

```
1 class Trie2 {
2 private:
3     struct node {
4         int val; node* childs[2]; int deg; node *parent; int cnt;
5     };
6     int height;
7     node *root;
8     bool multi_flag;
9     bool erase_leaf(node *ptr) {
10         if (ptr->deg != height) return false;
11         while (ptr != root) {
12             bool v = ptr->val;
13             ptr->cnt--;
14             ptr = ptr->parent;
15             if (ptr->childs[v]->cnt == 0) {
16                 delete(ptr->childs[v]);
17                 ptr->childs[v] = nullptr;
18             }
19         }
20         ptr->cnt--;
21         return true;
22     }
23     int lower_bit(int x, int bitp) {
24         return (x >> bitp) & 1LL;
25     }
26     int upper_bit(int x, int bitp) {
27         return (x >> (height - 1 - bitp)) & 1LL;
28     }
29 public:
30     Trie2(bool multi_flag, int height = 63) {
31         Trie2::height = height;
32         root = new node{ 0, { nullptr, nullptr }, 0, nullptr, 0 };
33         Trie2::multi_flag = multi_flag;
34     }
35     void add(int x) {
36         node *a = root;
37         Loop(i, height) {
38             int v = upper_bit(x, i);
39             if (a->childs[v] == nullptr) {
40                 node *node_buf = new node{ v, { nullptr, nullptr }, a->deg + 1, a, 0 };
41                 a->childs[v] = node_buf;
42             }
43             a->cnt++;
44             a = a->childs[v];
45         }
46         a->cnt++;
47         if (!multi_flag && a->cnt >= 2) erase_leaf(a);
48         return;
49     }
50     bool find(int x) {
51         node *a = root;
52         Loop(i, height) {
53             int v = upper_bit(x, i);
54             if (a->childs[v] == nullptr) return false;
55             else a = a->childs[v];
56         }
57         return true;
58     }
59     bool erase(int x) {
60         node *a = root;
61         Loop(i, height) {
62             bool v = upper_bit(x, i);
63             if (a->childs[v] == nullptr) return false;
64             else a = a->childs[v];
65         }
66         return erase_leaf(a);
67     }
68     int prior_find(int x) {
69         node *a = root;
70         if (a->cnt == 0) return -1;
71         int ret = 0;
```

```
72     Loop(i, height) {
73         int v = upper_bit(x, i);
74         if (a->childs[v] == nullptr) v ^= 1;
75         ret += ((ll)v << (height - 1 - i));
76         a = a->childs[v];
77     }
78     return ret;
79 }
80 };
```