Standard Code Library

Your TeamName

Your School

May 11, 2024

Contents

一切的开始

宏定义

- 需要 C++11

```cpp
#include <bits/stdc++.h>
using namespace std;
using LL = long long;
#define FOR(i, x, y) for (decay<decltype(y)>::type i = (x), _##i = (y); i < _##i; ++i)
#define FORD(i, x, y) for (decay<decltype(x)>::type i = (x), _##i = (y); i > _##i; --i)
#ifdef zerol
#define dbg(x...) do { cout << "\033[32;1m" << #x << " -> "; err(x); } while (0)
void err() { cout << "\033[39;0m" << endl; }
template<template<typename...> class T, typename t, typename... A>
void err(T<t> a, A... x) { for (auto v: a) cout << v << ' '; err(x...); }
template<typename T, typename... A>
void err(T a, A... x) { cout << a << ' '; err(x...); }
#else
#define dbg(...)
#endif
// ------------------------------------------------------------------------
```

数据结构

ST 表

```cpp
struct ST{
    int n;
    std::vector<array<int,21>> st;
    ST(int n):n(n),st(n + 1) {}
    void init(vector<int>& a){
        for(int i = 1;i <= n;i ++)st[i][0] = a[i - 1];
        for(int j = 1;j <= 18;j ++){
            for(int i = 1;i + (1 << j) <= n + 1;i ++){
                st[i][j] = max(st[i][j - 1],st[i + (1 << (j - 1))][j - 1]);
            }
        }
    }
    int rmq(int l,int r){
        int j = log(r - l + 1)/log(2);
        return max(st[l][j],st[r - (1 << j) + 1][j]);
    }
};
```

线段树

```cpp
struct SegTree {
    int l, r;
    SegTree *ls, *rs;
    ll sum;
    ll plus;
    SegTree (const int L, const int R) : l(L), r(R) {
        plus = 0;
        if (L == R) {
            /*Initial*/
            ls = rs = nullptr;
        } else {
            int M = (L + R) >> 1;
            ls = new SegTree (L, M);
            rs = new SegTree (M + 1, R);
            pushup();
        }
    }
    void pushup() {
        sum = ls->sum + rs->sum;
        //  std::cerr << "AAA" << l << ' ' << r << ' ' << sum;
    }
    void make_tag(long long w) {
        sum += (r - l + 1) * w;
```

```
24          plus += w;
25      }
26      void pushdown() {
27          if (plus == 0) return;
28          ls->make_tag(plus);
29          rs->make_tag(plus);
30          plus = 0;
31      }
32      void upd(const int L, const int R, const int w) {
33          if ((L > r) || (l > R)) return;
34          if ((L <= l) && (r <= R)) {
35              make_tag(w);
36          } else {
37              pushdown();
38              ls->upd(L, R, w);
39              rs->upd(L, R, w);
40              pushup();
41          }
42      }
43  };
```

树状数组

```
1   template <typename T>
2   struct Fenwick {
3       int n;
4       std::vector<T> a;
5       Fenwick(int n) : n(n), a(n) {}
6       void add(int x, T v) {
7           for (int i = x + 1; i <= n; i += i & -i) {
8               a[i - 1] += v;
9           }
10      }
11      T sum(int x) {
12          T ans = 0;
13          for (int i = x; i > 0; i -= i & -i) {
14              ans += a[i - 1];
15          }
16          return ans;
17      }
18      T rangeSum(int l, int r) {
19          return sum(r) - sum(l);
20      }
21      int kth(T k) {
22          int x = 0;
23          // 先从高位开始取，如果当前这一位可以取，那么就考虑下一位是取 1 还是 0
24          // 到最后找到的就是最大的那个 pos 并且对应的 <=x 的
25          for (int i = 1 << std::__lg(n); i; i /= 2) {
26              if (x + i <= n && k >= a[x + i - 1]) {
27                  x += i;
28                  k -= a[x - 1];
29              }
30          }
31          return x;
32      }//树状数组上倍增本质上是通过倍增来快速找出对应的区间
33  };
```

DSU

```
1   struct DSU {
2       std::vector<int> f, siz;
3       DSU(int n) : f(n), siz(n, 1) { std::iota(f.begin(), f.end(), 0); }
4       int leader(int x) {
5           while (x != f[x]) x = f[x] = f[f[x]];
6           return x;
7       }
8       bool same(int x, int y) { return leader(x) == leader(y); }
9       bool merge(int x, int y) {
10          x = leader(x);
11          y = leader(y);
```

```
12          if (x == y) return false;
13          siz[x] += siz[y];
14          f[y] = x;
15          return true;
16      }
17      int size(int x) { return siz[leader(x)]; }
18  };
```

