数据结构

哈希表

```
struct Hash {
    int operator()(const Node &b) const {
        return b;
    }
};
template <typename Key, typename Value, typename Hash>
struct HashMap {
    Hash hash;
    struct D {
        int nxt;
        Key key;
        Value value;
        D() { nxt = 0; }
        D(int nxt, Key key): nxt(nxt), key(key) {
            value = Value();
        }
    } e[400105];
    int head[50105], vis[50105], tot, timet;
    HashMap() { tot = 0, timet = 1, memset(vis, 0, sizeof vis); }
    int _hash(Key d) { return (hash(d) % 50003 + 50003) % 50003 + 1; }
    int find(Key d) {
        int h = _hash(d);
        if (vis[h] != timet) return 0;
        for (int x = head[h]; x; x = e[x].nxt) {
            if (e[x].key == d) return x;
        }
        return 0;
    }
    int end() { return 0; }
    Value& operator[](Key d) {
        int h = _hash(d);
        if (vis[h] == timet) {
            for (int x = head[h]; x; x = e[x].nxt) {
                if (e[x].key == d) return e[x].value;
            }
        } else {
            vis[h] = timet;
            head[h] = 0;
        }
        vis[h] = timet;
        e[++tot] = D(head[h], d);
        head[h] = tot;
        return e[tot].value;
    }
    void clear() {
        tot = 0;
        timet++;
    }
};
```

ST 表

```
int f[2000005][20], log[2000005]={0,0,1};
int main() {
    int n=read(), m=read();
    for(int i=1;i<=n;i++) f[i][0]=read();
    for(int i=3;i<=n;i++) log[i]=log[i/2]+1;
    for(int j=1;j<=20;j++)
        for(int i=1;i+(1<<j)-1<=n;i++)
            f[i][j]=max(f[i][j-1], f[i+(1<<(j-1))][j-1]);
    for(int i=1;i<=m;i++) {
        int x=read(), y=read();
        int s=log[y-x+1];
        printf("%d\n", max(f[x][s], f[y-(1<<s)+1][s]));
    }
    return 0;
}
```

树状数组

单点加

```
#define l(x) ((x)&(-x))
int tree[500005], n, m, a, b, c;
int add(int i, int v) { for(;i<=n;i+=l(i))tree[i]+=v; }
int ask(int i, int ans=0) { for(;i>=1;i-=l(i))ans+=tree[i]; return ans; }
int main() {
    scanf("%d%d", &n, &m);
    for(int i=1;i<=n;i++) scanf("%d", &a), add(i, a);
    for(int i=1;i<=m;i++){
        scanf("%d%d%d", &a, &b, &c);
        if(a==1) add(b,c);
        else printf("%d\n", ask(c)-ask(b-1));
    }
}
```

区间加

```
#define l(x) ((x)&(-x))
#define ll long long
ll tree[500005], n;
int add(int i, ll v) { for(;i<=n;i+=l(i))tree[i]+=v; }
int ask(int i, ll ans=0) { for(;i>=1;i-=l(i))ans+=tree[i]; return ans; }
int main() {
        int m, l=0, a, b, c, d;
    scanf("%lld%d", &n, &m);
    for(int i=1;i<=n;i++) scanf("%d", &a), add(i, a-l), l=a;
    for(int i=1;i<=m;i++){
        scanf("%d%d", &a, &b);
        if(a==1) scanf("%d%d", &c, &d), add(b,d), add(c+1,-d);
        else printf("%d\n", ask(b));
    }
}
```

权值线段树

1. 插入 数
2. 删除 数（若有多个相同的数，只删除一个）；
3. 查询 数的排名（若有多个相同的数，输出最小的排名）；
4. 查询排名为 的数；
5. 求 的前驱（前驱定义为小于 ，且最大的数）；
6. 求 的后继（后继定义为大于 ，且最小的数）。

```cpp
int cnt, root, lch[maxn], rch[maxn], sizeh[maxn];
void add(int *r, int L, int R, int p, int v) {
        !*r ?  *r = ++cnt : 0;
        sizeh[*r] += v;
        return L==R?void():(p <= mid) ? add(&lch[*r], L, mid, p, v) : add(&rch[*r], mid
}
int rankh(int r, int L, int R, int p) {
        return sizeh[r] == 0 || p > R  ?sizeh[r]: (p <= mid) ? rankh(lch[r], L, mid, p)
}
int kth(int r, int L, int R, int p) {
        return L == R ? L :(sizeh[lch[r]] >= p) ? kth(lch[r], L, mid, p) : kth(rch[r], 
}
int main() {
        const int n = read();
        for(int i = 1; i <= n; i++) {
                int op = read(), num = read();
                if (op == 1) add(&root, -1e7, 1e7, num, 1);
                else if (op == 2) add(&root, -1e7, 1e7, num, -1);
                else if (op == 3) print(rankh(root, -1e7, 1e7, num) + 1);
                else if (op == 4) print(kth(root, -1e7, 1e7, num));
                else if (op == 5) print(kth(root, -1e7, 1e7, rankh(root, -1e7, 1e7, num
                else if (op == 6) print(kth(root, -1e7, 1e7, rankh(root, -1e7, 1e7, num
        }
}
```

