

# Standard Cool Library

Patricky

Southwestern Petroleum University

September 2022

# Contents

<b>初</b>	<b>2</b>
快读 . . . . .	2
对拍 . . . . .	2
<b>数据结构</b>	<b>3</b>
线段树 . . . . .	3
lazy 线段树 . . . . .	3
二维数点 . . . . .	4
<b>图</b>	<b>5</b>
DSU on tree . . . . .	5
Dinic . . . . .	5
mcmf . . . . .	6
2SAT . . . . .	7
<b>智力题</b>	<b>8</b>
找出数组中仅出现一次的数。 . . . .	8
子集和 . . . . .	8
子段和 . . . . .	8
子段最值和 . . . . .	8
子段 gcd 和 . . . . .	8
最少修改多少数使得原数组非严格递增 . . . . .	9
最少修改多少数使得原数组严格递增 . . . . .	9
最大子段积 . . . . .	9
<b>一些题</b>	<b>9</b>
对区间的异或依然是区间吗 . . . . .	9
多次询问某区间内所有区间的异或和。 . . . .	11
<b>数学</b>	<b>12</b>
Z . . . . .	12
Poly . . . . .	12
<b>杂项</b>	<b>14</b>
随机数 . . . . .	15
浮点数比较 . . . . .	15
二维偏序排序规则 . . . . .	15
G . . . . .	15

## 初

```
1  #!/bin/zsh
2  g++ $1 -std=c++23 -fsanitize=undefined -g -Wall -Wextra -Wfatal-errors -Os -DLOCAL -o ${1%%.*} && time ./${1%%.*}

1  BasedOnStyle: LLVM
2  UseTab: Never
3  IndentWidth: 4
4  DerivePointerAlignment: false
5  PointerAlignment: true
6  AlwaysBreakAfterReturnType: None
7  AlwaysBreakTemplateDeclarations: true
8  AlwaysBreakBeforeMultilineStrings: true
9  AlignOperands: true
10 AlignAfterOpenBracket: true
11 AlignConsecutiveBitFields: true
12 AlignConsecutiveMacros: true
13 ConstructorInitializerAllOnOneLineOrOnePerLine: true
14 AllowAllConstructorInitializersOnNextLine: false
15 BinPackArguments: false
16 BinPackParameters: false
17 IncludeBlocks: Regroup
```

## 快读

```
1  inline char nc() {
2      static char buf[100000], *p1 = buf, *p2 = buf;
3      return p1 == p2 && (p2 = (p1 = buf) + fread(buf, 1, 100000, stdin), p1 == p2) ? EOF : *p1++;
4  }
5
6  template <typename T>
7  bool read(T& v) {
8      static char ch;
9      while (ch != EOF && !isdigit(ch)) ch = nc();
10     if (ch == EOF) return false;
11     for (v = 0; isdigit(ch); ch = nc()) v = v * 10 + ch - '0';
12     return true;
13 }
14
15 template <typename T>
16 void write(T p) {
17     static int stk[70], tp;
18     if (p == 0) { putchar('0'); return; }
19     if (p < 0) { p = -p; putchar('-'); }
20     while (p) stk[++tp] = p % 10, p /= 10;
21     while (tp) putchar(stk[tp--] + '0');
22 }
```

## 对拍

```
1  #!/bin/zsh
2  g++ -o r main.cpp -O2 -std=c++17
3  g++ -o std std.cpp -O2 -std=c++17
4  while true; do
5      python gen.py > in
6      ./std < in > stdout
7      ./r < in > out
8      if test $? -ne 0; then
9          exit 0
10     fi
11     if diff stdout out; then
12         printf "AC\n"
13     else
14         printf "GG\n"
15         exit 0
16     fi
17 done
```

# 数据结构

## 线段树

```
1  template <class S, S (*op)(S, S), S (*e)()>
2  class SegTree {
3      const int n, siz;
4      vector<int> node;
5      vector<S> tr;
6      void pushup(int u) { tr[u] = op(tr[u << 1], tr[u << 1 | 1]); }
7      S get(int u, int l, int r, int ql, int qr) {
8          if (qr <= l or r <= ql) { return e(); }
9          if (ql <= l and r <= qr) { return tr[u]; }
10
11          int m = (l + r) >> 1;
12          return op( get(u << 1, l, m, ql, qr), get(u << 1 | 1, m, r, ql, qr));
13      }
14      template <class F>
15      int lower_bound(int u, int l, int r, F &&check) {
16          if (r - l == 1) { return l; }
17          int m = (l + r) >> 1;
18          if (check(tr[u << 1])) {
19              return lower_bound(u << 1, l, m, check);
20          } else {
21              return lower_bound(u << 1 | 1, m, r, check);
22          }
23      }
24      template <class F>
25      int upper_bound(int u, int l, int r, F &&check) {
26          if (r - l == 1) { return l; }
27          int m = (l + r) >> 1;
28          if (check(tr[u << 1 | 1])) {
29              return lower_bound(u << 1 | 1, m, r, check);
30          } else {
31              return lower_bound(u << 1, l, m, check);
32          }
33      }
34      public:
35      SegTree(int n) : SegTree(vector<S>(n, e())) { }
36      SegTree(const vector<S> &a): n(a.size()), siz(4 << __lg(n)), node(n), tr(siz) {
37          function<void(int, int, int)> build = [&](int u, int l, int r) {
38              if (r - l == 1) { node[l] = u; tr[u] = a[l]; return; }
39              int m = (l + r) >> 1; build(u << 1, l, m); build(u << 1 | 1, m, r); pushup(u);
40          };
41          build(1, 0, n);
42      }
43      void set(int p, const S &v) { assert(0 <= p < n); p = node[p]; tr[p] = v; while (p >= 1) { pushup(p); } }
44      S get(int p) { assert(0 <= p < n); return tr[node[p]]; }
45      S get(int l, int r) { assert(0 <= l <= r < n); return get(1, 0, n, l, r); }
46      int lower_bound(function<bool(S)> &&check) { return lower_bound(1, 0, n, check); }
47      int upper_bound(function<bool(S)> &&check) { return upper_bound(1, 0, n, check); }
48  };
```