Splay

```
1   struct Node {
2     int v, sz, sm;
3     Node *ch[2], *fa;
4
5     Node(const int V, Node *const f) : v(V), sz(1), sm(1), fa(f) {
6       ch[0] = ch[1] = nullptr;
7     }
8
9     inline int GetRela(const int x) { return (v == x) ? -1 : (x > v); }
10
11    void pushup() { sm = (ch[0] ? ch[0]->sm : 0) + (ch[1] ? ch[1]->sm : 0) + sz; }
12
13    inline void rotate(const int x) {
14      auto nrt = ch[x];
15      ch[x] = nrt->ch[x ^ 1];
16      nrt->ch[x ^ 1] = this;
17      if (ch[x]) ch[x]->fa = this;
18      nrt->fa = fa; fa = nrt;
19      if (nrt->fa) nrt->fa->ch[nrt->fa->GetRela(nrt->v)] = nrt;
20      pushup(); nrt->pushup();
21    }
22
23    void splay(const Node *p) {
24      while (fa != p) {
25        auto pa = fa->fa;
26        if (pa == p) {
27          fa->rotate(fa->GetRela(v));
28        } else {
29          int k1 = fa->GetRela(v), k2 = pa->GetRela(fa->v);
30          if (k1 == k2) {
31            pa->rotate(k1);
32            fa->rotate(k1);
33          } else {
34            fa->rotate(k1);
35            fa->rotate(k2);
36          }
37        }
38      }
39    }
40  };
```

LCT

```
1   struct Node {
2     int v, s;
3     bool tag;
4     Node *ch[2], *fa;
5
6     inline void maketag() {
7       tag = !tag;
8       std::swap(ch[0], ch[1]);
9     }
10    inline void pushup() {
11      s = v;
12      for (auto u : ch) if (u != nullptr) {
13        s ^= u->s;
14      }
15    }
16    inline void pushdown() {
17      if (tag) {
```

4

```cpp
        for (auto u : ch) if (u != nullptr) {
          u->maketag();
        }
        tag = false;
      }
    }

    inline int Getson() { return fa->ch[1] == this; }

    inline bool IsRoot() { return (fa == nullptr) || (fa->ch[Getson()] != this); }

    void rotate(const int x) {
      auto nt = ch[x];
      ch[x] = nt->ch[x ^ 1];
      nt->ch[x ^ 1] = this;
      if (ch[x]) ch[x]->fa = this;
      nt->fa = fa;
      if (!IsRoot()) { fa->ch[Getson()] = nt; }
      fa = nt;
      pushup(); nt->pushup();
    }

    void splay() {
      static Node* stk[maxn];
      int top = 0;
      stk[++top] = this;
      for (auto u = this; !u->IsRoot(); stk[++top] = u = u->fa);
      while (top) stk[top--]->pushdown();
      while (!IsRoot()) {
        if (fa->IsRoot()) {
          fa->rotate(Getson());
        } else {
          auto pa = fa->fa;
          int l1 = Getson(), l2 = fa->Getson();
          if (l1 == l2) {
            pa->rotate(l2);
            fa->rotate(l1);
          } else {
            fa->rotate(l1);
            fa->rotate(l2);
          }
        }
      }
    }
};
Node *node[maxn], Mem[maxn];

void Cut(const int x, const int y);
void Link(const int x, const int y);
void Query(const int x, const int y);
void Update(const int x, const int y);

void access(Node *u) {
  for (Node *v = nullptr; u; u = (v = u)->fa) {
    u->splay();
    u->ch[1] = v; u->pushup();
  }
}

void makeroot(Node *const u) {
  access(u);
  u->splay();
  u->maketag();
}

void Query(const int x, const int y) {
  auto u = node[x], v = node[y];
  makeroot(u);
  access(v);
  v->splay();
  qw(v->s, '\n');
```

```
89     }
90
91     void Link(const int x, const int y) {
92       auto u = node[x], v = node[y];
93       makeroot(u);
94       access(v); v->splay();
95       if (u->IsRoot() == false) return;
96       u->fa = v;
97     }
98
99     void Cut(const int x, const int y) {
100      auto u = node[x], v = node[y];
101      makeroot(u); access(v); u->splay();
102      if ((u->ch[1] != v) || (v->ch[0] != nullptr)) return;
103      u->ch[1] = v->fa = nullptr;
104      u->pushup();
105    }
106
107    // w[x] -> y
108    void Update(const int x, const int y) {
109      auto u = node[x];
110      u->splay();
111      u->s ^= u->v;
112      u->s ^= (u->v = a[x] = y);
113    }
```

