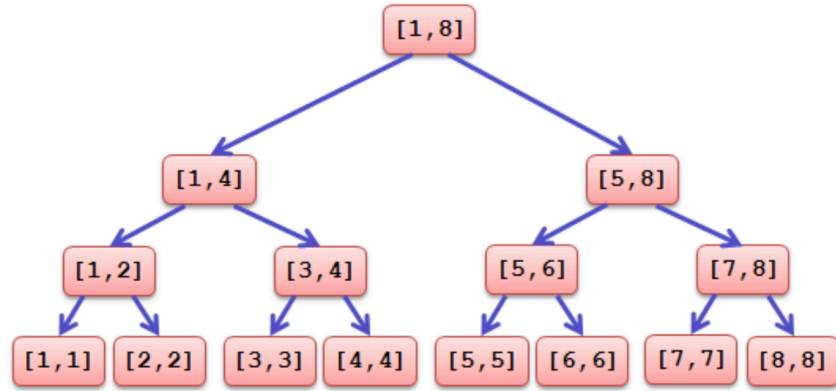


线段树

Segment Tree

Idea: 递归地二分区间使之成为树的结构。



Complexity: 单次操作 $O(\lg n)$

Code (双标记) :

```
1  #define lid id<<1
2  #define rid id<<1|1
3  #define len(id) (tr[id].r - tr[id].l + 1)
4  #define mid ((tr[id].l + tr[id].r) >> 1)
5  struct segTree{
6      int l, r;
7      LL sum, add, mul;
8  }tr[N<<2];
9  inline void pushup(int id){
10     tr[id].sum = tr[lid].sum + tr[rid].sum;
11 }
12 inline void pushdown(int id){
13     if(tr[id].mul != 1 && tr[id].l != tr[id].r){
14         (tr[lid].mul *= tr[id].mul) %= p;
15         (tr[lid].add += tr[id].mul) %= p;
16         (tr[lid].sum += tr[id].mul) %= p;
17         (tr[rid].mul *= tr[id].mul) %= p;
18         (tr[rid].add += tr[id].mul) %= p;
19         (tr[rid].sum += tr[id].mul) %= p;
20         tr[id].mul = 1;
21     }
22     if(tr[id].add != 0 && tr[id].l != tr[id].r){
23         (tr[lid].add += tr[id].add) %= p;
24         (tr[lid].sum += len(lid) * tr[id].add % p) %= p;
25         (tr[rid].add += tr[id].add) %= p;
26         (tr[rid].sum += len(rid) * tr[id].add % p) %= p;
27         tr[id].add = 0;
28     }
29 }
30 void build(int id, int l, int r){
31     tr[id].l = l; tr[id].r = r;
32     tr[id].sum = tr[id].add = 0;
33     tr[id].mul = 1;
34     if(tr[id].l == tr[id].r){
35         tr[id].sum = c[l] % p;
```

```

36         return;
37     }
38     build(lid, l, mid);
39     build(rid, mid+1, r);
40     pushup(id);
41 }
42 void add(int id, int l, int r, LL v){
43     pushdown(id);
44     if(tr[id].l == l && tr[id].r == r){
45         (tr[id].add += v) %= p;
46         (tr[id].sum += len(id) * v % p) %= p;
47         return;
48     }
49     if(r <= mid)
50         add(lid, l, r, v);
51     else if(l > mid)
52         add(rid, l, r, v);
53     else{
54         add(lid, l, mid, v);
55         add(rid, mid+1, r, v);
56     }
57     pushup(id);
58 }
59 void mul(int id, int l, int r, LL v){
60     pushdown(id);
61     if(tr[id].l == l && tr[id].r == r){
62         (tr[id].mul *= v) %= p;
63         (tr[id].add *= v) %= p;
64         (tr[id].sum *= v) %= p;
65         return;
66     }
67     if(r <= mid)
68         mul(lid, l, r, v);
69     else if(l > mid)
70         mul(rid, l, r, v);
71     else{
72         mul(lid, l, mid, v);
73         mul(rid, mid+1, r, v);
74     }
75     pushup(id);
76 }
77 LL querySum(int id, int l, int r){
78     pushdown(id);
79     if(tr[id].l == l && tr[id].r == r)
80         return tr[id].sum % p;
81     if(r <= mid)
82         return querySum(lid, l, r) % p;
83     else if(l > mid)
84         return querySum(rid, l, r) % p;
85     else
86         return (querySum(lid, l, mid) + querySum(rid, mid+1, r)) % p;
87 }

```