## lazy 线段树

```
1  #define ls u << 1
2  #define rs ls | 1
3  template < class S, S (*op)(S, S), S (*e)(), class F, S (*mapping)(F, S), F (*merge)(F, F), F(*id)() >
4  // S 是元素类型
5  // op 是 S 和 S 的合并操作
6  // e 是 S 的单位元
7  // F 是懒标记
8  // mapping 是 F 对 S 的映射
9  // merge 是 F 和 F 的合并操作
10 // id 是 F 的单位元
11
12 /* 区间加 & 区间乘
13 * struct S { Z sum; Z len; };
14 * struct F { Z mul; Z add; };
15 * S op(S a, S b) { return { a.sum + b.sum, a.len + b.len }; }
```

```

16 * S e() { return { 0, 0 }; }
17 * S mapping(F f, S x) { return { x.sum * f.mul + f.add * x.len, x.len }; }
18 * F merge(F f, F g) { return { f.mul * g.mul, f.mul * g.add + f.add }; }
19 * F id() { return { 1, 0 }; }
20 */
21 class LazySegTree {
22     const int n, siz;
23     vector<int> node;
24     vector<S> tr;
25     vector<F> tag;
26     void pushup(int u) { tr[u] = op(tr[ls], tr[rs]); }
27     void pushtag(int u, F t) {
28         if (u >= siz) return;
29         tr[u] = mapping(t, tr[u]);
30         tag[u] = merge(t, tag[u]);
31     }
32     void pushdown(int u) { pushtag(ls, tag[u]); pushtag(rs, tag[u]); tag[u] = id(); }
33     S get(int u, int l, int r, int ql, int qr) {
34         if (qr <= l or r <= ql) return e();
35         if (ql <= l and r <= qr) return tr[u];
36         pushdown(u); int m = (l + r) >> 1;
37         return op( get(ls, l, m, ql, qr), get(rs, m, r, ql, qr));
38     }
39     template <class Func>
40     int lower_bound(int u, int l, int r, Func &&check) {
41         if (r - l == 1) { return l; }
42         pushdown(u); int m = (l + r) >> 1;
43         if (check(tr[ls])) {
44             return lower_bound(ls, l, m, check);
45         } else {
46             return lower_bound(rs, m, r, check);
47         }
48     }
49     template <class Func>
50     int upper_bound(int u, int l, int r, Func &&check) {
51         if (r - l == 1) { return l; }
52         pushdown(u); int m = (l + r) >> 1;
53         if (check(tr[rs])) {
54             return lower_bound(rs, m, r, check);
55         } else {
56             return lower_bound(ls, l, m, check);
57         }
58     }
59     void apply(int u, int l, int r, int ql, int qr, const F &v) {
60         if (qr <= l or r <= ql) return;
61         if (ql <= l and r <= qr) { pushtag(u, v); return; }
62         pushdown(u); int m = (l + r) >> 1; apply(ls, l, m, ql, qr, v); apply(rs, m, r, ql, qr, v); pushup(u);
63     }
64 public:
65     LazySegTree(int n) : LazySegTree(vector<S>(n, e())) { }
66     LazySegTree(const vector<S> &a): n(a.size()), siz(4 << __lg(n)), node(n), tr(siz), tag(siz, id()) {
67         function<void(int, int, int)> build = [&](int u, int l, int r) {
68             if (r - l == 1) { node[l] = u; tr[u] = a[l]; return; }
69             int m = (l + r) >> 1; build(ls, l, m); build(rs, m, r); pushup(u);
70         };
71         build(1, 0, n);
72     }
73     void set(int p, const S &v) { assert(0 <= p < n); p = node[p]; tr[p] = v; while (p >>= 1) { pushup(p); } }
74     S get(int p) { assert(0 <= p < n); return tr[node[p]]; }
75     S get(int l, int r) { assert(0 <= l <= r < n); return get(1, 0, n, l, r); }
76     void apply(int l, int r, const F &t) { assert(0 <= l <= r < n); apply(1, 0, n, l, r, t); }
77     int lower_bound(function<bool(S)> &&check) { return lower_bound(1, 0, n, check); }
78     int upper_bound(function<bool(S)> &&check) { return upper_bound(1, 0, n, check); }
79 };