扫描线

```
1     //二维数点
2     struct Segment{
3         int l,r,h,add;
4         bool operator <(const Segment a)const{
5             return h < a.h;
6         }
7     };
8     struct SegTree {
9         int l, r;
10        SegTree *ls, *rs;
11        int mn,len;
12        int plus;
13        SegTree (const int L, const int R) : l(L), r(R) {
14            plus = 0;len = 0;
15            if (L == R) {
16                ls = rs = nullptr;
17            } else {
18                int M = (L + R) >> 1;
19                ls = new SegTree (L, M);
20                rs = new SegTree (M + 1, R);
21                pushup();
22            }
23        }
24        void pushup() {
25            if(plus) len = r - l + 1;
26            else if(l == r)len = 0;
27            else len = ls->len + rs->len;
28        }
29        void make_tag(int w) {
30            plus += w;
31        }
32        void pushdown() {
33            if (plus == 0) return;
34            ls->make_tag(plus);
35            rs->make_tag(plus);
36            plus = 0;
37        }
38        void update(const int L, const int R, const int w) {
39            if ((L > r) || (l > R)) {
40                return;
41            }
42            if ((L <= l) && (r <= R)) {
43                make_tag(w);
```

```
44              pushup();
45              return ;
46          } else {
47              ls->update(L, R, w);
48              rs->update(L, R, w);
49              pushup();
50          }
51      }
52  };
53  //矩形面积并
54  #include<bits/stdc++.h>
55
56  using namespace std;
57  typedef long long ll;
58  const double eps = 1e-8;
59  const int maxn = 2e5 + 7;
60  std::vector<int> x;
61  struct Segment{
62      int l,r,h,add;
63      bool operator <(const Segment a)const{
64          return h < a.h;
65      }
66  };
67  struct SegTree {
68      int l, r;
69      SegTree *ls, *rs;
70      int mn,len;
71      int plus;
72      SegTree (const int L, const int R) : l(L), r(R) {
73          plus = 0;len = 0;
74          if (L == R) {
75              ls = rs = nullptr;
76          } else {
77              int M = (L + R) >> 1;
78              ls = new SegTree (L, M);
79              rs = new SegTree (M + 1, R);
80              pushup();
81          }
82      }
83      void pushup() {
84          if(plus) len = x[r] - x[l - 1];
85          else if(l == r)len = 0;
86          else len = ls->len + rs->len;
87      }
88      void make_tag(int w) {
89          plus += w;
90      }
91      void pushdown() {
92          if (plus == 0) return;
93          ls->make_tag(plus);
94          rs->make_tag(plus);
95          plus = 0;
96      }
97      void update(const int L, const int R, const int w) {
98          if ((L >= x[r]) || (x[l - 1] >= R)) {
99              return;
100         }
101         if ((L <= x[l - 1]) && (x[r] <= R)) {
102             make_tag(w);
103             pushup();
104             return ;
105         } else {
106             //pushdown();
107             ls->update(L, R, w);
108             rs->update(L, R, w);
109             pushup();
110         }
111     }
112 };
113 int main(){
114     ios::sync_with_stdio(false);
```

```
115        cin.tie(0);
116
117        vector<Segment> s;
118        int n;
119        cin >> n;
120        for(int i = 0;i < n;i ++){
121            int xa,ya,xb,yb;
122            cin >> xa >> ya >> xb >> yb;
123            x.push_back(xa);
124            x.push_back(xb);
125            s.push_back({xa,xb,ya,1});
126            s.push_back({xa,xb,yb,-1});
127        }
128        sort(s.begin(),s.end());
129        sort(x.begin(),x.end());
130        x.erase(unique(x.begin(),x.end()),x.end());
131        int N = x.size();
132        SegTree Seg(1,N - 1);
133        ll ans = 0;
134        if(s.size()){
135            Seg.update(s[0].l,s[0].r,s[0].add);
136            for(int i = 1;i < s.size();i ++){
137                ans += 1ll * Seg.len * (s[i].h - s[i - 1].h);
138                Seg.update(s[i].l,s[i].r,s[i].add);
139            }
140        }
141        cout << ans << "\n";
142        return 0;
143    }
```

