线段树入门

mip001

从入门到入土

2023 年 11 月 29 日

目录

目录

区间求和

区间求和

给定 n 个正整数组成的数列 $a_1, a_2, \cdots, a_n$ 和 m 个区间 $[l_i, r_i]$，分别求这 m 个区间的区间和。$n, m \leq 10^5$

区间求和

区间求和

给定 n 个正整数组成的数列 $a_1, a_2, \cdots, a_n$ 和 m 个区间 $[l_i, r_i]$，分别求这 m 个区间的区间和。$n, m \leq 10^5$

前缀和即可

区间加法

区间加法

给定 n 个正整数组成的序列 $a_1, a_2, \cdots, a_n$，m 次操作，每次给区间 $[l_i, r_i]$ 加上 x_i，输出最终的序列。$n, m \leq 10^5$

区间加法

区间加法

给定 n 个正整数组成的序列 $a_1, a_2, \cdots, a_n$，m 次操作，每次给区间 $[l_i, r_i]$ 加上 x_i，输出最终的序列。$n, m \leq 10^5$

差分即可

区间加 & 区间查询

P3372【模板】线段树 1

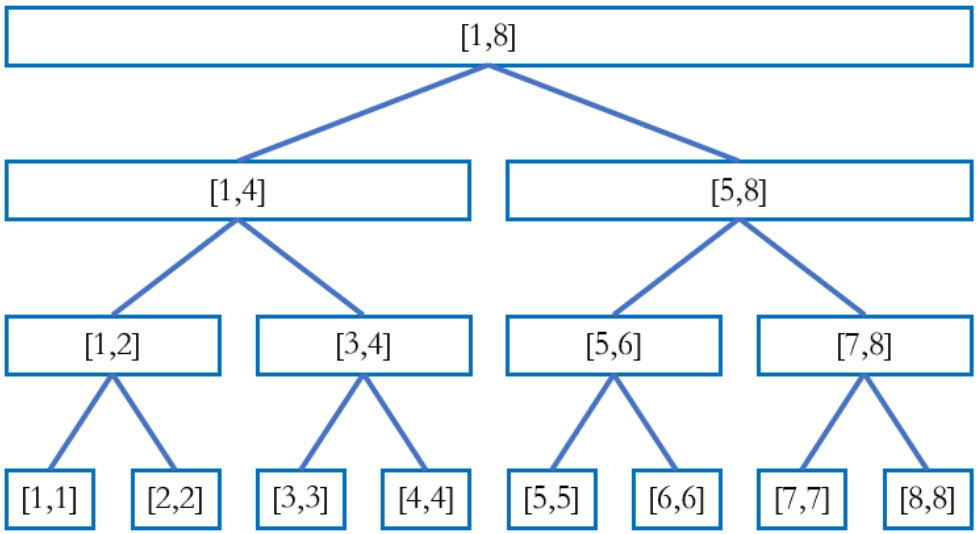
已知一个数列，你需要进行下面两种操作：

1. 将某区间每一个数加上 k。
2. 求出某区间每一个数的和。

数列长度 n，操作数 m。$n, m \leq 10^5$

简介

线段树被用来处理序列上的区间问题，它是一棵完全二叉树，它的每一个子节点都代表一个序列中的一段连续子区间，每个叶子节点都代表序列中的单个元素信息。对于一个节点，它的两个子节点会等分它所代表的区间。子节点会不断向其父亲节点传递信息，而父亲节点存储的是它每个子节点信息的整合。线段树只能维护满足结合律的信息，比如区间最值、区间求和、区间异或等。



目录

建树

```
1  int tree[maxn << 2], tag[maxn << 2]; //四倍空间!!!
2
3  void push_up(int u) //合并信息，写法取决于需要维护的信息
4  {
5      tree[u] = tree[u * 2] + tree[u * 2 + 1];
6  }
7
8  void build(int u, int l, int r) //递归建树，时间复杂度 O(n)
9  {
10     if (l == r)
11     {
12         tree[u] = a[l];
13         return;
14     }
15     int mid = (l + r) / 2;
16     build(u * 2, l, mid);
17     build(u * 2 + 1, mid + 1, r);
18     push_up(u);
19 }
```