```

## 二维数点

$$\sum_{i=0}^x \sum_{j=0}^y S_{i,j}$$

$$\sum_{i=0}^x \sum_{j=0}^y [(i, j) \prec (x, y)], \text{ where } i \leq x \wedge j \leq y$$

```

1 // x, y
2 array<int, 2> a[maxn];
3 // x, y, id, weight
4 array<int, 4> q[maxn << 2];
5 int n, m, t, ans[maxn];
6 int c[maxn * 5], tot;
7
8 int tr[maxn];
9 int qry(int i) { int ans{}; for (; i; i -= i & -i) { ans += tr[i]; } return ans; }
10 void mdf(int i, int v) { for (; i <= n; i += i & -i) { tr[i] += v; } }
11
12 int main() {
13     cin >> n >> m;
14     for (int i = 1; i <= n; ++i) { cin >> a[i][0] >> a[i][1]; a[i][0] += 1, a[i][1] += 1; c[++tot] = a[i][1]; }
15     for (int i = 1; i <= m; ++i) { array<int, 4> qi{}; for (int j = 0; j < 4; ++j) { cin >> qi[j]; qi[j] += 1; }
16         // 询问左上角的点记的是 y_1 - 1
17         c[++tot] = qi[1] - 1;
18         c[++tot] = qi[3];
19         q[++t] = {qi[0] - 1, qi[1] - 1, i, 1}; // (x1 - 1, y1 - 1)
20         q[++t] = {qi[0] - 1, qi[3], i, -1}; // (x1 - 1, y2)
21         q[++t] = {qi[2], qi[1] - 1, i, -1}; // (x2, y1 - 1)
22         q[++t] = {qi[2], qi[3], i, 1}; // (x2, y2)
23     }
24
25     sort(c + 1, c + 1 + tot); tot = unique(c + 1, c + 1 + tot) - (c + 1);
26     for (int i = 1; i <= n; ++i) { a[i][1] = lower_bound(c + 1, c + 1 + tot, a[i][1]) - c; }
27     for (int i = 1; i <= m; ++i) { q[i][1] = lower_bound(c + 1, c + 1 + tot, q[i][1]) - c; }
28     sort(a + 1, a + 1 + n);
29     sort(q + 1, q + 1 + t);
30
31     for (int i = 1, j = 1; i <= t; ++i) {
32         while (j <= n && a[j][0] <= q[i][0]) { mdf(a[j++][1], 1); }
33         ans[q[i][2]] += q[i][3] * qry(q[i][1]);
34     }
35     for (int i = 1; i <= m; ++i) { cout << ans[i] << "\n"; }
36     return 0;
37 }

```

图

## DSU on tree

```

1 void gsz(int u) { sz[u] = 1; for (int
    ↳ v: g[u]) { dep[v] = dep[u] + 1; xsum[v] ^= xsum[u]; gsz(v); sz[u] += sz[v]; if (sz[v] > sz[son[u]]) son[u] = v; } }
2 void cal(int u, int p) { rep(i, 0, 22) ckmax(ans[p], dep[u] + f[xsum[u] ^ bits[i]]); for (int v: g[u]) cal(v, p); }
3 void add(int u) { ckmax(f[xsum[u]], dep[u]); for (int v: g[u]) add(v); }
4 void del(int u) { f[xsum[u]] = -inf; for (int v: g[u]) del(v); }
5 void dsu(int u, int kp) {
6     for (int v: g[u]) if (v != son[u]) dsu(v, 0);
7     if (son[u]) dsu(son[u], 1);
8     ckmax(f[xsum[u]], dep[u]);
9     rep(i, 0, 22) ckmax(ans[u], dep[u] + f[xsum[u] ^ bits[i]]);
10    for (int v: g[u]) if (v != son[u]) cal(v, u), add(v);
11    if (!kp) del(u);
12 }

```

## Dinic

```

1 template<class T> struct Flow {
2     const int n;
3     struct Edge { int to; T cap; Edge(int _to, T _cap) : to(_to), cap(_cap) { } };
4     vector<Edge> e;
5     vector<vector<int>> g;
6     vector<int> cur, h;

```

```

7 Flow(int n) : n(n), g(n) {}
8
9 bool bfs(int s, int t) {
10     h.assign(n, -1);
11     h[s] = 0;
12     queue<int> q; q.push(s); while (q.size()) {
13         const int u = q.front(); q.pop();
14         for (int i : g[u]) {
15             auto [v, c] = e[i];
16             if (c > 0 and h[v] == -1) {
17                 h[v] = h[u] + 1;
18                 if (v == t) { return true; }
19                 q.push(v);
20             }
21         }
22     }
23     return { };
24 }
25
26 T dfs(int u, int t, T f) {
27     if (u == t) { return f; }
28     auto r = f;
29     for (int &i = cur[u]; i < static_cast<int>(g[u].size()); ++i) {
30         const int j = g[u][i];
31         auto [v, c] = e[j];
32         if (c > 0 and h[v] == h[u] + 1) {
33             auto a = dfs(v, t, min(r, c));
34             e[j].cap -= a; e[j ^ 1].cap += a; r -= a;
35             if (r == 0) { return f; }
36         }
37     }
38     return f - r;
39 }
40
41 void link(int u, int v, T c) {
42     g[u].push_back(e.size()); e.emplace_back(v, c);
43     g[v].push_back(e.size()); e.emplace_back(u, 0);
44 }
45
46 T run(int s, int t) {
47     T ans = 0;
48     while (bfs(s, t)) { cur.assign(n, 0); ans += dfs(s, t, ::numeric_limits<T>::max()); }
49     return ans;
50 }
51 };
52
53 static constexpr int inf = 1E9;

