数据结构

1.树状数组

```
class BIT
{
    int n = 2e6;
    long long *a;

  public:
    BIT(int size) : n(size)
    {
        a = new long long[size + 10];
    }
    void update(int p, long long x)
    {
        while (p <= n)
            a[p] += x, p += (p & (-p));
    }

    long long query(int l, int r)
    {
        long long ret = 0;
        l--;
        while (r > 0)
            ret += a[r], r -= (r & (-r));
        while (l > 0)
            ret -= a[l], l -= (l & (-l));
        return ret;
    }
};
```

2. 并查集

```
#include <cassert>
#include <iostream>
#include <set>
/* Last modified: 23/08/01 */
class DSU
{
  private:
    int *f;
    int size;

  public:
    DSU(int size) : size(size)
```

```
    {
        assert(size > 1);
        f = new int[size + 10];
        for (int i = 1; i <= size; i++)
            f[i] = i;
    }
    int find(int x)
    {
        return f[x] == x ? x : (f[x] = find(f[x]));
    };
    bool same(int x, int y)
    {
        return find(x) == find(y);
    };
    bool merge(int x, int y)
    {
        int fx = find(x), fy = find(y);
        return ((fx != fy) ? f[fx] = fy : false);
    };
    int count()
    {
        std::set<int> s;
        for (int i = 1; i <= size; i++)
            s.insert(find(i));
        return s.size();
    }
};
```

3. ST表

```
// log2(x) 的预处理
// 1. 递推
lg[2] = 1;
for (int i = 3; i < N; i++)
    lg[i] = lg[i / 2] + 1;
// 2. 基于编译期计算
using std::array;
// WARNING: LOG_SIZE may cause CE if too big.
const int LOG_SIZE = 1e5 + 10;
constexpr array<int, LOG_SIZE> LOG = []() {
    array<int, LOG_SIZE> l{0, 0, 1};
    for (int i = 3; i < LOG_SIZE; i++)
        l[i] = l[i / 2] + 1;
    return l;
}();
// 3. 直接计算
int lg(int x)
{
    return 31 - __builtin_clz(x);
}
// STL 提供了 std::lg(), 底数是e.
```

```
class SparseTable
{
  private:
    // SIZE depends on range of f[i][0].
    // 22 is suitable for 1e5.
    static const int SIZE = 22;
    // f[i][j] maintains the result from i to i + 2 ^ j - 1;
    int (*f)[SIZE];
    using func = std::function<int(int, int)>;
    func op;
    // length of f from 1 to l;
    int l;

  public:
    SparseTable(int a[][SIZE], func foo, int len) : f(a), op(foo), l(len)
    {
        for (int j = 1; j < SIZE; j++)
            for (int i = 1; i + (1 << j) - 1 <= len; i++)
                // f[i][j] comes from f[i][j - 1].
                // f[i][j - 1], f[i + 2^(j - 1)] cover the range of f[i][j].
                f[i][j] = foo(f[i][j - 1], f[i + (1 << (j - 1))][j - 1]);
    };
    int query(int x, int y)
    {
        int s = LOG[y - x + 1];
        return op(f[x][s], f[y - (1 << s) + 1][s]);
    }
};
```

4. 线段树

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

using namespace std;
#define IOS ios::sync_with_stdio(false), cin.tie(nullptr), cout.tie(nullptr);
#define int long long
#define ull unsigned long long
#define lowbit(i) ((i) & (-i))
#define ls(p) (p << 1)
#define rs(p) (p << 1 | 1)
#define rep(i, a, b) for (int i = a; i <= b; i++)
#define per(i, a, b) for (int i = a, i >= b, i--)

typedef pair<int, int> PII;
const int mod = 1e9 + 7;
const int inf = 0x3f3f3f3f;
const int N = 1e5 + 200;
int qpow(int a, int n)
{
    int ans = 1;
    while (n)
```

```
    {
        if (n & 1)
        {
            ans = ans * a % mod;
        }
        a = a * a % mod;
        n >>= 1;
    }
    return ans;
}
int a[N];
int tag[4 * N];
int tree[4 * N];
int n;
void push_up(int p)
{
    tree[p] = tree[ls(p)] + tree[rs(p)];
}
void build(int p, int l, int r)
{
    if (l == r)
    {
        tree[p] = a[l];
        return;
    }
    int mid = (l + r) >> 1;
    build(ls(p), l, mid);
    build(rs(p), mid + 1, r);
    push_up(p);
}
void push_down(int p, int l, int r)
{
    int mid = (l + r) >> 1;
    tag[ls(p)] += tag[p];
    tag[rs(p)] += tag[p];
    tree[ls(p)] += tag[p] * (mid - l + 1);
    tree[rs(p)] += tag[p] * (r - mid);
    tag[p] = 0;
}
void update(int nl, int nr, int k, int p = 1, int l = 1, int r = n)
{
    if (nl <= l && r <= nr)
    {
        tag[p] += k;
        tree[p] += k * (r - l + 1);
        return;
    }
    push_down(p, l, r);
    int mid = (l + r) >> 1;
    if (nl <= mid)
        update(nl, nr, k, ls(p), l, mid);
    if (nr > mid)
```

```
        update(nl, nr, k, rs(p), mid + 1, r);
    push_up(p);
}
int query(int x, int y, int l = 1, int r = n, int p = 1)
{
    int res = 0;
    if (x <= l && y >= r)
        return tree[p];
    int mid = (l + r) >> 1;
    push_down(p, l, r);
    if (x <= mid)
        res += query(x, y, l, mid, ls(p));
    if (y > mid)
        res += query(x, y, mid + 1, r, rs(p));
    return res;
}
signed main()
{
    IOS int q;
    cin >> n >> q;
    for (int i = 1; i <= n; i++)
        cin >> a[i];
    build(1, 1, n);
    while (q--)
    {
        int op, x, y, k;
        cin >> op;
        if (op == 1)
        {
            cin >> x >> y >> k;
            update(x, y, k);
        }
        else
        {
            cin >> x >> y;
            cout << query(x, y) << endl;
        }
    }
    return 0;
}
```