单点修改

```
1  void modify(int u, int l, int r, int x, int val)
2  {
3      if (l == r)
4      {
5          tree[u] += val;
6          return;
7      }
8      int mid = (l + r) / 2;
9      if (x <= mid)
10         modify(u * 2, l, mid, x, val);
11     else
12         modify(u * 2 + 1, mid + 1, r, x, val);
13     push_up(u);
14 }
```

时间复杂度 $O(\log n)$

单点查询

```
1  int query(int u, int l, int r, int x)
2  {
3      if (l == r)
4          return tree[u];
5      int mid = (l + r) / 2;
6      if (x <= mid)
7          return query(u * 2, l, mid, x);
8      else
9          return query(u * 2 + 1, mid + 1, r, x);
10 }
```

时间复杂度 $O(\log n)$

Lazy Tag & push_down

Lazy Tag 会打在区间上，表示整个区间在过去的某个时刻曾被进行过某种操作。之后在穿过标记时，只需要将标记不断下放，直到区间长度为 1。

下放标记的过程就是 push_down。

```
1  void push_down(int u, int l, int r)
2  {
3      int mid = (l + r) / 2;
4      tree[u * 2] += tag[u] * (mid - l + 1);
5      tree[u * 2 + 1] += tag[u] * (r - mid);
6      tag[u * 2] += tag[u];
7      tag[u * 2 + 1] += tag[u];
8      tag[u] = 0;
9  }
```

区间修改

```
1  void modify(int u, int l, int r, int nl, int nr, int val)
2  {
3      if (l >= nl && r <= nr)
4      {
5          tree[u] += val * (r - l + 1);
6          tag[u] += val;
7          return;
8      }
9      int mid = (l + r) / 2;
10     push_down(u, l, r); //先下传标记
11     if (mid >= nl)
12         modify(u * 2, l, mid, nl, nr, val);
13     if (mid < nr)
14         modify(u * 2 + 1, mid + 1, r, nl, nr, val);
15     push_up(u);
16 }
```

时间复杂度 $O(\log n)$

区间查询

```
1  int query(int u, int l, int r, int nl, int nr)
2  {
3      if (l >= nl && r <= nr)
4          return tree[u];
5      int mid = (l + r) / 2;
6      push_down(u, l, r);
7      int res = 0;
8      if (mid >= nl)
9          res += query(u * 2, l, mid, nl, nr);
10     if (mid < nr)
11         res += query(u * 2 + 1, mid + 1, r, nl, nr);
12     return res;
13 }
```

时间复杂度 $O(\log n)$

完整代码

https://www.luogu.com.cn/paste/9x1cy3du

模板至少得打个五六遍

动态开点

```
1  void modify(int &u, int l, int r, int nl, int nr, int val)
2  {
3      if (!u)
4          u = ++cnt;
5      if (l >= nl && r <= nr)
6      {
7          tree[u] += val * (r - l + 1);
8          tag[u] += val;
9          return;
10     }
11     int mid = (l + r) / 2;
12     push_down(u, l, r);
13     if (mid >= nl)
14         modify(son[u][0], l, mid, nl, nr, val);
15     if (mid < nr)
16         modify(son[u][1], mid + 1, r, nl, nr, val);
17     push_up(u);
18 }
```

练习

1. P3373 【模板】线段树 2
2. P1531 I Hate It
3. P4513 小白逛公园
4. P1198 [JSOI2008] 最大数
5. P4588 [TJOI2018] 数学计算
6. P2471 [SCOI2007] 降雨量
7. P2574 XOR 的艺术
8. P2846 [USACO08NOV] Light Switching G
9. P3870 [TJOI2009] 开关
10. P2023 [AHOI2009] 维护序列
11. P1471 方差
12. CF527C Glass Carving
13. P7706 「Wdsr-2.7」文文的摄影布置

练习

14. P6105 [Ynoi2010] y-fast trie
15. P6327 区间加区间 sin 和
16. P5278 算术天才 □ 与等差数列
17. P3792 由乃与大母神原型和偶像崇拜
18. P4198 楼房重建
19. CF438D The Child and Sequence
20. P1972 [SDOI2009] HH 的项链
21. P5069 [Ynoi2015] 纵使日薄西山

目录

权值线段树

P3369 【模板】普通平衡树

您需要写一种数据结构（可参考题目标题），来维护一些数，其中需要提供以下操作：

1. 插入 x 数
2. 删除 x 数 (若有多个相同的数，应只删除一个)
3. 查询 x 数的排名 (排名定义为比当前数小的数的个数 $+1$)
4. 查询排名为 x 的数
5. 求 x 的前驱 (前驱定义为小于 x，且最大的数)
6. 求 x 的后继 (后继定义为大于 x，且最小的数)

