]),
            swap(res[id << 1 | 1].arr[i], res_neg[id << 1 | 1].arr[i]);
        xfs[id] = false;
    }
    if (add[id] != 0){
        add[id << 1] = modadd(add[id << 1], add[id]);
        add[id << 1 | 1] = modadd(add[id << 1 | 1], add[id]);
        add_neg[id << 1] = modadd(add_neg[id << 1], add_neg[id]);
        add_neg[id << 1 | 1] = modadd(add_neg[id << 1 | 1], add_neg[id]);
        modify(id << 1, len, add[id], add_neg[id]);
        modify(id << 1 | 1, len, add[id], add_neg[id]);
        add[id] = add_neg[id] = 0;
    }
}
void update(int id, int l, int r){
    if (l > _b || r < _a) return ;
    if (l >= _a && r <= _b){
        if (opt == 1){
            // add
            add[id] = modadd(add[id], _v);
            int neg_v = takexfs(_v);
            add_neg[id] = modadd(add_neg[id], neg_v);
            modify(id, (r - l + 1), _v, neg_v);
        }else {
            // takes minus
            swap(add[id], add_neg[id]);
            xfs[id] = !xfs[id];
            xfs_neg[id] = !xfs_neg[id];
            for (int i = 1; i <= 10; ++i)
                swap(res[id].arr[i], res_neg[id].arr[i]);
        }
        return ;
    }
```

```
    int mid = (l + r) >> 1;
    pushdown(id, mid - l + 1);
    if (_a <= mid) update(id << 1, l, mid);
    if (_b > mid) update(id << 1 | 1, mid + 1, r);
    maintain(id);
}
Q query(int id, int l, int r){
    if (l > _b || r < _a) return Q();
    if (l >= _a && r <= _b) return res[id];
    int mid = (l + r) >> 1;
    pushdown(id, mid - l + 1);
    if (_b <= mid) return query(id << 1, l, mid);
    if (_a > mid) return query(id << 1 | 1, mid + 1, r);
    Q lres = query(id << 1, l, mid), rres = query(id << 1 | 1, mid + 1,
r);
    Q curres;
    curres.arr[1] = modadd(lres.arr[1], rres.arr[1]);
    for (int j = 2; j <= 10; ++j){
        curres.arr[j] = modadd(lres.arr[j], rres.arr[j]);
        for (int t = 1; t < j; ++t)
            curres.arr[j] = modadd(1ll * lres.arr[t] * rres.arr[j - t] %
M, curres.arr[j]);
    }
    return curres;
}
void init(){
    n = read(), m = read();
    for (siz = 1; siz < n; siz <<= 1) ;
    for (int i = 1; i <= n; ++i){
        int a = read();
        if (a < 0) a += M;
        res[i + siz - 1].arr[1] = a;
        res_neg[i + siz - 1].arr[1] = takexfs(a);
    }
    for (int i = siz - 1; i >= 1; --i)
        maintain(i);

    // build C
    fac[1] = inv[1] = invfac[1] = 1;
    fac[0] = invfac[0] = 1;
    for (int i = 2; i <= siz; ++i)
        inv[i] = 1ll * (M - M / i) * inv[M % i] % M,
        fac[i] = 1ll * fac[i - 1] * i % M,
        invfac[i] = 1ll * invfac[i - 1] * inv[i] % M;
    CC[0][0] = 1;
    for (int i = 1, j = 1; i <= siz; i <<= 1, ++j){
        logg[i] = j;
        for (int u = i; u >= i - 10 && u > 0; --u)
            for (int t = min(10, u); t >= 0; --t)
                CC[u][t] = C(u, t);
    }
}
void solve(){
    while (m--) {
        opt = read();
        if (opt == 1){
            _a = read(), _b = read(), _v = read();
            if (_v < 0) _v += M;
```

```
            update(1, 1, siz);
        }
        if (opt == 2){
            _a = read(), _b = read();
            update(1, 1, siz);
        }
        if (opt == 3){
            _a = read(), _b = read(), _v = read();
            Q curres = query(1, 1, siz);
            printf("%d\n", curres.arr[_v]);
        }
    }
}
int main(){
    init();
    solve();
    return 0;
}
```