```cpp
class Trie2 {
private:
  struct node {
    int val; node* childs[2]; int deg; node *parent; int cnt;
  };
  int height;
  node *root;
  bool multi_flag;
  bool erase_leaf(node *ptr) {
    if (ptr->deg != height) return false;
    while (ptr != root) {
      bool v = ptr->val;
      ptr->cnt--;
      ptr = ptr->parent;
      if (ptr->childs[v]->cnt == 0) {
        delete(ptr->childs[v]);
        ptr->childs[v] = nullptr;
      }
    }
    ptr->cnt--;
    return true;
  }
  int lower_bit(ll x, int bitp) {
    return (x >> bitp) & 1LL;
  }
  int upper_bit(ll x, int bitp) {
    return (x >> (height - 1 - bitp)) & 1LL;
  }
public:
  Trie2(bool multi_flag, int height = 63) {
    Trie2::height = height;
    root = new node{ 0,{ nullptr, nullptr }, 0, nullptr, 0 };
    Trie2::multi_flag = multi_flag;
  }
  void add(ll x) {
    node *a = root;
    Loop(i, height) {
      int v = upper_bit(x, i);
      if (a->childs[v] == nullptr) {
        node *node_buf = new node{ v,{ nullptr, nullptr }, a->deg + 1, a, 0 };
        a->childs[v] = node_buf;
      }
      a->cnt++;
      a = a->childs[v];
    }
    a->cnt++;
    if (!multi_flag && a->cnt >= 2) erase_leaf(a);
    return;
  }
  bool find(ll x) {
    node *a = root;
    Loop(i, height) {
      int v = upper_bit(x, i);
      if (a->childs[v] == nullptr) return false;
      else a = a->childs[v];
    }
    return true;
  }
  bool erase(ll x) {
    node *a = root;
    Loop(i, height) {
      bool v = upper_bit(x, i);
      if (a->childs[v] == nullptr) return false;
      else a = a->childs[v];
    }
    return erase_leaf(a);
  }
  ll prior_find(ll x) {
    node *a = root;
    if (a->cnt == 0) return -1;
    ll 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  };
```