可持久化线段树

```
const int maxn = 1e5;  // 数据范围
int tot, n, m;
int sum[(maxn << 5) + 10], rt[maxn + 10], ls[(maxn << 5) + 10], rs[(maxn << 5) + 10];
int a[maxn + 10], ind[maxn + 10], len;
int getid(const int &val) {  // 离散化
  return lower_bound(ind + 1, ind + len + 1, val) - ind;
}
int build(int l, int r) {  // 建树
  int root = ++tot;
  if (l == r) return root;
  int mid = l + r >> 1;
  ls[root] = build(l, mid);
  rs[root] = build(mid + 1, r);
  return root;  // 返回该子树的根节点
}
int update(int k, int l, int r, int root) {  // 插入操作
  int dir = ++tot;
  ls[dir] = ls[root], rs[dir] = rs[root], sum[dir] = sum[root] + 1;
  if (l == r) return dir;
  int mid = l + r >> 1;
  if (k <= mid)
    ls[dir] = update(k, l, mid, ls[dir]);
  else
    rs[dir] = update(k, mid + 1, r, rs[dir]);
  return dir;
}
int query(int u, int v, int l, int r, int k) {  // 查询操作
  int mid = l + r >> 1,
      x = sum[ls[v]] - sum[ls[u]];  // 通过区间减法得到左儿子中所存储的数值个数
  if (l == r) return l;
  if (k <= x)  // 若 k 小于等于 x ，则说明第 k 小的数字存储在在左儿子中
    return query(ls[u], ls[v], l, mid, k);
  else  // 否则说明在右儿子中
    return query(rs[u], rs[v], mid + 1, r, k - x);
}
int main() {
  scanf("%d%d", &n, &m);
  for (int i = 1; i <= n; ++i) scanf("%d", a + i);
  memcpy(ind, a, sizeof ind);
  sort(ind + 1, ind + n + 1);
  len = unique(ind + 1, ind + n + 1) - ind - 1;
  rt[0] = build(1, len);
  for (int i = 1; i <= n; ++i) rt[i] = update(getid(a[i]), 1, len, rt[i - 1]);
  while (m--) {
    scanf("%d%d%d", &l, &r, &k);
    printf("%d\n", ind[query(rt[l - 1], rt[r], 1, len, k)]);  // 回答询问
  }
  }
```