Seg beats

本质上是维护了两棵线段树，A 树维护区间内最大值产生的贡献，B 树维护剩下树的贡献。注意 A 树某节点的孩子不一定全部能贡献到该节点，因为孩子的最大值不一定是父亲的最大值。所以要注意下传标记时，A 树的孩子下传的可能是 B 的标记。

beats 的部分是，每次让序列里每个数对另一个数 V 取 min，则直接暴力递归到 inRange 且 B 的最大值小于 V 的那些节点上，转化成对 A 那个节点的区间加法（加上 $V - val_A$）即可。这么做的均摊复杂度是 $O(\log n)$。

做区间历史最大值的方法是，维护两个标记 x, y，x 是真正的加标记，y 是 x 在上次下传结束并清零后的历史最大值。下传时注意先下传 y 再下传 x。实现历史最值是平凡的，不需要 beats。beats 解决的仅是取 min 的操作。

下面五个操作分别是：区间加，区间对 k 取 min，区间求和，区间最大值，区间历史最大值。

```cpp
1   #include <array>
2   #include <iostream>
3   #include <algorithm>
4
5   typedef long long int ll;
6
7   const int maxn = 500005;
8
9   ll a[maxn];
10
11  const ll inf = 0x3f3f3f3f3f3f3f3fll;
12
13  struct Node {
14    Node *ls, *rs;
15    int l, r, maxCnt;
16    ll v, add, maxAdd, sum, maxV, maxHistory;
17
18    Node(const int L, const int R) :
19        ls(nullptr), rs(nullptr), l(L), r(R), maxCnt(0),
20        v(0), add(0), maxAdd(0), sum(0), maxV(-inf), maxHistory(-inf) {}
21
22    inline bool inRange(const int L, const int R) {
23      return L <= l && r <= R;
24    }
25    inline bool outRange(const int L, const int R) {
26      return l > R || L > r;
27    }
```

```
28
29    void addVal(const ll t, int len) {
30      add += t;
31      sum += len * t;
32      maxV += t;
33    }
34
35    void makeAdd(const ll t, int len) {
36      addVal(t, len);
37      maxHistory = std::max(maxHistory, maxV);
38      maxAdd = std::max(maxAdd, add);
39    }
40  };
41
42  void pushup(Node *x, Node *y) {
43    y->maxV = std::max(y->ls->maxV, y->rs->maxV);
44    y->sum = y->ls->sum + y->rs->sum;
45    y->maxHistory = std::max({y->maxHistory, y->ls->maxHistory, y->rs->maxHistory});
46    if (x->ls->maxV != x->rs->maxV) {
47      bool flag = x->ls->maxV < x->rs->maxV;
48      if (flag) std::swap(x->ls, x->rs);
49      x->maxV = x->ls->maxV;
50      x->maxCnt = x->ls->maxCnt;
51      y->maxV = std::max(y->maxV, x->rs->maxV);
52      y->sum += x->rs->sum;
53      x->sum = x->ls->sum;
54      if (flag) std::swap(x->ls, x->rs);
55    } else {
56      x->maxCnt = x->ls->maxCnt + x->rs->maxCnt;
57      x->sum = x->ls->sum + x->rs->sum;
58      x->maxV = x->ls->maxV;
59    }
60    x->maxHistory = std::max({x->ls->maxHistory, x->rs->maxHistory, x->maxHistory, y->maxHistory});
61  }
62
63  void New(Node *&u1, Node *&u2, int L, int R) {
64    u1 = new Node(L, R);
65    u2 = new Node(L, R);
66    if (L == R) {
67      u1->v = u1->sum = u1->maxV = u1->maxHistory = a[L];
68      u1->maxCnt = 1;
69    } else {
70      int M = (L + R) >> 1;
71      New(u1->ls, u2->ls, L, M);
72      New(u1->rs, u2->rs, M + 1, R);
73      pushup(u1, u2);
74    }
75  }
76
77  void pushdown(Node *x, Node *y) {
78    ll val = std::max(x->ls->maxV, x->rs->maxV);
79    std::array<Node*, 2> aim({y, x});
80    Node *curl = aim[x->ls->maxV == val], *curr = aim[x->rs->maxV == val];
81    x->ls->maxAdd = std::max(x->ls->maxAdd, x->ls->add + curl->maxAdd);
82    x->ls->maxHistory = std::max(x->ls->maxHistory, x->ls->maxV + curl->maxAdd);
83    x->ls->addVal(curl->add, x->ls->maxCnt);
84    x->rs->maxAdd = std::max(x->rs->maxAdd, x->rs->add + curr->maxAdd);