```

## mcmf

```

1 template <class Cap, class Cost>
2 class MCMF {
3     static constexpr Cap INF = numeric_limits<Cap>::max();
4     struct iedge { int v; Cap c; Cost f; iedge(int v, Cap c, Cost f) : v(v), c(c), f(f) {} };
5     const int n;
6     vector<iedge> e;
7     vector<vector<int>> g;
8     vector<Cost> h, dis;
9     vector<int> pre;
10    bool dijkstra(int s, int t) {
11        dis.assign(n, INF);
12        pre.assign(n, -1);
13        using T = pair<Cost, int>;
14        priority_queue<T, vector<T>, greater<T>> q; q.emplace(dist[s] = 0, s);
15        while (!q.empty()) {
16            Cost d = q.top().first;
17            int u = q.top().second; q.pop();
18            if (dis[u] != d) continue;
19            for (int i : g[u]) {
20                auto [v, c, f] = e[i];
21                if (c > 0 && dis[v] > d + h[u] - h[v] + f) {

```

```

22         dis[v] = d + h[u] - h[v] + f;
23         pre[v] = i;
24         q.emplace(dis[v], v);
25     }
26 }
27 }
28 return dis[t] != INF;
29 }
30 public:
31     struct Edge {
32         int u, v;
33         Cap flow;
34         Cost cost;
35         Edge(int u, int v, Cap flow, Cost cost) : u(u), v(v), flow(flow), cost(cost) { }
36     };
37     Flow(int n) : n(n), g(n) {}
38     void addEdge(int u, int v, Cap cap, Cost cost) {
39         g[u].push_back(e.size()); e.emplace_back(v, cap, cost);
40         g[v].push_back(e.size()); e.emplace_back(u, 0, - cost);
41     }
42     pair<Cap, Cost> flow(int s, int t) {
43         Cap flow = 0;
44         Cost cost = 0;
45         h.assign(n, 0);
46         while (dijkstra(s, t)) {
47             for (int i = 0; i < n; ++i) h[i] += dis[i];
48             Cap aug = INF;
49             for (int i = t; i != s; i = e[pre[i] ^ 1].v) aug = min(aug, e[pre[i]].c);
50             for (int i = t; i != s; i = e[pre[i] ^ 1].v) { e[pre[i]].c -= aug; e[pre[i] ^ 1].c += aug; }
51             flow += aug;
52             cost += Cost(aug) * h[t];
53         }
54         return {flow, cost};
55     }
56     vector<Edge> edges() const {
57         int m = e.size(); vector<Edge> res; res.reserve(m / 2);
58         for (int i = 0; i < m; i += 2) { res.emplace_back(e[i ^ 1].v, e[i].v, e[i ^ 1].c, Cost(e[i ^ 1].c) * e[i].f); }
59         return res;
60     }
61 };

```

## 2SAT

```

1 struct TwoSat {
2     int n;
3     vector<vector<int>>> e;
4     vector<bool> ans;
5
6     TwoSat(int _n) : n(_n * 2), e(_n * 2), ans(_n) {};
7
8     void addClause(int u, bool x, int v, bool y) {
9         e[u << 1 | !x].push_back(v << 1 | y);
10        e[v << 1 | !y].push_back(u << 1 | x);
11    }
12
13    const vector<bool>* run() {
14        vector<int> dfn(n), low(n), id(n), stk;
15        vector<bool> ins(n);
16        int ts = 0, cnt = 0, y = 0;
17        auto tarjan = y_combinator([&](auto tarjan, int u) -> void {
18            dfn[u] = low[u] = ++ ts;
19            stk.push_back(u), ins[u] = 1;
20            for (int &v : e[u]) { if (!dfn[v]) { tarjan(v);
21                low[u] = min(low[u], low[v]);
22            } else if (ins[v]) {
23                low[u] = min(low[u], dfn[v]);
24            }
25        }
26        if (dfn[u] == low[u]) {
27            cnt++; do { y = stk.back(); stk.pop_back(); ins[y] = 0, id[y] = cnt; } while (y != u);
28        }
29    }

```



```

29     });
30
31     for (int i = 0; i < n; ++i) { if (!dfn[i]) { tarjan(i); } }
32     for (int i = 0; i < n / 2; ++i) { if (id[i << 1] == id[i << 1 | 1]) { return nullptr; }
33         ans[i] = id[i << 1] > id[i << 1 | 1];
34     }
35     return & ans;
36 }
37 };