K-D Tree

给定 n 个点，找出最近 2 点距离。

```
const int maxn = 200010;
int n, d[maxn], lc[maxn], rc[maxn];
double ans = 2e18;
struct node {
  double x, y;
} s[maxn];
double L[maxn], R[maxn], D[maxn], U[maxn];
double dist(int a, int b) {
  return (s[a].x - s[b].x) * (s[a].x - s[b].x) +
         (s[a].y - s[b].y) * (s[a].y - s[b].y);
}
bool cmp1(node a, node b) { return a.x < b.x; }
bool cmp2(node a, node b) { return a.y < b.y; }
void maintain(int x) {
  L[x] = R[x] = s[x].x;
  D[x] = U[x] = s[x].y;
  if (lc[x])
    L[x] = min(L[x], L[lc[x]]), R[x] = max(R[x], R[lc[x]]),
    D[x] = min(D[x], D[lc[x]]), U[x] = max(U[x], U[lc[x]]);
  if (rc[x])
    L[x] = min(L[x], L[rc[x]]), R[x] = max(R[x], R[rc[x]]),
    D[x] = min(D[x], D[rc[x]]), U[x] = max(U[x], U[rc[x]]);
}
int build(int l, int r) {
  if (l > r) return 0;
  if (l == r) {
    maintain(l);
    return l;
  }
  int mid = (l + r) >> 1;
  double avx = 0, avy = 0, vax = 0, vay = 0;  // average variance
  for (int i = l; i <= r; i++) avx += s[i].x, avy += s[i].y;
  avx /= (double)(r - l + 1);
  avy /= (double)(r - l + 1);
  for (int i = l; i <= r; i++)
    vax += (s[i].x - avx) * (s[i].x - avx),
        vay += (s[i].y - avy) * (s[i].y - avy);
  if (vax >= vay)
    d[mid] = 1, nth_element(s + l, s + mid, s + r + 1, cmp1);
  else
    d[mid] = 2, nth_element(s + l, s + mid, s + r + 1, cmp2);
  lc[mid] = build(l, mid - 1), rc[mid] = build(mid + 1, r);
  maintain(mid);
  return mid;
}
double f(int a, int b) {
  double ret = 0;
  if (L[b] > s[a].x) ret += (L[b] - s[a].x) * (L[b] - s[a].x);
  if (R[b] < s[a].x) ret += (s[a].x - R[b]) * (s[a].x - R[b]);
  if (D[b] > s[a].y) ret += (D[b] - s[a].y) * (D[b] - s[a].y);
  if (U[b] < s[a].y) ret += (s[a].y - U[b]) * (s[a].y - U[b]);
  return ret;
}
void query(int l, int r, int x) {
  if (l > r) return;
  int mid = (l + r) >> 1;
  if (mid != x) ans = min(ans, dist(x, mid));
  if (l == r) return;
  double distl = f(x, lc[mid]), distr = f(x, rc[mid]);
  if (distl < ans && distr < ans) {
    if (distl < distr) {
      query(l, mid - 1, x);
      if (distr < ans) query(mid + 1, r, x);
    } else {
      query(mid + 1, r, x);
      if (distl < ans) query(l, mid - 1, x);
    }
  } else {
    if (distl < ans) query(l, mid - 1, x);
    if (distr < ans) query(mid + 1, r, x);
  }
}
int main() {
  scanf("%d", &n);
  for (int i = 1; i <= n; i++) scanf("%lf%lf", &s[i].x, &s[i].y);
  build(1, n);
  for (int i = 1; i <= n; i++) query(1, n, i);
  printf("%.4lf\n", sqrt(ans));
  return 0;
}
```

Link Cut tree

一棵　个点的树，每个点的初始权值为　。

对于这棵树有　个操作，每个操作为以下四种操作之一：

- `+ u v c`：将　到　的路径上的点的权值都加上自然数　；
- `- u1 v1 u2 v2`：将树中原有的边　　　　删除，加入一条新边　　　　，保证操作完之后仍然是一棵树；
- `* u v c`：将　到　的路径上的点的权值都乘上自然数　；
- `/ u v`：询问　到　的路径上的点的权值和，将答案对　　　　取模。

