整体二分

2020-11-23 | 2021-08-11 | OI & ACM | 2
4.3k | 4 分钟

简介

就是把所有的修改和询问都拿进去二分

对于每次二分的值，我们都能判断每个询问的答案在这之上或者在这之下

大概就是这样 能整体二分的东西要满足下面的性质（来自 2013 XHR 的集训队论文

定义当前二分的答案为判定标准。每个修改，结合具体询问和判定标准，可以算出该修改对该询问的判定答案的贡献。询问的判定答案是各个修改的贡献的和，而每个询问都有一个要求的判定答案，根据要求的判定答案与实际的判定答案的关系，我们可以确定询问的真实答案在当前判定标准之上还是之下

1. 询问的答案满足可二分性
2. 修改对判定标准的贡献互相独立，修改之间互相不影响
3. 修改如果对判定标准有贡献，则贡献唯一确定的与判定标准无关的值
4. 贡献满足交换律，结合律，具有可加性
5. 题目允许离线

无修整体二分

对于无修的整体二分，比较简单的写法是将二分出来的答案在全局拿一个数据结构来维护

能够证明指针的移动是 $O(n \log n)$ 的，或者说每个点都被经过 $O(\log n)$ 次

以 Luogu P3834 【模板】可持久化线段树 2（主席树）(整体二分) 为例

```
1  #include <iostream>
```

```cpp
#include <cstdio>
#include <algorithm>
#include <vector>
#define maxn 200010
#define lowbit(i) ((i) & (-i))
using namespace std;

int n, m, a[maxn];

vector<int> v[maxn];

int b[maxn], cnt;
void init_hash() {
    for (int i = 1; i <= n; ++i) b[i] = a[i];
    sort(b + 1, b + n + 1); cnt = unique(b + 1, b + n + 1) - b - 1;
    for (int i = 1; i <= n; ++i) a[i] = lower_bound(b + 1, b + cnt + 1, a[i]) - b;
}

int Bit[maxn];
void add(int i, int v) { while (i <= n) Bit[i] += v, i += lowbit(i); }

int get_sum(int i) {
    int s = 0;
    while (i) s += Bit[i], i -= lowbit(i);
    return s;
}

struct Query {
    int l, r, k, id;
} Q[maxn], t1[maxn], t2[maxn]; int ans[maxn];

int l, r;
inline void add(int x) { for (auto u : v[x]) add(u, 1); }

inline void del(int x) { for (auto u : v[x]) add(u, -1); }

void update(int L, int R) {
    while (r < R) add(++r);
    while (l > L) add(--l);
    while (r > R) del(r--);
    while (l < L) del(l++);
}

void solve(int l, int r, int L, int R) {
    if (L > R) return ;
    if (l == r) {
```

```
48          for (int i = L; i <= R; ++i) ans[Q[i].id] = b[l];
49          return ;
50      } int m = l + r >> 1, c1 = 0, c2 = 0; update(l, m);
51      for (int i = L; i <= R; ++i) {
52          int l = Q[i].l, r = Q[i].r, k = Q[i].k, v;
53          v = get_sum(r) - get_sum(l - 1);
54          if (k <= v) t1[++c1] = Q[i];
55          else t2[++c2] = Q[i], t2[c2].k -= v;
56      }
57      for (int i = 1; i <= c1; ++i) Q[L + i - 1] = t1[i];
58      for (int i = 1; i <= c2; ++i) Q[L + c1 + i - 1] = t2[i];
59      solve(l, m, L, L + c1 - 1); solve(m + 1, r, L + c1, R);
60  }
61
62  int main() {
63      ios::sync_with_stdio(false);
64      cin.tie(nullptr); cout.tie(nullptr);
65
66      cin >> n >> m;
67      for (int i = 1; i <= n; ++i) cin >> a[i]; init_hash();
68      for (int i = 1; i <= n; ++i) v[a[i]].push_back(i);
69      for (int i = 1; i <= m; ++i) cin >> Q[i].l >> Q[i].r >> Q[i].k, Q[i].id = i;
70      add(l = r = 1); solve(1, cnt, 1, m);
71      for (int i = 1; i <= m; ++i) cout << ans[i] << "\n";
72      return 0;
73  }
74
```