```

## 智力题

找出数组中仅出现一次的数。

其他数都出现 3 次。

$$\begin{cases} x' = x\bar{y}\bar{z} + \bar{x}yz = ? \\ y' = \bar{x}\bar{y}z + \bar{x}y\bar{z} = \bar{x}(y \oplus z) \end{cases}$$

```

1 int x{}, y{};
2 for (int z : A) {
3     tie(x, y) = tuple{ (x & ~y & ~z) | (~x & y & z), ~x & (y ^ z) };
4 }
5 return y;

```

## 子集和

每一个元素在  $2^n$  个子集出现  $2^{n-1}$  次，因此答案为  $2^{n-1} \times \sum_{i=1}^n a_i$ 。

## 子段和

考虑每一个元素前驱后继覆盖到的区间数量，两部分独立，因此答案为  $\sum_{i=1}^n i \times (n - i) \times a_i$

## 子段最值和

从目标来看，任何子段的最值至多有  $n$  种取值，因此考虑每个  $a_i$  对哪些区间产生了贡献：答案明显是两边的 NGE，可使用单调栈。

```

1 long long ans { };
2
3 auto calc = [&](auto cmp, int sign) {
4     vector stk(n + 1, 0), l(stk), r(stk);
5     int top { };
6     for (int i : ranges::iota_view(1, n + 1)) {
7         while (top and cmp(a[i], a[stk[top]])) {
8             r[stk[top --]] = i - 1;
9         }
10        l[i] = stk[top] + 1;
11        stk[++top] = i;
12    }
13    while (top) { r[stk[top --]] = n; }
14    for (int i : ranges::iota_view(1, n + 1)) {
15        ans += sign * static_cast<long long>(i - l[i] + 1) * (r[i] - i + 1) * a[i];
16    }
17 };

```

## 子段 gcd 和

每个 gcd 的贡献分为左右两部分实际上不如直接向右，因为有  $\gcd(x, y) \leq \min(x, y)$  这样的单调性，只需记下每一个 gcd 的出现次数。

```

1 #include <bits/stdc++.h>
2
3 int main() {

```

```

4     ios::sync_with_stdio(not cin.tie(nullptr));
5
6     int n;
7     cin >> n;
8     vector<int> a(n + 1);
9     for (int i = 1; i <= n; ++i) {
10         cin >> a[i];
11     }
12
13     long long ans = 0;
14     map<int, int> mp, nmp;
15     for (int i = 1; i <= n; i++) {
16         nmp.clear(); nmp[a[i]] = 1;
17         for (auto [k, v] : mp) { nmp[__gcd(a[i], k)] += v; }
18         for (auto [k, v] : nmp) { ans += static_cast<long long>(k) * v; }
19         mp = move(nmp); // mp = nmp
20     }
21
22     cout << ans << "\n";
23
24     return 0;
25 }

```

### 最少修改多少数使得原数组非严格递增

考虑贪心,  $n - \text{LIS}\{a_i\}$  即为所求。其中 LIS 非严格。

### 最少修改多少数使得原数组严格递增

如果两数大小关系不小于间隔距离, 那么这两个点可以保留。形式化地说, 需要满足:

$$a_j - a_i \geq j - i \Rightarrow a_j - j \geq a_i - i$$

于是  $n - \text{LIS}\{a_i - i\}$  即为所求。

### 最大子段积

需要考虑负数, 因此维护两个最值。

```

1     long long m, M, ans;
2     m = M = ans = a[0];
3
4     for (int i : a | views::drop(1)) {
5         tie(m, M) = minmax({ i, i * m, i * M });
6         ans = max(ans, M);
7     }

```

## 一些题

### 对区间的异或依然是区间吗

$n$  个节点的树, 给定边权为两点点权异或和  $w_{u,v} = w_u \oplus w_v$ 。每个点权的取值范围  $l_i \leq w_i \leq r_i$ , 求满足条件的  $w_i$  的数量。

- $n \in [1, 10^5], l_i, r_i \in [1, 2^{30}]$

确定任一点即可确定整棵树, 不妨以根为中心。如果现在根节点是  $x \in [l_0, r_0]$ , 需要考虑对于其他节点来说是否有:

$$w_i \in [l_i, r_i] \stackrel{?}{\Rightarrow} w_i \oplus x \in [l_i, r_i]$$

等号成立, 当且仅当  $[l, r]$  中包含  $2^k$  个数并且低  $k$  位包含  $0 \sim 2^k - 1$ 。低位包含全部的数意为这些点是 Trie 的一棵子树。用 Trie 标记出不合法的区间, 再计算该子树的贡献即可 (相当于对若干合法区间求交)。

```

1  #include <bits/stdc++.h>
2
3  using ll = long long;
4
5  using namespace std;
6
7  enum { N = 30 * 100010 };
8  int n;
9
10 int head[N], cnt, u, v, w;
11 struct {
12     int next, to, w;
13 }
14 edges[N << 1];
15 void addEdge(int u, int v, int w) {
16     edges[++cnt] = { head[u], v, w };
17     head[u] = cnt;
18 }
19
20 // Trie 部分. T tag {0/1/2} 表示 {2/1/0} 个儿子合法
21 int son[N][2], tot;
22 int T[N];
23 int newNode() {
24     ++tot;
25     T[tot] = son[tot][0] = son[tot][1] = 0;
26     return tot;
27 }
28
29 int trie(int u, int b, int l, int r, int w) {
30     // 已经不合法/这颗子树覆盖  $2^b$  棵树
31     // 即  $r - l = 2^b - 1$  下同
32     if (T[u] == 2 || r - l == ~(1 << -b)) {
33         return T[u];
34     }
35
36     for (int v: { 0, 1 }) {
37         if (!son[u][v]) {
38             son[u][v] = newNode();
39         }
40     }
41
42     // l, r 第 b 位相同
43     if (int m = 1 << b;
44         (l & m) == (r & m)) {
45         // 当前的权与 l 第 b 位相同
46         // 不相同的那一部分 l, r 也管不了直接标记为无用
47         bool st = (l & m) ^ (w & m);
48         T[son[u][!st]] = 2;
49         //  $m = 2^b$ . %m 其实就是  $(m - 1)$ 
50         T[u] = trie(son[u][st], b - 1, l % m, r % m, w) == 2 ? 2 : 1;
51     } else {
52         bool st = w & m;
53         // b 位是 1/0 时子树的情况
54         int lft = trie(son[u][st], b - 1, l % m, m - 1, w);
55         int rgt = trie(son[u][!st], b - 1, 0, r % m, w);
56
57         if (lft == 2 && rgt == 2) {
58             T[u] = 2;
59         } else if (lft != 2 || rgt != 2) {
60             T[u] = 1;
61         }
62     }
63
64     return T[u];
65 }
66
67 int L[N], R[N];
68 void dfs(int u, int p, int w) {
69     trie(1, 29, L[u], R[u], w);
70     for (int i = head[u]; i; i = edges[i].next) {
71         if (int v = edges[i].to; v != p) {

```

```

72         dfs(v, u, w ^ edges[i].w);
73     }
74 }
75 }
76
77 ll ans {};
78 void getAns(int u, int b) {
79     if (!T[u]) {
80         // 整棵树都可以用
81         ans += 1 LL << --b;
82     } else if (T[u] == 1) {
83         // 只有一边能用
84         for (int v: { 0, 1 }) {
85             if (T[son[u][v]] != 2) {
86                 getAns(son[u][v], b - 1);
87             }
88         }
89     }
90 }
91
92 int main() {
93     scanf("%d", & n);
94     for (int i = 1; i <= n; ++i) {
95         scanf("%d%d", L + i, R + i);
96     }
97
98     for (int i = 1; i < n; ++i) {
99         scanf("%d%d%d", & u, & v, & w);
100         addEdge(u, v, w);
101         addEdge(v, u, w);
102     }
103
104     newNode();
105     dfs(1, 0, 0);
106     getAns(1, 29);
107     printf("%lld", ans);
108
109     return 0 ^ 0;
110 }

```

多次询问某区间内所有区间的异或和。

按位考虑，对于每一位来说，其贡献  $s_i$  为 1 的位。于是答案为「前缀异或和数组中两两异或再求和的值」，也就是只有 01 之间才有贡献，于是每一段的贡献为：

$$2^i \times \text{cnt}_1 \times \text{cnt}_0$$

```

1  int s[bits][maxn];
2
3  for (int i = 1, x; i <= n; ++i) {
4      std::cin >> x;
5      for (int j = 0; j < bits; ++j) {
6          s[j][i] = s[j][i - 1] ^ (x >> j & 1);
7      }
8  }
9
10 for (int j = 0; j < bits; ++j) {
11     for (int i = 1; i <= n; ++i) {
12         s[j][i] += s[j][i - 1];
13     }
14 }
15
16 l -= 1;
17
18 ll ans {};
19 for (int j = 0; j < bits; ++j) {
20     int o = s[j][r] - s[j][l - 1], z = (r - l + 1) - o;
21     ans = (ans + 1 LL * o * z % mod * (1 LL << j) % mod) % mod;
22 }

```

# 数学

## Z

```
1 static constexpr int P = 1000000007;
2
3 // assume -P <= x < 2P
4 int norm(int x) { return x >= P ? x - P : x < 0 ? x + P : x; }
5 template <class T> T power(T a, int b) {
6     T res = 1;
7     for (; b; b /= 2, a *= a) {
8         if (b % 2) { res *= a; }
9     }
10    return res;
11 }
12 struct Z {
13     int x;
14     Z(int x = 0) : x(norm(x)) {}
15     Z(long long x) : x(norm(x % P)) {}
16     int val() const { return x; }
17     friend ostream &operator<<(ostream &os, const Z &x) {
18         return os << x.val();
19     }
20     Z operator-() const { return Z(norm(P - x)); }
21     Z inv() const { assert(x != 0); return power(*this, P - 2); }
22     Z &operator*=(const Z &rhs) { x = static_cast<long long>(x) * rhs.x % P; return *this; }
23     Z &operator+=(const Z &rhs) { x = norm(x + rhs.x); return *this; }
24     Z &operator-=(const Z &rhs) { x = norm(x - rhs.x); return *this; }
25     Z &operator/=(const Z &rhs) { return *this *= rhs.inv(); }
26     friend Z operator*(const Z &lhs, const Z &rhs) { Z res = lhs; res *= rhs; return res; }
27     friend Z operator+(const Z &lhs, const Z &rhs) { Z res = lhs; res += rhs; return res; }
28     friend Z operator-(const Z &lhs, const Z &rhs) { Z res = lhs; res -= rhs; return res; }
29     friend Z operator/(const Z &lhs, const Z &rhs) { Z res = lhs; res /= rhs; return res; }
30 };
```