第一行两个整数

```
const int maxn = 100010;
const int mod = 51061;
int n, q, u, v, c;
char op;
struct Splay {
  int ch[maxn][2], fa[maxn], siz[maxn], val[maxn], sum[maxn], rev[maxn],
      add[maxn], mul[maxn];

  void clear(int x) {
    ch[x][0] = ch[x][1] = fa[x] = siz[x] = val[x] = sum[x] = rev[x] = add[x] =
        0;
    mul[x] = 1;
  }

  int getch(int x) { return (ch[fa[x]][1] == x); }

  int isroot(int x) {
    clear(0);
    return ch[fa[x]][0] != x && ch[fa[x]][1] != x;
  }

  void maintain(int x) {
    clear(0);
    siz[x] = (siz[ch[x][0]] + 1 + siz[ch[x][1]]) % mod;
    sum[x] = (sum[ch[x][0]] + val[x] + sum[ch[x][1]]) % mod;
  }

  void pushdown(int x) {
    clear(0);
    if (mul[x] != 1) {
      if (ch[x][0])
        mul[ch[x][0]] = (mul[x] * mul[ch[x][0]]) % mod,
        val[ch[x][0]] = (val[ch[x][0]] * mul[x]) % mod,
        sum[ch[x][0]] = (sum[ch[x][0]] * mul[x]) % mod,
        add[ch[x][0]] = (add[ch[x][0]] * mul[x]) % mod;
      if (ch[x][1])
        mul[ch[x][1]] = (mul[x] * mul[ch[x][1]]) % mod,
        val[ch[x][1]] = (val[ch[x][1]] * mul[x]) % mod,
        sum[ch[x][1]] = (sum[ch[x][1]] * mul[x]) % mod,
        add[ch[x][1]] = (add[ch[x][1]] * mul[x]) % mod;
      mul[x] = 1;
    }
    if (add[x]) {
      if (ch[x][0])
        add[ch[x][0]] = (add[ch[x][0]] + add[x]) % mod,
        val[ch[x][0]] = (val[ch[x][0]] + add[x]) % mod,
        sum[ch[x][0]] = (sum[ch[x][0]] + add[x] * siz[ch[x][0]] % mod) % mod;
      if (ch[x][1])
        add[ch[x][1]] = (add[ch[x][1]] + add[x]) % mod,
        val[ch[x][1]] = (val[ch[x][1]] + add[x]) % mod,
```

```cpp
      sum[ch[x][1]] = (sum[ch[x][1]] + add[x] * siz[ch[x][1]] % mod) % mod;
    add[x] = 0;
  }
  if (rev[x]) {
    if (ch[x][0]) rev[ch[x][0]] ^= 1, swap(ch[ch[x][0]][0], ch[ch[x][0]][1]);
    if (ch[x][1]) rev[ch[x][1]] ^= 1, swap(ch[ch[x][1]][0], ch[ch[x][1]][1]);
    rev[x] = 0;
  }
}

void update(int x) {
  if (!isroot(x)) update(fa[x]);
  pushdown(x);
}

void print(int x) {
  if (!x) return;
  pushdown(x);
  print(ch[x][0]);
  printf("%lld ", x);
  print(ch[x][1]);
}

void rotate(int x) {
  int y = fa[x], z = fa[y], chx = getch(x), chy = getch(y);
  fa[x] = z;
  if (!isroot(y)) ch[z][chy] = x;
  ch[y][chx] = ch[x][chx ^ 1];
  fa[ch[x][chx ^ 1]] = y;
  ch[x][chx ^ 1] = y;
  fa[y] = x;
  maintain(y);
  maintain(x);
  maintain(z);
}

void splay(int x) {
  update(x);
  for (int f = fa[x]; f = fa[x], !isroot(x); rotate(x))
    if (!isroot(f)) rotate(getch(x) == getch(f) ? f : x);
}

void access(int x) {
  for (int f = 0; x; f = x, x = fa[x]) splay(x), ch[x][1] = f, maintain(x);
}

void makeroot(int x) {
  access(x);
  splay(x);
  swap(ch[x][0], ch[x][1]);
  rev[x] ^= 1;
  }

  int find(int x) {
    access(x);
    splay(x);
    while (ch[x][0]) x = ch[x][0];
    splay(x);
    return x;
  }
} st;

int main() {
  scanf("%lld%lld", &n, &q);
  for (int i = 1; i <= n; i++) st.val[i] = 1, st.maintain(i);
  for (int i = 1; i < n; i++) {
    scanf("%lld%lld", &u, &v);
    if (st.find(u) != st.find(v)) st.makeroot(u), st.fa[u] = v;
  }
  while (q--) {
    scanf(" %c%lld%lld", &op, &u, &v);
    if (op == '+') {
      scanf("%lld", &c);
      st.makeroot(u), st.access(v), st.splay(v);
      st.val[v] = (st.val[v] + c) % mod;
      st.sum[v] = (st.sum[v] + st.siz[v] * c % mod) % mod;
      st.add[v] = (st.add[v] + c) % mod;
    }
    if (op == '-') {
      st.makeroot(u);
      st.access(v);
      st.splay(v);
      if (st.ch[v][0] == u && !st.ch[u][1]) st.ch[v][0] = st.fa[u] = 0;
      scanf("%lld%lld", &u, &v);
      if (st.find(u) != st.find(v)) st.makeroot(u), st.fa[u] = v;
    }
    if (op == '*') {
      scanf("%lld", &c);
      st.makeroot(u), st.access(v), st.splay(v);
      st.val[v] = st.val[v] * c % mod;
      st.sum[v] = st.sum[v] * c % mod;
      st.mul[v] = st.mul[v] * c % mod;
    }
    if (op == '/')
      st.makeroot(u), st.access(v), st.splay(v), printf("%lld\n", st.sum[v]);
  }
  return 0;
}
```