85    x->rs->maxHistory = std::max(x->rs->maxHistory, x->rs->maxV + curr->maxAdd);
86    x->rs->addVal(curr->add, x->rs->maxCnt);
87    y->ls->maxAdd = std::max(y->ls->maxAdd, y->ls->add + y->maxAdd);
88    y->rs->maxAdd = std::max(y->rs->maxAdd, y->rs->add + y->maxAdd);
89    y->ls->addVal(y->add, x->ls->r - x->ls->l + 1 - x->ls->maxCnt);
90    y->rs->addVal(y->add, x->rs->r - x->rs->l + 1 - x->rs->maxCnt);
91    x->add = y->add = x->maxAdd = y->maxAdd = 0;
92  }
93
94  void addV(Node *x, Node *y, int L, int R, ll k) {
95    if (x->inRange(L, R)) {
96      x->makeAdd(k, x->maxCnt);
97      y->makeAdd(k, x->r - x->l + 1 - x->maxCnt);
98    } else if (!x->outRange(L, R)) {
```

```
 99        pushdown(x, y);
100        addV(x->ls, y->ls, L, R, k);
101        addV(x->rs, y->rs, L, R, k);
102        pushup(x, y);
103      }
104  }
105
106  std::array<ll, 3> qry(Node *x, Node *y, const int L, const int R) {
107      if (x->inRange(L, R)) return {x->sum + y->sum * ((x->r - x->l + 1) != x->maxCnt), x->maxV, x->maxHistory};
108      else if (x->outRange(L, R)) return {0, -inf, -inf};
109      else {
110        pushdown(x, y);
111        auto A = qry(x->ls, y->ls, L, R), B = qry(x->rs, y->rs, L, R);
112        return {A[0] + B[0], std::max(A[1], B[1]), std::max(A[2], B[2])};
113      }
114  }
115
116  void minV(Node *x, Node *y, const int L, const int R, int k) {
117      if (x->maxV <= k) return;
118      if (x->inRange(L, R) && y->maxV < k) {
119        ll delta = k - x->maxV;
120        x->makeAdd(delta, x->maxCnt);
121      } else if (!x->outRange(L, R)) {
122        pushdown(x, y);
123        minV(x->ls, y->ls, L, R, k);
124        minV(x->rs, y->rs, L, R, k);
125        pushup(x, y);
126      }
127  }
128
129  int main() {
130      std::ios::sync_with_stdio(false);
131      std::cin.tie(nullptr);
132      int n, m;
133      std::cin >> n >> m;
134      for (int i = 1; i <= n; ++i) std::cin >> a[i];
135      Node *rot1, *rot2;
136      New(rot1, rot2, 1, n);
137      for (int op, l, r; m; --m) {
138        std::cin >> op >> l >> r;
139        if (op == 1) {
140          std::cin >> op;
141          addV(rot1, rot2, l, r, op);
142        } else if (op == 2) {
143          std::cin >> op;
144          minV(rot1, rot2, l, r, op);
145        } else {
146          std::cout << qry(rot1, rot2, l, r)[op - 3] << '\n';
147        }
148      }
149  }
```

珂朵莉树

```
 1  auto getPos(int pos) {
 2    return --s.upper_bound({pos + 1, 0, 0});
 3  }
 4
 5  void split(int pos) {
 6    auto it = getPos(pos);
 7    auto [l, r, v] = *it;
 8    s.erase(it);
 9    if (pos > l) s.insert({l, pos - 1, v});
10    s.insert({pos, r, v});
11  }
12
13  void add(int l, int r, int v) {
14    split(l); split(r + 1);
15    for (auto x = getPos(l), y = getPos(r + 1); x != y; ++x) {
16      x->v += v;
17    }
```

```
18    }
19
20    void upd(int l, int r, int v) {
21      split(l); split(r + 1);
22      s.erase(getPos(l), getPos(r + 1));
23      s.insert({l, r, v});
24    }
```

getPos(pos)：找到 pos 所在的迭代器
split(pos)：把 pos 所在的迭代器区间 [l, r] 分成 [l, pos - 1] 和 [pos, r] 两个