## Poly

```
1 vector<int> rev;
2 vector<Z> roots{0, 1};
3 void dft(vector<Z> &a) {
4     int n = a.size();
5
6     if (int(rev.size()) != n) {
7         int k = __builtin_ctz(n) - 1;
8         rev.resize(n);
9         for (int i = 0; i < n; i++) { rev[i] = rev[i >> 1] >> 1 | (i & 1) << k; }
10    }
11
12    for (int i = 0; i < n; i++) { if (rev[i] < i) { swap(a[i], a[rev[i]]); } }
13    if (int(roots.size()) < n) {
14        int k = __builtin_ctz(roots.size());
15        roots.resize(n);
16        while ((1 << k) < n) {
17            Z e = power(Z(3), (P - 1) >> (k + 1));
18            for (int i = 1 << (k - 1); i < (1 << k); i++) { roots[2 * i] = roots[i]; roots[2 * i + 1] = roots[i] * e; }
19            k++;
20        }
21    }
22    for (int k = 1; k < n; k *= 2) {
23        for (int i = 0; i < n; i += 2 * k) {
24            for (int j = 0; j < k; j++) {
25                Z u = a[i + j], v = a[i + j + k] * roots[k + j];
26                a[i + j] = u + v; a[i + j + k] = u - v;
27            }
28        }
29    }
30 }
31 void idft(vector<Z> &a) {
32     int n = a.size();
33     reverse(a.begin() + 1, a.end());
```

```

34     dft(a);
35     Z inv = (1 - P) / n;
36     for (int i = 0; i < n; i++) { a[i] *= inv; }
37 }
38 struct Poly {
39     vector<Z> a;
40     Poly() {}
41     Poly(const vector<Z> &a) : a(a) {}
42     Poly(const initializer_list<Z> &a) : a(a) {}
43     int size() const { return a.size(); }
44     void resize(int n) { a.resize(n); }
45     Z operator[](int idx) const { if (idx < 0 || idx >= size()) { return 0; } return a[idx]; }
46     Z &operator[](int idx) { return a[idx]; }
47     Poly mulxk(int k) const { auto b = a; b.insert(b.begin(), k, 0); return Poly(b); }
48     Poly modxk(int k) const { k = min(k, size()); return Poly(vector<Z>(a.begin(), a.begin() + k)); }
49     Poly divxk(int k) const { if (size() <= k) { return Poly(); } return Poly(vector<Z>(a.begin() + k, a.end())); }
50     friend Poly operator+(const Poly &a, const Poly &b) {
51         vector<Z> res(max(a.size(), b.size()));
52         for (int i = 0; i < int(res.size()); i++) { res[i] = a[i] + b[i]; }
53         return Poly(res);
54     }
55     friend Poly operator-(const Poly &a, const Poly &b) {
56         vector<Z> res(max(a.size(), b.size()));
57         for (int i = 0; i < int(res.size()); i++) { res[i] = a[i] - b[i]; }
58         return Poly(res);
59     }
60     friend Poly operator*(Poly a, Poly b) {
61         if (a.size() == 0 || b.size() == 0) { return Poly(); }
62         int sz = 1, tot = a.size() + b.size() - 1;
63         while (sz < tot) sz *= 2;
64         a.a.resize(sz); b.a.resize(sz);
65         dft(a.a); dft(b.a);
66         for (int i = 0; i < sz; ++i) { a.a[i] = a[i] * b[i]; }
67         idft(a.a);
68         a.a.resize(tot); return a;
69     }
70     friend Poly operator*(Z a, Poly b) {
71         for (int i = 0; i < int(b.size()); i++) { b[i] *= a; }
72         return b;
73     }
74     friend Poly operator*(Poly a, Z b) {
75         for (int i = 0; i < int(a.size()); i++) { a[i] *= b; }
76         return a;
77     }
78     Poly &operator+=(Poly b) { return (*this) = (*this) + b; }
79     Poly &operator-=(Poly b) { return (*this) = (*this) - b; }
80     Poly &operator*=(Poly b) { return (*this) = (*this) * b; }
81     Poly deriv() const {
82         if (a.empty()) { return Poly(); }
83         vector<Z> res(size() - 1);
84         for (int i = 0; i < size() - 1; ++i) { res[i] = (i + 1) * a[i + 1]; }
85         return Poly(res);
86     }
87     Poly integr() const {
88         vector<Z> res(size() + 1);
89         for (int i = 0; i < size(); ++i) { res[i + 1] = a[i] / (i + 1); }
90         return Poly(res);
91     }
92     Poly inv(int m) const {
93         Poly x{a[0].inv()};
94         int k = 1;
95         while (k < m) { k *= 2; x = (x * (Poly{2} - modxk(k) * x)).modxk(k); }
96         return x.modxk(m);
97     }
98     Poly log(int m) const { return (deriv() * inv(m)).integr().modxk(m); }
99     Poly exp(int m) const {
100         Poly x{1};
101         int k = 1;
102         while (k < m) { k *= 2; x = (x * (Poly{1} - x.log(k) + modxk(k))).modxk(k); }
103         return x.modxk(m);
104     }