对于 100% 的数据，$1 \leq n \leq 10^5$，$|x| \leq 10^7$

权值线段树

P3369 【模板】普通平衡树

您需要写一种数据结构（可参考题目标题），来维护一些数，其中需要提供以下操作：

1. 插入 x 数
2. 删除 x 数 (若有多个相同的数，应只删除一个)
3. 查询 x 数的排名 (排名定义为比当前数小的数的个数 $+1$)
4. 查询排名为 x 的数
5. 求 x 的前驱 (前驱定义为小于 x，且最大的数)
6. 求 x 的后继 (后继定义为大于 x，且最小的数)

对于 100% 的数据，$1 \leq n \leq 10^5$，$|x| \leq 10^7$

离线下来进行离散化，然后对值域建线段树。

权值线段树

查询排名为 x 的数：

```
int query2(int p, int l, int r, int rnk)
{
    if (l == r)
        return l;
    int mid = (l + r) / 2;
    if (tree[p * 2] >= rnk)
        return query2(p * 2, l, mid, rnk);
    else
        return query2(p * 2 + 1, mid + 1, r, rnk - tree[p * 2]);
}
```

完整代码：https://www.luogu.com.cn/paste/o2ajcehg

目录

例题

P5494 【模板】线段树分裂

给出一个可重集 a（编号为 1），它支持以下操作：

- `0 p x y`：将可重集 p 中大于等于 x 且小于等于 y 的值移动到一个新的可重集中（新可重集编号为从 2 开始的正整数，是上一次产生的新可重集的编号 +1）。
- `1 p t`：将可重集 t 中的数放入可重集 p，且清空可重集 t（数据保证在此后的操作中不会出现可重集 t）。
- `2 p x q`：在 p 这个可重集中加入 x 个数字 q。
- `3 p x y`：查询可重集 p 中大于等于 x 且小于等于 y 的值的个数。
- `4 p k`：查询在 p 这个可重集中第 k 小的数，不存在时输出 `-1`。

可重集中的数在 $1 \sim n$ 的范围内，有 m 个操作。$1 \leq n, m \leq 2 \times 10^5$

动态开点与垃圾回收

```
1  typedef long long ll;
2  ll tree[maxn << 5];
3  int son[maxn << 5][2];
4  int root[maxn << 2], tot; //存储每棵线段树的根节点
5  int rub[maxn], top; //垃圾回收
6  int cnt; //节点个数
7  int new_node()
8  {
9      if (top) //优先重复利用
10         return rub[top--];
11     else
12         return ++cnt;
13 }
14 void del(int &u)
15 {
16     tree[u] = son[u][0] = son[u][1] = 0;
17     rub[++top] = u;
18     u = 0;
19 }
```

分裂

```
1  void split(int &u, int &v, int l, int r, int nl, int nr)
2  {
3      if (!u)
4          return;
5      if (l >= nl && r <= nr)
6      {
7          v = u;
8          u = 0;
9          return;
10     }
11     if (!v)
12         v = new_node();
13     int mid = (l + r) / 2;
14     if (mid >= nl)
15         split(son[u][0], son[v][0], l, mid, nl, nr);
16     if (mid < nr)
17         split(son[u][1], son[v][1], mid + 1, r, nl, nr);
18     push_up(u), push_up(v);
19 }
```

合并

```
1  int merge(int &u, int &v, int l, int r)
2  {
3      if (!u || !v)
4          return u + v;
5      if (l == r)
6      {
7          tree[u] += tree[v];
8          del(v);
9          return u;
10     }
11     int mid = (l + r) / 2;
12     son[u][0] = merge(son[u][0], son[v][0], l, mid);
13     son[u][1] = merge(son[u][1], son[v][1], mid + 1, r);
14     push_up(u);
15     del(v);
16     return u;
17 }
```

完整代码：https://www.luogu.com.cn/paste/5di5sr80

习题

1. P2824 [HEOI2016/TJOI2016] 排序
2. CF911G Mass Change Queries
3. P4556 [Vani 有约会] 雨天的尾巴 /【模板】线段树合并
4. P3224 [HNOI2012] 永无乡

题单：https://www.luogu.com.cn/training/1124