李超树

插入线段 $kx + b$ 求某点最值

```
1     constexpr long long INF = 1'000'000'000'000'000'000;
2     constexpr int C = 100'000;
3     struct Line {
4         int k;
5         long long b;
6         Line(int k, long long b) : k(k), b(b) {}
7     };
8     long long f(const Line &line, int x) {
9         return 1LL * line.k * x + line.b;
10    }
11    struct Node {
12        Node *lc, *rc;
13        Line line;
14        Node(const Line &line) : lc(nullptr), rc(nullptr), line(line) {}
15    };
16    void modify(Node *&p, int l, int r, Line line) {
17        if (p == nullptr) {
18            p = new Node(line);
19            return;
20        }
21        int m = (l + r) / 2;
22        bool le = f(p -> line, l) < f(line, l);
23        bool mi = f(p -> line, m) < f(line, m);
24        if (!mi)
25            std::swap(p -> line, line);
26        if (r - l == 1)
27            return;
28        if (le != mi) {
29            modify(p -> lc, l, m, line);
30        } else {
31            modify(p -> rc, m, r, line);
32        }
33    }
34    Node *merge(Node *p, Node *q, int l, int r) {
35        if (p == nullptr)
36            return q;
37        if (q == nullptr)
38            return p;
39        int m = (l + r) / 2;
40        p -> lc = merge(p -> lc, q -> lc, l, m);
41        p -> rc = merge(p -> rc, q -> rc, m, r);
42        modify(p, l, r, q -> line);
43        return p;
44    }
45    long long query(Node *p, int l, int r, int x) {
46        if (p == nullptr)
47            return INF;
48        long long ans = f(p -> line, x);
49        if (r - l == 1)
50            return ans;
51        int m = (l + r) / 2;
52        if (x < m) {
53            return std::min(ans, query(p -> lc, l, m, x));
54        } else {
55            return std::min(ans, query(p -> rc, m, r, x));
56        }
```

```
57   }
```

动态维护凸壳

```
1    /**
2     * Author: Simon Lindholm
3     * Date: 2017-04-20
4     * License: CC0
5     * Source: own work
6     * Description: Container where you can add lines of the form kx+m, and query maximum values at points x.
7     *  Useful for dynamic programming.
8     * Time: O(\log N)
9     * Status: tested
10    */
11
12   struct Line {
13     mutable ll k, m, p;
14     bool operator<(const Line &o) const { return k < o.k; }
15     bool operator<(ll x) const { return p < x; }
16   };
17
18   struct LineContainer: multiset<Line, less<>> {
19     const ll inf = LLONG_MAX;
20     ll val_offset = 0;
21     void offset(ll x) {
22       val_offset += x;//整体加
23     }
24     ll div(ll a, ll b) {
25       return a / b - ((a^b) < 0 && a%b);
26     }
27     bool isect(iterator x, iterator y) {
28       if (y == end()) {
29         x->p = inf;
30         return 0;
31       }
32       if (x->k == y->k) {
33         x->p = (x->m > y->m)? inf: -inf;
34       } else {
35         x->p = div(y->m - x->m, x->k - y->k);
36       }
37       return x->p >= y->p;
38     }
39     void add(ll k, ll m) {
40       auto z = insert({k, m - val_offset, 0}), y = z++, x = y;//这里加减看情况
41       while (isect(y, z)) z = erase(z);
42       if (x != begin() && isect(--x, y)) isect(x, y = erase(y));
43       while ((y = x) != begin() && (--x)->p >= y->p) isect(x, erase(y));
44     }
45     ll query(ll x) {
46       assert(!empty());
47       auto l = *lower_bound(x);
48       return l.k * x + l.m + val_offset;
49     }
50   };
51
52   LineContainer* merge(LineContainer *S, LineContainer *T) {
53     if (S->size() > T->size())
54       swap(S, T);
55     for (auto l: *S) {
56       T->add(l.k, l.m + S->val_offset);
57     }
58     return T;
59   }
```

