

```

1 #include "../bits/stdc++.h"
2 // http://kazuma8128.hatenablog.com/entry/2018/05/06/022654
3 // verified(insert, erase, k-th element): https://atcoder.jp/contests/arc033/submissions/4648138
4 // verified(xor minimum): https://codeforces.com/contest/947/submission/51609141
5 class BinaryTrie
6 {
7     using T = unsigned;
8     // bit 数
9     static const int B = 32;
10    struct Node
11    {
12        // 要素数
13        int cnt;
14        Node *ch[2];
15        Node() : cnt(0)
16        {
17            ch[0] = nullptr, ch[1] = nullptr;
18        }
19    };
20
21    Node *root;
22
23    // Node t 以下に 値 val を追加
24    Node *add(Node *t, T val, int b = B - 1)
25    {
26        if (!t)
27            t = new Node;
28        t->cnt += 1;
29        if (b < 0)
30            return t;
31        bool f = (val >> (T)b) & (T)1;
32        t->ch[f] = add(t->ch[f], val, b - 1);
33        return t;
34    }
35    // Node t 以下から値 val を削除
36    Node *sub(Node *t, T val, int b = B - 1)
37    {
38        // 要素が無ければエラー
39        assert(t);
40        t->cnt -= 1;
41        if (t->cnt == 0)
42            return nullptr;
43        if (b < 0)
44            return t;
45        bool f = (val >> (T)b) & (T)1;
46        t->ch[f] = sub(t->ch[f], val, b - 1);
47        return t;
48    }
49    // Node t 以下の k-th element を取得
50    T get(Node *t, int k, int b = B - 1) const
51    {
52        assert(t);
53        if (b < 0)
54            return 0;
55        int m = t->ch[0] ? t->ch[0]->cnt : 0;
56        return k < m ? get(t->ch[0], k, b - 1) : get(t->ch[1], k - m, b - 1) | ((T)1 << (T)b);
57    }
58    // Node t 以下で val と xor するときの最小値取得
59    T getMinimum(Node *t, T val, int b = B - 1) const
60    {
61        assert(t);
62        if (b < 0)
63            return 0;
64        bool f = (val >> (T)b) & (T)1;
65        f ^= !t->ch[f];
66        return getMinimum(t->ch[f], val, b - 1) | ((T)f << (T)b);
67    }
68    // Node t 以下の val 未満の要素数
69    int countLessThan(Node *t, T val, int b = B - 1) const
70    {
71        if (!t || b < 0)
72            return 0;
73        bool f = (val >> (T)b) & (T)1;
74        return (f && t->ch[0] ? t->ch[0]->cnt : 0) + countLessThan(t->ch[f], val, b - 1);
75    }
76
77    public:
78    BinaryTrie() : root(nullptr) {}
79    int size() const
80    {
81        return root ? root->cnt : 0;
82    }
83    bool empty() const
84    {
85        return !root;
86    }
87    // val を集合に1つ追加
88    void insert(T val)
89    {
90        root = add(root, val);
91    }
92    // val を集合から1つ削除(なければRE)
93    void erase(T val)
94    {
95        root = sub(root, val);
96    }

```

```
96     }
97     // 集合内の最大値
98     T maximumElement(T mask = 0) const
99     {
100         return getMinimum(root, ~mask);
101     }
102     // 集合内の最小値
103     T minimumElement(T mask = 0) const
104     {
105         return getMinimum(root, mask);
106     }
107     // 集合内で val 以上の要素の最小index
108     int lowerBound(T val) const
109     {
110         return countLessThan(root, val);
111     }
112     // 集合内で val より大きい要素の最小index
113     int upperBound(T val) const
114     {
115         return countLessThan(root, val + 1);
116     }
117     // k-th element(0-indexed) を取得
118     T get(int k) const
119     {
120         assert(0 <= k && k < size());
121         return get(root, k);
122     }
123     // val の個数
124     int count(T val) const
125     {
126         if (!root)
127             return 0;
128         Node *t = root;
129         for (int i = B - 1; i >= 0; i--)
130         {
131             t = t->ch[(val >> (T)i) & (T)1];
132             if (!t)
133                 return 0;
134         }
135         return t->cnt;
136     }
137 };
138
```