

```

1  #include <bits/stdc++.h>
2
3  using namespace std;
4
5  template<typename T>
6  class AVL;
7
8  template<typename T>
9  void delete_avl(AVL<T> *p);
10
11  // f(l, r) -> l < r
12  template<typename T>
13  class AVL {
14      T data;
15      function<bool(T, T)> f;
16      int size, height, balance;
17      AVL<T> *left, *right;
18  public:
19      AVL<T>(T data, function<bool(T, T)> f) : data(data), f(f) {
20          size = 1;
21          height = 1;
22          balance = 0;
23          left = right = nullptr;
24      }
25
26      void update() {
27          int l_height = 0, r_height = 0, l_size = 0, r_size;
28          if (left != nullptr) {
29              l_height = left->height;
30              l_size = left->size;
31          }
32          if (right != nullptr) {
33              r_height = right->height;
34              r_size = right->size;
35          }
36          height = max(l_height, r_height) + 1;
37          size = 1 + l_size + r_size;
38          balance = l_height - r_height;
39      }
40
41      void print() {
42          cerr << "(";
43          if (left != nullptr) left->print();
44          cerr << data;
45          if (right != nullptr) right->print();
46          cerr << ")";
47      }
48
49      AVL<T> *rotate_r() {
50          AVL<T> *l = left, *lr = l->right;

```

```

51     l->right = this;
52     left = lr;
53     update();
54     l->update();
55     return l;
56 }
57
58 AVL<T> *rotate_l() {
59     AVL<T> *r = right, *rl = r->left;
60     r->left = this;
61     right = rl;
62     update();
63     r->update();
64     return r;
65 }
66
67 AVL<T> *insert(T val) {
68     if (f(val, data)) {
69         if (left == nullptr) {
70             left = new AVL<T>(val, f);
71         } else {
72             left = left->insert(val);
73         }
74     } else {
75         if (right == nullptr) {
76             right = new AVL<T>(val, f);
77         } else {
78             right = right->insert(val);
79         }
80     }
81     update();
82     if (balance < -1) {
83         if (right->balance == 1) right = right->rotate_r();
84         return rotate_l();
85     } else if (balance > 1) {
86         if (left->balance == -1) left = left->rotate_l();
87         return rotate_r();
88     }
89     return this;
90 }
91
92 // バグってる
93 AVL<T> *erase(T val) {
94     if (f(val, data)) {
95         if (left != nullptr) {
96             left = left->erase(val);
97         }
98     } else if (f(data, val)) {
99         if (right != nullptr) {
100             right = right->erase(val);

```

```

101     }
102 } else {
103     // delete all
104     // delete_avl(this);
105
106     if (left != nullptr) {
107         left = left->erase(val);
108     } else if (right != nullptr) {
109         right = right->erase(val);
110     } else {
111         delete this;
112         return nullptr;
113     }
114 }
115 update();
116 if (balance < -1) {
117     if (right->balance == -1) right = right->rotate_l();
118     return rotate_l();
119 } else if (balance > 1) {
120     if (left->balance == 1) left = left->rotate_r();
121     return rotate_r();
122 }
123 return this;
124 }
125 };
126
127 template<typename T>
128 void delete_avl(AVL<T> *p) {
129     if (p == nullptr) return;
130     delete_avl<T>(p->left);
131     delete_avl<T>(p->right);
132     delete p;
133 }
134
135 using avl = AVL<int>;
136
137 int main() {
138     avl *root = new avl(5, [](int l, int r) {
139         return l < r;
140     });
141     for (int i = 0; i < 10; i++) {
142         root = root->insert(i);
143         root->print();
144         cerr << endl;
145     }
146     for (int i = 0; i < 10; i++) {
147         root = root->erase(i);
148         root->print();
149         cerr << endl;
150     }

```

```
151  return 0;  
152 }
```