TODO

线段树合并和分裂

数学

类欧几里得

- $m = \lfloor \frac{an+b}{c} \rfloor$.
- $f(a,b,c,n) = \sum_{i=0}^{n} \lfloor \frac{ai+b}{c} \rfloor$: 当 $a \geq c$ or $b \geq c$ 时，$f(a,b,c,n) = (\frac{a}{c})n(n+1)/2 + (\frac{b}{c})(n+1) + f(a \bmod c, b \bmod c, c, n)$；否则 $f(a,b,c,n) = nm - f(c, c-b-1, a, m-1)$。
- $g(a,b,c,n) = \sum_{i=0}^{n} i \lfloor \frac{ai+b}{c} \rfloor$: 当 $a \geq c$ or $b \geq c$ 时，$g(a,b,c,n) = (\frac{a}{c})n(n+1)(2n+1)/6 + (\frac{b}{c})n(n+1)/2 + g(a \bmod c, b \bmod c, c, n)$；否则 $g(a,b,c,n) = \frac{1}{2}(n(n+1)m - f(c, c-b-1, a, m-1) - h(c, c-b-1, a, m-1))$。
- $h(a,b,c,n) = \sum_{i=0}^{n} \lfloor \frac{ai+b}{c} \rfloor^2$: 当 $a \geq c$ or $b \geq c$ 时，$h(a,b,c,n) = (\frac{a}{c})^2 n(n+1)(2n+1)/6 + (\frac{b}{c})^2(n+1) + (\frac{a}{c})(\frac{b}{c})n(n+1) + h(a \bmod c, b \bmod c, c, n) + 2(\frac{a}{c})g(a \bmod c, b \bmod c, c, n) + 2(\frac{b}{c})f(a \bmod c, b \bmod c, c, n)$；否则 $h(a,b,c,n) = nm(m+1) - 2g(c, c-b-1, a, m-1) - 2f(c, c-b-1, a, m-1) - f(a,b,c,n)$。

图论

LCA

- 倍增

```cpp
void dfs(int u, int fa) {
    pa[u][0] = fa; dep[u] = dep[fa] + 1;
    FOR (i, 1, SP) pa[u][i] = pa[pa[u][i - 1]][i - 1];
    for (int& v: G[u]) {
        if (v == fa) continue;
        dfs(v, u);
    }
}

int lca(int u, int v) {
    if (dep[u] < dep[v]) swap(u, v);
    int t = dep[u] - dep[v];
    FOR (i, 0, SP) if (t & (1 << i)) u = pa[u][i];
    FORD (i, SP - 1, -1) {
        int uu = pa[u][i], vv = pa[v][i];
        if (uu != vv) { u = uu; v = vv; }
    }
    return u == v ? u : pa[u][0];
}
```

计算几何

二维几何：点与向量

```cpp
#define y1 yy1
#define nxt(i) ((i + 1) % s.size())
typedef double LD;
const LD PI = 3.14159265358979323846;
const LD eps = 1E-10;
int sgn(LD x) { return fabs(x) < eps ? 0 : (x > 0 ? 1 : -1); }
struct L;
struct P;
typedef P V;
struct P {
    LD x, y;
    explicit P(LD x = 0, LD y = 0): x(x), y(y) {}
    explicit P(const L& l);
};
struct L {
    P s, t;
    L() {}
    L(P s, P t): s(s), t(t) {}
};

P operator + (const P& a, const P& b) { return P(a.x + b.x, a.y + b.y); }
P operator - (const P& a, const P& b) { return P(a.x - b.x, a.y - b.y); }
```

```
23  P operator * (const P& a, LD k) { return P(a.x * k, a.y * k); }
24  P operator / (const P& a, LD k) { return P(a.x / k, a.y / k); }
25  inline bool operator < (const P& a, const P& b) {
26      return sgn(a.x - b.x) < 0 || (sgn(a.x - b.x) == 0 && sgn(a.y - b.y) < 0);
27  }
28  bool operator == (const P& a, const P& b) { return !sgn(a.x - b.x) && !sgn(a.y - b.y); }
29  P::P(const L& l) { *this = l.t - l.s; }
30  ostream &operator << (ostream &os, const P &p) {
31      return (os << "(" << p.x << "," << p.y << ")");
32  }
33  istream &operator >> (istream &is, P &p) {
34      return (is >> p.x >> p.y);
35  }
36
37  LD dist(const P& p) { return sqrt(p.x * p.x + p.y * p.y); }
38  LD dot(const V& a, const V& b) { return a.x * b.x + a.y * b.y; }
39  LD det(const V& a, const V& b) { return a.x * b.y - a.y * b.x; }
40  LD cross(const P& s, const P& t, const P& o = P()) { return det(s - o, t - o); }
41  // ----------------------------------------
```

