以ZJOI2013K大数为例

（只针对本题）

外层是一颗权值线段树

内层是传统的区间线段树

插入：在a-b插入权值c

在外层线段树所有包含权值c的线段中，在其里层的线段树插入线段a-b

询问：在a-b中询问第k大的权值

从外层线段树（权值）开始，如果右孩子的内层线段树在区间a-b中的计数超过k个，显然答案在右孩子，否则在左孩子，以此类推

#include<bits/stdc++.h>

using namespace std;

#define lchd (u << 1)

#define rchd (u << 1 | 1)

#define MID ((L + R) >> 1)

struct SEG

{

int ROOT;

} P[200100];

struct seg

{

int L, R, sum, lazy;

} p[20000100];

int INDEX = 0, n, m;

void Read(int &x)

{

char c = getchar();

while (c > '9' || c < '0') c = getchar();

x = 0;

while (c <= '9' && c >= '0')

{

x = x \* 10 + c - '0';

c = getchar();

}

}

void init(int &u)

{

u = ++INDEX;

p[u].L = p[u].R = p[u].lazy = p[u].sum = 0;

}

void insert(int u, int L, int R, int l, int r)

{

p[u].sum += r - l + 1;

if (L == l && R == r)

{

++p[u].lazy;

return;

}

if (r <= MID)

{

if (!p[u].L)

init(p[u].L);

insert(p[u].L, L, MID, l, r);

}

else if (l > MID)

{

if (!p[u].R)

init(p[u].R);

insert(p[u].R, MID + 1, R, l, r);

}

else

{

if (!p[u].L)

init(p[u].L);

insert(p[u].L, L, MID, l, MID);

if (!p[u].R)

init(p[u].R);

insert(p[u].R, MID + 1, R, MID + 1, r);

}

}

void INSERT(int u, int L, int R, int l, int r, int x)

{

if (!P[u].ROOT)

init(P[u].ROOT);

insert(P[u].ROOT, 1, n, l, r);

if (L == R)

return;

if (x <= MID)

INSERT(lchd, L, MID, l, r, x);

else

INSERT(rchd, MID + 1, R, l, r, x);

}

int query(int u, int L, int R, int l, int r)

{

if (L == l && R == r)

return p[u].sum;

int num = (r - l + 1) \* p[u].lazy;

if (r <= MID)

num += query(p[u].L, L, MID, l, r);

else if (l > MID)

num += query(p[u].R, MID + 1, R, l, r);

else

num += query(p[u].L, L, MID, l, MID) + query(p[u].R, MID + 1, R, MID + 1, r);

return num;

}

int QUERY(int u, int L, int R, int l, int r, int x)

{

if (L == R) return L;

int num = query(P[rchd].ROOT, 1, n, l, r);

if (num >= x)

return QUERY(rchd, MID + 1, R, l, r, x);

else

return QUERY(lchd, L, MID, l, r, x - num);

}

struct XXX

{

int type, a, b, c;

} opt[50001];

map <int, int> H;

int HH[50001];

int main()

{

memset(P, 0, sizeof (P));

Read(n);

Read(m);

for (int i = 1; i <= m; i++)

{

Read(opt[i].type);

Read(opt[i].a);

Read(opt[i].b);

Read(opt[i].c);

H[opt[i].c] = 0;

}

int num = 0;

for (map <int, int>::iterator it = H.begin(); it != H.end(); ++it)

{

it->second = ++num;

HH[num] = it->first;

}

for (int i = 1; i <= m; i++)

{

if (opt[i].type == 1)

INSERT(1, 1, n, opt[i].a, opt[i].b, H[opt[i].c]);

else

printf("%d\n", HH[QUERY(1, 1, n, opt[i].a, opt[i].b, opt[i].c)]);

}

return 0;

}