例题

1. Luogu P3834 【模板】可持久化线段树 2（主席树）(整体二分)
2. Luogu P3527 [POI2011]MET-Meteors

带修整体二分

我们把修改带着一起二分，注意到要保证相对顺序不会变化

以 Luogu P2617 Dynamic Rankings(整体二分) 为例

```
1  #include <iostream>
2  #include <cstdio>
3  #include <algorithm>
```

```cpp
#include <vector>
#define maxn 200010
#define maxm 400010
#define lowbit(i) ((i) & (-i))
#define INF 1000000000
using namespace std;

int n, m, a[maxn];

struct Query {
    int opt, l, r, k, id; // -1 add 1 del 0 query
} Q[maxm], t1[maxm], t2[maxm]; int cnt;

int Bit[maxn];
void add(int i, int v) { while (i <= n) Bit[i] += v, i += lowbit(i); }

int get_sum(int i) {
    int s = 0;
    while (i) s += Bit[i], i -= lowbit(i);
    return s;
}

int ans[maxn], cans;
void solve(int l, int r, int L, int R) {
    if (L > R) return ;
    if (l == r) {
        for (int i = L; i <= R; ++i) ans[Q[i].id] = l;
        return ;
    } int m = l + r >> 1, c1 = 0, c2 = 0;
    for (int i = L; i <= R; ++i) {
        int opt = Q[i].opt, l = Q[i].l, r = Q[i].r, k = Q[i].k, v;
        if (opt == 0) {
            v = get_sum(r) - get_sum(l - 1);
            if (k <= v) t1[++c1] = Q[i];
            else t2[++c2] = Q[i], t2[c2].k -= v;
        }
        else {
            if (l <= m) t1[++c1] = Q[i], add(k, opt);
            else t2[++c2] = Q[i];
        }
    }
    for (int i = 1; i <= c1; ++i) add(t1[i].k, -t1[i].opt);
    for (int i = 1; i <= c1; ++i) Q[L + i - 1] = t1[i];
    for (int i = 1; i <= c2; ++i) Q[L + c1 + i - 1] = t2[i];
    solve(l, m, L, L + c1 - 1); solve(m + 1, r, L + c1, R);
}
```

```cpp


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

    cin >> n >> m; cnt = n;
    for (int i = 1; i <= n; ++i) cin >> a[i], Q[i] = { 1, a[i], 0, i, 0 };
    for (int i = 1; i <= m; ++i) {
        char s[3]; cin >> s + 1;
        if (s[1] == 'Q') {
            ++cnt; Q[cnt].opt = 0; Q[cnt].id = ++cans;
            cin >> Q[cnt].l >> Q[cnt].r >> Q[cnt].k;
        }
        else {
            int x, y; cin >> x >> y;
            Q[++cnt] = { -1, a[x], 0, x, 0 };
            Q[++cnt] = { 1, y, 0, x, 0 }; a[x] = y;
        }
    }
    solve(0, INF, 1, cnt);
    for (int i = 1; i <= cans; ++i) cout << ans[i] << "\n";
    return 0;
}

```

例题

1. Luogu P2617 Dynamic Rankings(整体二分)

2. 简要题意：给定一个长度为 n 的序列 a_i 和 m 次操作，操作分两种，每次操作给定三个整数 l, r, k，第一种操作是将给定 $[l, r]$ 的数字都置为 k，第二种是查询区间 $[l, r]$ 的第 k 小数

   $n, m \leq 10^5, 1 \leq a_i, k \leq 10^9$

   简要题解：用线段树维护连续段，然后套用整体二分板子

   Luogu U71972 鸽子的序列

Tech # 整体二分

字符串hash

Luogu P3834 【模板】可持久化线段树 2（主席树）(整体二分)

1.8m | 27:07

9089 | 17845