字符串

后缀自动机

杂项

STL
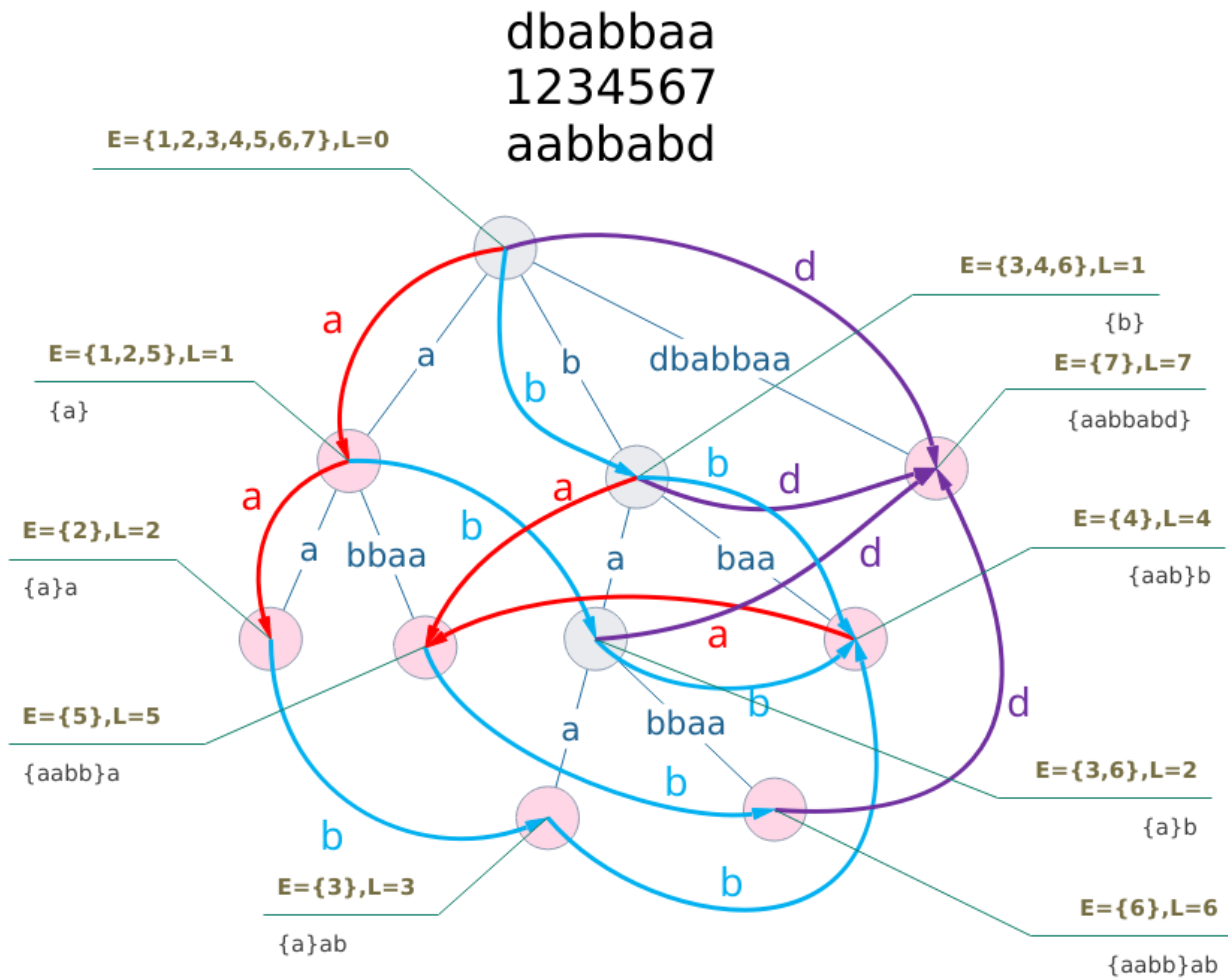
- copy

```
template <class InputIterator, class OutputIterator>
  OutputIterator copy (InputIterator first, InputIterator last, OutputIterator result);
```