分块

基础分块模板

```
#include <bits/stdc++.h>
using namespace std;

typedef long long ll;
const int N = 1e5 + 10, INF = 0x3f3f3f3f, MOD = 10007;

int len, n, st[N], ed[N], bel[N];
ll a[N];

void update(int l, int r, int c){
    if(bel[l] == bel[r]){
        for(int i = l; i <= r; i++){
            //  暴力更新
        }
        return ;
    }
    int i = l, j = r;
    while(bel[i] == bel[l]){
        //  暴力更新
        i++;
    }
    while(bel[j] == bel[r]){
        //  暴力更新
        j--;
    }

    for(int k = bel[i]; k <= bel[j]; k++){
       //   对块操作
    }
}

int query(int l, int r, int c){
    int ans = 0;
    if(bel[l] == bel[r]){
        for(int i = l; i <= r; i++){
            //  暴力更新
        }
        return ans;
    }
    int i = l, j = r;
    while(bel[i] == bel[l]){
        //  暴力更新
        i++;
    }
    while(bel[j] == bel[r]){
        //  暴力更新
        j--;
    }
```

```
    for(int k = bel[i]; k <= bel[j]; k++){
        //  对块操作
    }
    return ans;
}

int main(){
    cin >> n;
    len = sqrt(n);
    for(int i = 1; i <= n; i++){
        cin >> a[i];
    }
    for(int i = 1; i <= len; i++){
        st[i] = n / len * (i - 1) + 1;
        ed[i] = (i != len ? n / len * i : n);
        for(int j = st[i]; j <= ed[i]; j++){
            bel[j] = i;
        }
    }

    for(int i = 1; i <= n; i++){
        int op, l, r, c;
        cin >> op >> l >> r >> c;
        if(op == 0) update(l, r, c);
        else cout << query(l, r, c) << '\n';
    }
    return 0;
}
```

带push_down的分块

```
#include <bits/stdc++.h>
using namespace std;

typedef long long ll;
typedef pair<ll, ll> PLL;
const int N = 1e5 + 10, INF = 0x3f3f3f3f, MOD = 10007;

int len, n, st[N], ed[N], bel[N];
ll a[N];

void push_down(int k){
    for(int j = st[k]; j <= ed[k]; j++){
        //  暴力更新
    }
    //  初始化标记
}

void update(int l, int r, int c){
    if(bel[l] == bel[r]){
        push_down(bel[l]);
        for(int i = l; i <= r; i++){
            //  暴力更新
        }
```

```
        return ;
    }
    int i = l, j = r;
    push_down(bel[l]);
    while(bel[i] == bel[l]){
        //  暴力更新
        i++;
    }
    push_down(bel[r]);
    while(bel[j] == bel[r]){
        //  暴力更新
        j--;
    }

    for(int k = bel[i]; k <= bel[j]; k++){
        //  对块操作
    }
}

int query(int l, int r, int c){
    int ans = 0;
    if(bel[l] == bel[r]){
        push_down(bel[l]);
        for(int i = l; i <= r; i++){
            //  暴力更新
        }
        return ans;
    }
    int i = l, j = r;
    push_down(bel[l]);
    while(bel[i] == bel[l]){
        //  暴力更新
        i++;
    }
    push_down(bel[r]);
    while(bel[j] == bel[r]){
        //  暴力更新
        j--;
    }
    for(int k = bel[i]; k <= bel[j]; k++){
        //  对块操作
    }
    return ans;
}

int main(){
    cin >> n;
    len = sqrt(n);
    for(int i = 1; i <= n; i++){
        cin >> a[i];
    }
    for(int i = 1; i <= len; i++){
        st[i] = n / len * (i - 1) + 1;
        ed[i] = (i != len ? n / len * i : n);
        for(int j = st[i]; j <= ed[i]; j++){
```

```
            bel[j] = i;
        }
    }

    for(int i = 1; i <= n; i++){
        int op, l, r, c;
        cin >> op >> l >> r >> c;
        if(op == 0) update(l, r, c);
        else cout << query(l, r, c) << '\n';
    }
    return 0;
}
```

块状链表

```
#include <bits/stdc++.h>
using namespace std;

typedef long long ll;
typedef pair<ll, ll> PLL;
const int N = 1e5 + 10, INF = 0x3f3f3f3f, MOD = 10007;

int n, len;
int a[N], st[N], ed[N], bel[N];
vector<int> blk[450];

void insert(int l, int r){
    int now = 1, pos = l;
    while(pos > blk[now].size()){
        pos -= blk[now].size(), now++;
    }
    blk[now].insert(blk[now].begin() + pos - 1, r);
}

int query(int x){
    int now = 1, pos = x;
    while(pos > blk[now].size()){
        pos -= blk[now].size(), now++;
    }
    return blk[now][pos - 1];
}

int main(){
    cin >> n;
    len = sqrt(n);
    for(int i = 1; i <= n; i++){
        cin >> a[i];
    }
    for(int i = 1; i <= len; i++){
        st[i] = n / len * (i - 1) + 1;
        ed[i] = (i != len ? n / len * i : n);
        for(int j = st[i]; j <= ed[i]; j++){
            bel[j] = i;
            blk[i].push_back(a[j]);
```

```
        }
    }

    for(int i = 1; i <= n; i++){
        int op, l, r, c;
        cin >> op >> l >> r >> c;
        if(op == 0) insert(l, r);
        else cout << query(r) << '\n';
    }
    return 0;
}
```