```

```

105 Poly pow(int k, int m) const {
106     int i = 0;
107     while (i < size() && a[i].val() == 0) { i++; }
108     if (i == size() || 1LL * i * k >= m) { return Poly(vector<Z>(m)); }
109     Z v = a[i];
110     auto f = divxk(i) * v.inv();
111     return (f.log(m - i * k) * k).exp(m - i * k).mulxk(i * k) * power(v, k);
112 }
113 Poly sqrt(int m) const {
114     Poly x{1}; int k = 1;
115     while (k < m) { k *= 2; x = (x + (modxk(k) * x.inv(k)).modxk(k)) * ((P + 1) / 2); }
116     return x.modxk(m);
117 }
118 Poly mulT(Poly b) const {
119     if (b.size() == 0) { return Poly(); }
120     int n = b.size();
121     reverse(b.a.begin(), b.a.end());
122     return ((*this) * b).divxk(n - 1);
123 }
124 vector<Z> eval(vector<Z> x) const {
125     if (size() == 0) { return vector<Z>(x.size(), 0); }
126     const int n = max(int(x.size()), size());
127     vector<Poly> q(4 * n);
128     vector<Z> ans(x.size());
129     x.resize(n);
130     function<void(int, int, int)> build = [&](int p, int l, int r) {
131         if (r - l == 1) {
132             q[p] = Poly{1, -x[l]};
133         } else {
134             int m = (l + r) / 2; build(2 * p, l, m); build(2 * p + 1, m, r); q[p] = q[2 * p] * q[2 * p + 1];
135         }
136     };
137     build(1, 0, n);
138     function<void(int, int, int, const Poly &)> work = [&](int p, int l, int r, const Poly &num) {
139         if (r - l == 1) { if (l < int(ans.size())) { ans[l] = num[0]; }
140         } else {
141             int m = (l + r) / 2;
142             work(2 * p, l, m, num.mulT(q[2 * p + 1]).modxk(m - l));
143             work(2 * p + 1, m, r, num.mulT(q[2 * p]).modxk(r - m));
144         }
145     };
146     work(1, 0, n, mulT(q[1].inv(n)));
147     return ans;
148 }
149 };

```

## 杂项

```

1 template <class... Args> auto ndvector(size_t n, Args &&...args) {
2     if constexpr (sizeof...(args) == 1) {
3         return vector(n, args...);
4     } else {
5         return vector(n, ndvector(args...));
6     }
7 }
8
9 template <class Fun> struct y_combinator_result {
10     Fun fun_;
11     template <class T>
12     explicit y_combinator_result(T &&fun) : fun_(forward<T>(fun)) {}
13     template <class... Args> decltype(auto) operator()(Args &&...args) {
14         return fun_(ref(*this), forward<Args>(args)...);
15     }
16 };
17
18 template <class Fun> decltype(auto) y_combinator(Fun &&fun) {
19     return y_combinator_result<decay_t<Fun>>(forward<Fun>(fun));
20 }
21
22 template <typename... Args>

```

```

23 void log(const Args &...args) { ((clog << args << ", "), ...); }
24
25 template <typename... Ts>
26 void dump(const tuple<Ts...> &tp) {
27     apply([](const auto &...args) { ((cout << args << " "), ...); }, tp);
28 }

```

## 随机数

```

1 mt19937 rng(chrono::steady_clock::now().time_since_epoch().count());
2 auto rd = bind(uniform_real_distribution<double>(0, 1), rng);
3 auto rd2 = bind(uniform_int_distribution<int>(_1, _2), rng);

```

## 浮点数比较

意义	写法
$a = b$	<code>fabs(a - b) &lt; epsilon</code>
$a \neq b$	<code>fabs(a - b) &gt; epsilon</code>
$a < b$	<code>a - b &lt; - epsilon</code>
$a \leq b$	<code>a - b &lt; epsilon</code>
$a > b$	<code>a - b &gt; epsilon</code>
$a \geq b$	<code>a - b &gt; - epsilon</code>

## 二维偏序排序规则

除第一维是  $\geq$  外，其余情况还需满足可重。否则退化为普通情形，即直接按照符号（不带等号）排。

第一维	x
$\leq$	默认
$<$	第二维逆序排序
$\geq$	第一维逆序排序
$>$	两维都逆序排序

第二维	y
$\leq$	默认
$<$	查询时使用 <code>'query(x - 1)</code>
$\geq$	离散化时逆序排序
$>$	结合前两者

## G

- ☐ 有需要开 long long 的地方吗?
- ☐ BFS 边界写对了吗?
- ☐ 调试信息都删干净了吗?