```cpp
#include <bits/stdc++.h>

using namespace std;

class G {
public:
  vector<vector<int>> p, inv, sp, sinv;
  int n, m;
  vector<int> b, c, depth;
  vector<bool> a;

  G(int nn, vector<vector<int>> &np, vector<vector<int>> &ninv) {
    n = nn;
    p = np;
    inv = ninv;
    a.resize(n);
    c.resize(n);
    for (int i = 0; i < n; i++) {
      c[i] = -1;
    }
  }

  void dfs(int i) {
    if (a[i])
      return;
    a[i] = true;
    for (int j = 0; j < p[i].size(); j++) {
      if (a[p[i][j]]) {
        continue;
      }
      dfs(p[i][j]);
    }
    b.push_back(i);
  }

  void dfs2(int i, int id) {
    if (c[i] > -1)
      return;
    c[i] = id;
    for (int j = 0; j < inv[i].size(); j++) {
      if (c[inv[i][j]] > -1)
        continue;
      dfs2(inv[i][j], id);
    }
  }

  void solve() {
    for (int i = 0; i < n; i++) {
      dfs(i);
    }
```

```
51      int j = 0;
52      for (int i = n - 1; i >= 0; i--) {
53        if (c[b[i]] > -1)
54          continue;
55        dfs2(b[i], j++);
56      }
57      m = j;
58      sp.resize(m);
59      sinv.resize(m);
60      for (int i = 0; i < n; i++) {
61        for (int j = 0; j < p[i].size(); j++) {
62          if (c[i] == c[p[i][j]])
63            continue;
64          sp[c[i]].push_back(c[p[i][j]]);
65          sinv[c[p[i][j]]].push_back(c[i]);
66        }
67      }
68    }
69  };
70
71  struct SAT {
72    int n;
73    vector<pair<int, int>> conjs;
74
75    SAT() {
76      n = 0;
77    }
78
79    // a \/ b
80    void add_or(int a, int b) {
81      add_impl(-a, b);
82      add_impl(-b, a);
83    }
84
85    // a /\ b
86    void add_and(int a, int b) {
87      add_impl(a, a);
88      add_impl(b, b);
89    }
90
91    // a
92    void add_lit(int a) {
93      add_impl(a, a);
94    }
95
96    // a ^ b
97    void add_xor(int a, int b) {
98      add_or(a, b);
99      add_or(-a, -b);
100   }
```

```
101
102   // a => b
103   void add_impl(int a, int b) {
104     n = max(n, max(abs(a), abs(b)));
105     conjs.emplace_back(a, b);
106   }
107
108   // not(a \/ b)
109   void add_nor(int a, int b) {
110     add_and(-a, -b);
111   }
112
113   int f(int a) {
114     assert(a != 0);
115     if (a > 0) return a - 1;
116     else return n + abs(a) - 1;
117   }
118
119   bool solve() {
120     vector<vector<int>> p(2 * n), inv(2 * n);
121     for (auto &c : conjs) {
122       p[f(c.first)].push_back(f(c.second));
123       inv[f(c.second)].push_back(f(c.first));
124     }
125
126     G g(2 * n, p, inv);
127     g.solve();
128     bool ok = true;
129     for (int i = 0; i < n; i++) if (g.c[i] == g.c[i + n]) ok = false;
130     return ok;
131   }
132 };
133
134 int main() {
135   int n, m;
136   cin >> n >> m;
137   vector<int> l(n), r(n);
138   for (int i = 0; i < n; i++) {
139     cin >> l[i] >> r[i];
140     r[i]++;
141   }
142   SAT solver;
143
144   // p_1, .., p_2n : 左右
145   vector<int> mp(2 * n);
146   for (int i = 0; i < n; i++) {
147     mp[2 * i] = i + 1;
148     mp[2 * i + 1] = -(i + 1);
149   }
150   vector<pair<int, int>> ps(2 * n);
```

```
151   for (int i = 0; i < n; i++) {
152     ps[i * 2] = make_pair(l[i], r[i]);
153     ps[i * 2 + 1] = make_pair(m - r[i], m - l[i]);
154   }
155   for (int i = 0; i < n * 2; i++) {
156     for (int j = i + 1; j < 2 * n; j++) {
157       if (i / 2 == j / 2) continue;
158       if (!(ps[i].second <= ps[j].first || ps[j].second <= ps[i].first)) {
159         solver.add_or(-mp[i], -mp[j]);
160       }
161     }
162   }
163
164   if (solver.solve()) {
165     cout << "YES" << endl;
166   } else {
167     cout << "NO" << endl;
168   }
169
170   return 0;
171 }
172
```